

Opware Process Automation System Studio User Assistance



Process Automation System Studio

Opware™ Process Automation System, Version 7.0

Authors' Guide

For Opware Studio authors

Copyright

Copyright © 2000-2007 Opsware Inc. All Rights Reserved.

Opsware Inc. Unpublished Confidential Information. NOT for Redistribution. All Rights Reserved.

Opsware is protected by U.S. Patent Nos. 6,658,426, 6,751,702, 6,816,897, 6,763,361 and patents pending.

Opsware, SAS Web Client, Model Repository, Data Access Engine, Web Services Data Access Engine, Software Repository, Command Engine, Opsware Agent, Model Repository Multimaster Component, and Code Deployment & Rollback are trademarks and service marks of Opsware Inc. All other marks mentioned in this document are the property of their respective owners.

Additional proprietary information about third party and open source materials can be found at <http://www.opsware.com/support/sas65tpos.pdf>.

Updating documentation

Documentation enhancements are a continual project at Opsware. You can update the documentation set at any time using the following procedure (which is also available in the PAS readme file).

To obtain PAS documentation

1. On The Opsware Network Web site (<https://support1.opsware.com/support/index.php>), log in with the Opsware Network account name and password that you received when you purchased PAS.
2. On the **Support** tab, click the **Product Docs** subtab.
3. Under **Quick Jump**, click **Process Automation System**.
4. Under **Process Automation System**, click **ZIP** beside **PAS 7.0 Full Documentation Set**.
5. Extract the files in the .zip file to the appropriate locations on your system:
 - For the tutorials to run, you must store the .swf file and the .html file in the same directory.
 - To obtain the repository that reflects the state of the flow at the start of the tutorial, unzip the file Exportof<preceding_tutorial_name>.zip.
 - To obtain the scriptlet for the tutorial that includes using scriptlets, click the scriptlet .txt file name.
 - To update your Central or Studio Help:
 - a. Under **Help Files**, and then click **Studio Help File Bundle** or **Central Help File Bundle**.
 - b. In the **File Download** box appears, click either **Open** or **Save**.
 - c. Extract the files to the Opsware\PAS home directory, in either the **\Central\docs\help\Central** or **\Studio\docs\help\Studio** subdirectory, overwriting the existing file.

Where to find Help, tutorials, and more

The Process Automation System (PAS) documentation set is made up of the following:

- Help for PAS Central

PAS Central Help provides information to the following:

 - Finding and running Ops flows (also known as *PAS flows*)
 - For PAS administrators, configuring the functioning of PAS
 - Generating and viewing the information available from the outcomes of Ops flow runs

The Central Help system is also available as a PDF document in the PAS home directory, in **\Central\docs**.
- Help for PAS Studio

PAS Studio Help instructs Ops flow authors at varying levels of programming ability.

The Studio Help system is also available as a PDF document in the PAS home directory, in **\Studio\docs** directory.

- Animated tutorials for PAS Central and PAS Studio
PAS tutorials can each be completed in less than half an hour and provide basic instruction on the following:
 - In PAS Central, finding, running, and viewing information from Ops flows
 - In PAS Studio, modifying Ops flows
 The tutorials are available in the
 - Studio Welcome pane
 - PAS\Studio home directory, in the Tutorials subdirectory
 - The Opware Network
- Self-documentation for PAS operations, Ops flows, and Accelerator Packs
Self-documentation is available in the descriptions of the operations and steps that are included in the Ops flows.

This Help system and Guide

Help for PAS Studio (reproduced in PAS_StudioAuthorsGuide.pdf) provides an introduction to PAS Studio and detailed procedures that you will use to create Ops flows.

The Help breaks out into three sections, as follows:

- Quick View
- Basic Ops flow authoring
This section introduces you to PAS Studio and covers the basic tasks involved in creating Ops flows, including:
 - Importing Ops flows and Accelerator Packs
 - Finding and using Ops flows and operations
 - Creating Ops flows and Ops flow steps
 - Testing Ops flows
 - Making Ops flows available to PAS Central users
 A very effective way to learn basic Ops flow authoring tasks is to complete the PAS Studio tutorial on modifying an Ops flow before consulting this Help system.
- Advanced Ops flow authoring
This section goes into more depth in creating operations and steps, including defining inputs, moving information throughout an Ops flow, and creating rules that logically determine the course of a flow based on each step. This section also includes an introduction that explores the architecture of and data flow in an operation and discusses the uses of the various kinds of operations
- Creating IActions for Ops flow operations
This section introduces IAction programming and its uses for extending Ops flow functionality to integration with other systems and remote execution of Ops flows.

Quick View: Ops flows and how to build them

An Ops flow is a set of linked actions that automate health checks, troubleshooting or any other repetitive IT support tasks. Say 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.

Without Process Automation System (PAS), your options might be:

- Assign someone to look at the Web site every 15 minutes and, if necessary, manually publish content to the site.
- Program a monitoring system to check the site and raise events or alerts if the content isn't correct, with manual content publishing. One of your technical support personnel would have to see what is going wrong.

Or, with PAS Studio, you could author a flow to do all those tasks automatically: check the Web site periodically, create an event or alert, publish content to the site, troubleshoot and repair the underlying problem, and document the steps taken.

Let's use this example to learn about the basic concepts of an Ops flow. A manual runbook or procedure for a person to do this site check might read something like this:

Step 1. Browse to `mysite.com/mypage.htm`. If it does not contain the text "needed text" then do Step 2.

Step 2. Copy `mypage.htm` from server development1 to server production1.

Step 3. Keep track of how often the Web page has the correct text and how often you need to fix the problem

PAS Studio is where you create the Ops flows that automate the triage, diagnosis and resolution of problems, perform system and application health checks and handle repetitive maintenance procedures in your operations or data center. Just like the manual procedure, an Ops flow has steps that gather data or take actions. Each step accepts data and responds to it, the step's response differing according to what it detects. Steps can also provide complete tracking of their actions by recording the data used in each step and what took place.

Key parts of an Ops flow

Steps are the basic unit of a flow.

In addition to steps, Ops flows have several other key elements:

- **Operations** do the actual work of the flow. Step 1 in our example above uses an operation that checks a Web page to see whether it contains specific text. Step 2 uses an operation to copy a file. Steps are created from operations, which are the templates for steps. Thus, operations are the building blocks of any flow.
- **Inputs** give the operation the data that they need to act upon. Our operation to check a Web page needs to know which page to check (`mysite.com/mypage.htm`) and what text to look for ("needed text"). Our copy file operation needs a source location and a destination. Inputs can be entered by the person running the Ops flow, be set to a specific value, or obtained from information gathered by another step.

- **Responses** are the possible outcomes of the operation. Our get Web page operation has three responses: “page not found”, “text not found”, and “success” (if we got the page and it has the desired text). Our copy file operation might have just “success” and “failure”.
- **Transitions:** Our read Web page operation may need to respond differently if the Web page can’t be found, if the page is there but the text isn’t present, or if the page is there and the desired text is present. A transition leads from an operation response to one of the possible next steps, so that the operation’s response determines what the next step.

Let’s sum up the picture so far:

- An Ops flow step’s operation uses input data to perform a task, from which it obtains results.
- The operation has several possible responses, one of which is chosen depending on what its results were.
- Each response is connected by a transition to one of the possible next steps in the flow.

Thus the choice of response determines which is the next step in a particular run of the flow.

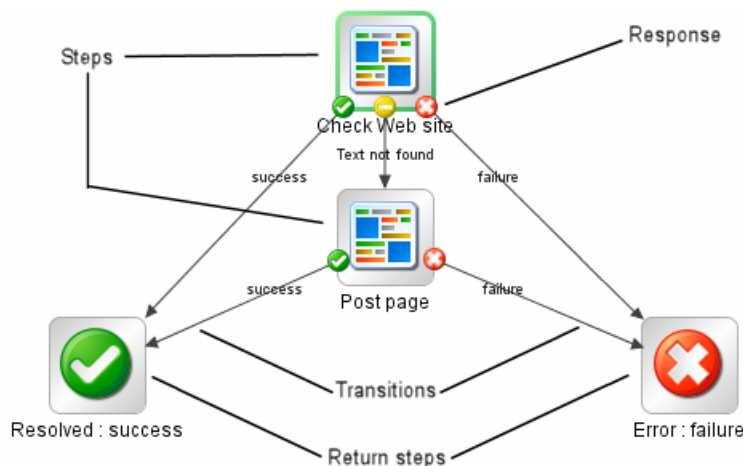


Figure 1 - Parts of a flow

Now let’s look at the main parts of an operation.

What an Ops flow needs to be valid

To be valid, an Ops flow must have the following:

- At least one step, with one of the steps designated as the start step.
- Transitions that connect each operation response to a subsequent step.
- A return step to return a value and end the Ops flow.
- Assignment (definition) of how each input gets its value.

Those are the essential features of a flow. Provide them, and you will have a productive flow. To see how you can increase a flow’s value, see the next topic, [Taking a flow beyond the basics](#) and other topics referenced therein.

Taking a flow beyond the basics

There are many dimensions that you can add to a flow's usefulness. The following list includes some of the initial ways you can enhance a flow's capabilities, robustness, and ability to return more information to the user. Some of these subjects, such as inputs and transitions, are listed in the preceding topic, but you get more out of them according to how you use them.

- Enable the flow to adapt to conditions in real time by how you assign data to inputs.
For information on the different ways you can assign data to inputs, see [Inputs: Providing data to operations](#).
- By storing data in flow variables for passing between steps in the flow or from a subflow to its parent flow (the flow in which the subflow is a step).
For creating flow variables that you can use in other steps in a flow, see [Creating a flow variable](#).
For passing flow variables to another flow, see [Passing data from a subflow to a parent flow](#).
- Capture and manipulate information that you can then use elsewhere.
For capturing data as a result and using filters to refine, manipulate, or format the data that you capture, see [Working with operation results](#).
- Tell PAS Central users what happened in each step of the flow, presenting them with information that the flow captured.
For telling Central users what happened in each step, see [Adding a transition](#).
- Record key information for charting in PAS Central Dashboards that the Central users define.
For recording information for Dashboard reporting, see

Operations: The model for steps

In addition to its responses, an operation is made up of the following:

- **Command:** This dictates most of what the operation actually does. (The operation may also contain a scriptlet, which processes data.) The command may be any of several types of commands, including command-line, an http operation, or a script to run. For example, an operation might get a directory listing, check to see if a service is running, execute a Web service, or run a Unix vmstat command.
- **Results:** When a command runs, it returns data, called results. For example, the results returned by a dir command include a file list. A ps command results are a list of processes. Most commands have more than one result, returning data such as a return code, standard output (stdout), and error output (stderr). Our get http page operation needs results for the http return code (200, 302, 404, etc) and the data on the page.

Some commands can return a lot of data that we may want to use later on in our flow. Using a single command, an operation to get memory statistics on Linux can tell us how much memory is free, how much total memory is available, how much swap memory is being used and much more.

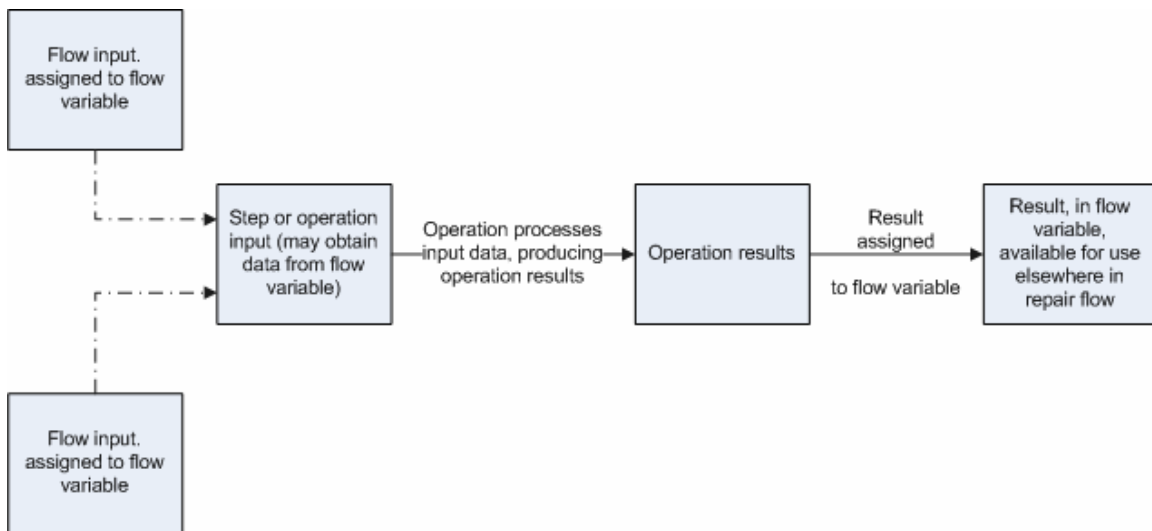
- **Flow Variables:** You can store the results data in *flow variables* that you create. You can use this data as inputs to other steps later in the flow.
- **Filters:** Figuring out what response to take based on the data that a command returns may require filtering a key piece of data out of a result or creating a scriptlet that manipulates the data.
- **Rules:** Rules evaluate the operation's results to determine which operation response to take when the step runs. Rules can evaluate any of the results fields, the data strings, return codes, or error codes.

A given response for an operation is selected when the conditions described in the response's rule match the data that you have specified in or extracted from the one of the results fields. The rules that you create are comparisons or matches between a string of text that you describe in the rule and the results field that you select for the rule.

Consider the get Web page operation in our example:

- The "page not found" response would be selected if the http return code were 404.
- The "text not found" response would be selected if text to check were not contained in the data on the page.
- **Scriptlets:** Scriptlets (written in JavaScript or Perl) 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.

The following diagram shows how operations can get values from flow, step, and their own operation inputs, and how data in operation results can be passed to flow variables, thus becoming available to other parts of the Ops flow.

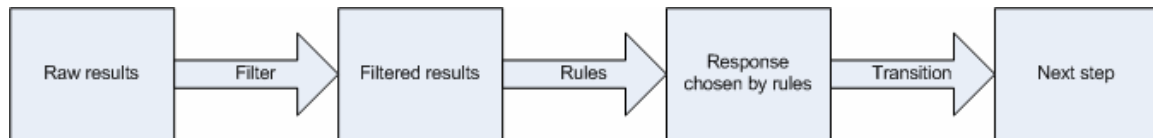


Apply this diagram to a flow that runs the Windows command-line dir command on a directory:

- Flow inputs could provide two needed pieces of data: the host computer and the name of the directory to run the dir command against
- These two input data items could be stored in flow variables named "host" and "directory," respectively.
- The operation inputs could obtain the values from the flow variables.

- After the operation performs its task based on the inputs, the step could assign the operation results to another flow variable, which another operation could use in another step in the flow.

When you create a step from an operation, each of the operation's responses must be the starting point for a transition to another step. Thus during a particular run of the Ops flow, the rules that evaluate the operation's results determine the response of the operation, which determines the transition that is followed and therefore the path that the run follows through the steps.



Note: You can also set the operation's result to supply values in fields to the result of a step created from that operation, then in turn set that step result to supply those values in fields to the Ops flow's result. You can use the Ops flow result to pass the values to other flows.



Creating or changing an Ops flow

Let's suppose that you want to check a network connection between your computer and a server. Quick View will guide you through creating and modifying a flow that pings the server and runs a traceroute command on the network between your computer and the server.

We will change the flow inputs from user prompts to specific values and make it possible for the Ops flow to run fully automatically. Our work will follow approximately these general steps:

1. Create a folder for holding the flow and its operations.
2. Get started with one of the following means:
 - Finding the Ops flow that you want to use and making a copy of the flow (see [Finding an Ops flow or operation](#)).
 - To get the Ops flow that you want to work on, you might need to import a repository. In our example you don't, but for information on importing a repository, see [Importing a repository](#).
 - Starting a new Ops flow from a template (see [Creating a new Ops flow](#)).
3. If necessary, change how values are assigned to inputs.
4. Record reporting data (identifiers for aspects of the Ops flow) for the inputs.
5. Look at the results and filters of each operation to make sure you're getting the data you need out of the operations.
6. If you need more flow variables, create them.
7. Review the flow, step, operation, and transition descriptions for usefulness to PAS Central users.
8. Test the flow.

9. To make the flow available to PAS Central users, you publish your repository.

Creating a folder

Whether you create an Ops flow or copy one for modifying, you may want to create a new folder for it.

To create a folder

1. Right-click the **Library** folder in the navigation pane and click **New Folder**.
2. In the dialog box that appears, type the name of the new folder in the text box, and then click **OK**.

Notes:

- In PAS Studio, you cannot give two folders the same name.
- Naming in Studio is not case-sensitive.
- Names can be a maximum of 128 characters long.

Creating a new Ops flow

The templates provided with PAS Studio provide steps for flows that perform certain frequently used tasks. For example, there is a template (**Restart Service**) for creating a flow to restart a service, so you could start your flow that way.

And there is a template for a flow that performs our example task: pinging a server and checking the network between your computer and a server (**Network Check**).

Or, if you want to start with only the Success and Error return steps (return steps are possible last steps of a flow), the last template (**Blank Flow**) provides you with them.

To create an Ops flow from a template

1. On the PAS Studio Welcome page, click the **New Flow** icon.
In the list of templates, when you highlight a flow template, its description appears.

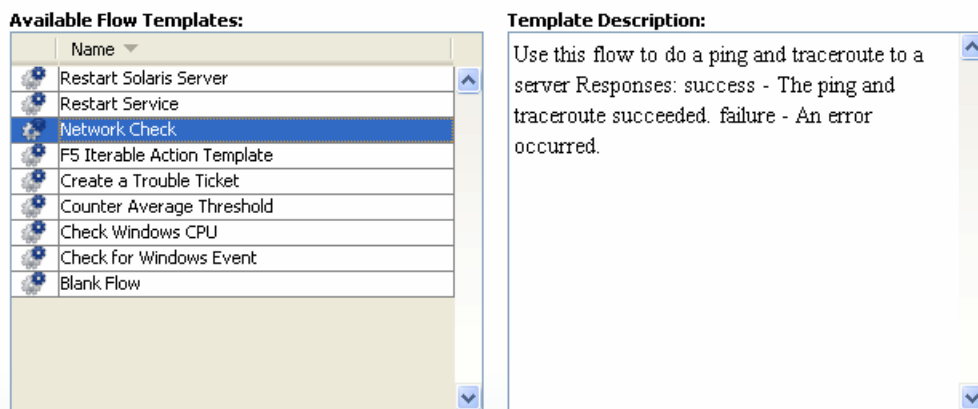


Figure 2 - Templates for commonly used flows

2. Select the flow template that meets your needs, and then click **Create**.
The flow opens in the authoring pane.

Note: Note To create a blank flow, without by right-clicking a folder, pointing to **New**, then selecting **Flow**.

Finding an Ops flow or operation

Ops flows and operations are stored in subfolders of the **Library** folder of the **Repository** pane in Studio. The main folders in the **Library** are:

- **Accelerator Packs**
The most commonly used Ops flows, organized into subfolders by technology.
- **My Ops Flows**
Where you might want to store Ops flows that you create. A new flow that you create from a template is automatically stored here. (For information on creating a flow from a template, see [Creating a new Ops flow.](#))
- **Operations**
This folder contains the essential pre-configured operations and subflows that are the building blocks of the Accelerator Packs and many of the Ops flows you will create. Many of these folders also contain a folder called “Samples” which show you examples of using many of these operations.

Note that the **Operations** folder is sealed, as indicated by the lock on the folder icon (🔒). It contains sealed (read-only) Ops flows and operations. Sealed Ops flows and operations are so widely useful and fundamentally important that PAS does not allow them to be modified. You can make copies of sealed flows and operations and modify the copies.

Tip: For quick access to Ops flows and operations that you commonly use, drag them to the **Bookmarks** pane.

For finding a flow or operation, the **Search** tab is a very useful alternative to browsing the Library.

Note: If the flow you need is in a repository that is not part of either your local PAS Studio repository or the remote PAS Central repository, you can get the flow into Studio by one of the following, depending on where the flow and its dependent objects (operations, system accounts, and other system objects that the flow uses) reside:

- Updating the local Studio repository from the Central repository
- Importing the remote repository into your local Studio repository

Searching for an Ops flow or operation

The search is a full-text search throughout the Library that uses the Apache Lucene search syntax. For information on constructing a search with the Apache Lucene syntax, see the Apache Software Foundation Web site.

Clicking (or pressing F3 on your keyboard) the **Search** tab on the bottom left of the Studio window opens the **Search** pane, on which you can find flows and operations by searching on the name or other field properties. You'll find examples farther down in this topic.



Figure 3 - Studio tabs

To keep the **Search** pane open, click the pin icon (📌) in the upper right hand corner of the pane.

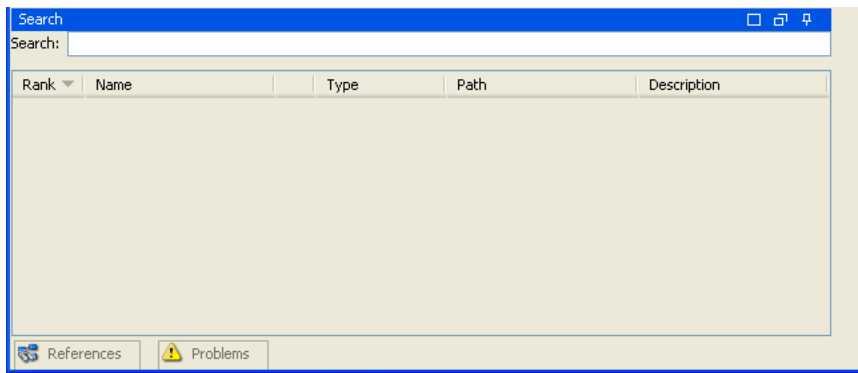


Figure 4 - Search box

For most of your searches you can simply type the search term you are interested in and press ENTER on the keyboard. For instance, to search for the Restart Service flow, in the **Search** text box, you could type **Restart Service** or just **service**. The search results are displayed with the most relevant first. You can sort by any of the columns by clicking on the column header. Note that the **Type** field will help you determine whether the search result is an operation or an Ops flow.

Rank	Name	Type	Path	Description
****	Determine Results	other	/Library/iConclude/HTTPClient/Samples	Checks the outcome of the HttpGetData...
****	Http Client Get	WebExtension	/Library/iConclude/HTTPClient	<pre>Service to perform an HTTP GET ...
***	Check URL via Http Client Get	Flow	/Library/iConclude/HTTPClient/Samples	This sample flow uses the HttpClientDet...
***	Refresh	WebExtension	/Library/Integrations/SiteScope	<pre>Issues a refresh command to a Si...
***	Acknowledge	WebExtension	/Library/Integrations/SiteScope	<pre>Acknowledges a SiteScope monito...
**	Unacknowledge	WebExtension	/Library/Integrations/SiteScope	<pre>Clears an acknowledgement of a ...
**	Total Server Shutdown	WebExtension	/Library/iConclude/Application Servers/B...	<pre>Shuts down a server and waits fo...
**	Suspend Server	WebExtension	/Library/iConclude/Application Servers/B...	<pre>Suspends a server and waits for ...
**	Start Server	WebExtension	/Library/iConclude/Application Servers/B...	<pre>starts a server and waits for it to ...
**	Validate XML Document	WebExtension	/Library/iConclude/XML Processing	<pre>Service to validate an XML docum...

Figure 5 - Search results

Note that you can read the flow or operation's description, to see whether it's the best choice.

To quickly employ an operation or flow from the list of search results

- Open an operation's **Properties** sheet or a flow's diagram by double-clicking in the row of the operation or flow of interest.

OR

Drag an operation onto a flow diagram.

To narrow your search down to particular fields

- Use the search syntax:
`<searchable_field_name>:<string to search for>`

Tip: The search uses a Boolean AND. If you type two words, the search returns only operations or flows that contain both words.

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

Examples:

```
type:cmd
```

- Category

Example:

```
categories:network
```

- Input name

Example:

```
inputs:server
```

- Flow or operation ID

Example:

```
id: 1234-3453-3242-32423
```

- String contained in flow or operation descriptions

Example:

```
description:clear
```

- RAS over which the operation or flow runs

Example:

```
ras:JRAS_Operator_Path
```

Tips:

- When you have found an operation or flow that interests you, you can learn more about how to use it by locating it in the Library, right-clicking it, and then clicking either **What Uses This** or **What Does This Use**. For more information, see [Finding out which flows use an operation](#).
- Before you search for an operation or flow from which a step was created, you can save yourself time by checking out the **Advanced** tab on the step's Inspector. The **Advanced** tab shows which operation or flow a step is associated with and the location of the flow or operation.

Descriptions: Finding the *right* operation

Suppose you want to test connectivity with a server. You do a search on the Search tab using the term "connectivity," and come up with the Connectivity Test and Iterative Connectivity Test flows. Which one is really the right flow? Let's take a look at the description for each of the two flows.

To see a flow's description

- Open the flow in the authoring pane and click **Properties** (at the bottom of the pane), then click the **Description** tab.

Note: Besides describing the flow, the description also tells you about the flow inputs, giving you hints to the kind of values to provide the inputs.

To see an operation's description


- Open the operation in the authoring pane, then click the **Description** tab.

OR

Double-click a step that was created from the operation and in the Inspector that opens, click the **Description** tab.

Note: Besides describing the flow, the description also tells you about the flow inputs, giving you hints to the kind of values to provide the inputs.

Tip: You can see operation and flow descriptions in the results area of the **Search** tab. For information on searching for an operation or flow, see [Searching for an Ops flow or operation](#).

If you want to further explore an operation, highlight the step whose operation you're interested in, open its Inspector (by clicking the  icon), and click the **Description** tab. Because the step was created from its operation, its description is the description of the operation. The description tells you about the operation's (and step's) inputs and the operation's responses and results, explaining each one. It also includes notes and tips on usage, as in the following (because the operation is sealed and so is read-only, the description is gray).

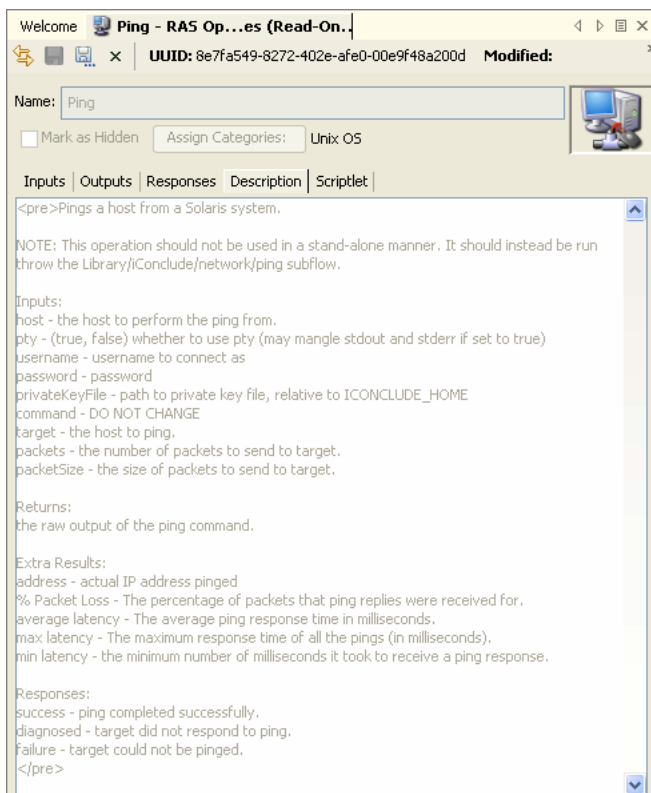


Figure 6 - Operation description

Viewing many operation descriptions

You can consolidate all the information on the operations and flows in a folder into one place, in a coherent, linked list of HTML files.

You can click any of the links in the list to see the self-documentation for a given flow or operation (or all the flows and operations within a folder). Scrolling in the right-hand pane brings up information that provides guidance on using the selected flow or operation, such as the following:

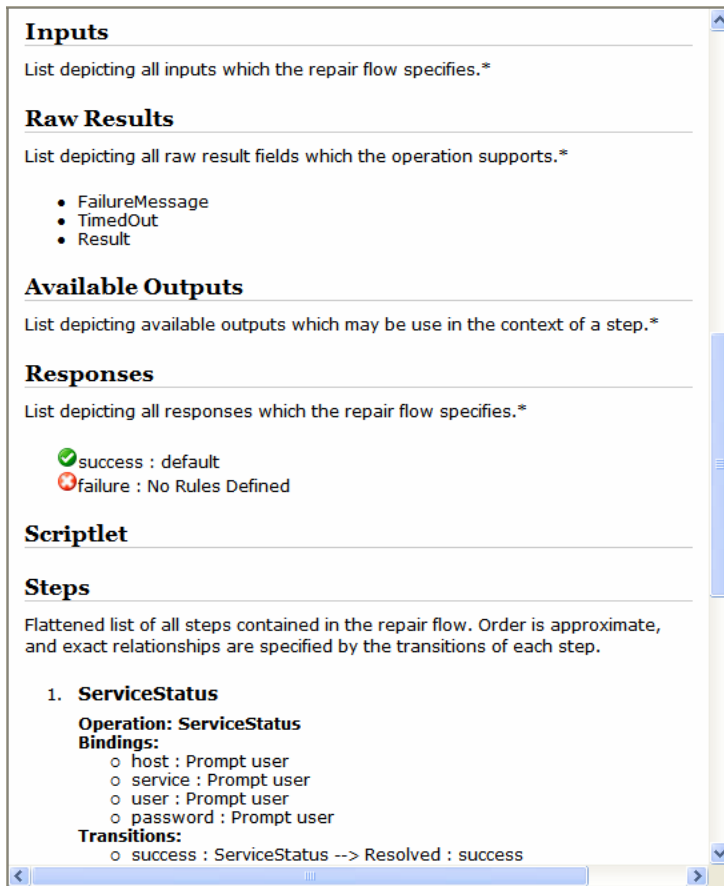


Figure 7 - Detailed information about a flow or operation

You can generate this documentation for individual operations or flows, or for the contents of folders. The documentation is generated in the form of HTML files, which are stored in a folder that you specify.

To generate documentation of the flows and operations in a folder

1. Right-click the folder whose contents you want to see information about.
2. From the context menu that appears, click **Generate Documentation**.
3. In the **Choose an output directory** dialog, specify where PAS should put the HTML files.

The folder content for which you have generated documentation appears in your Web browser.

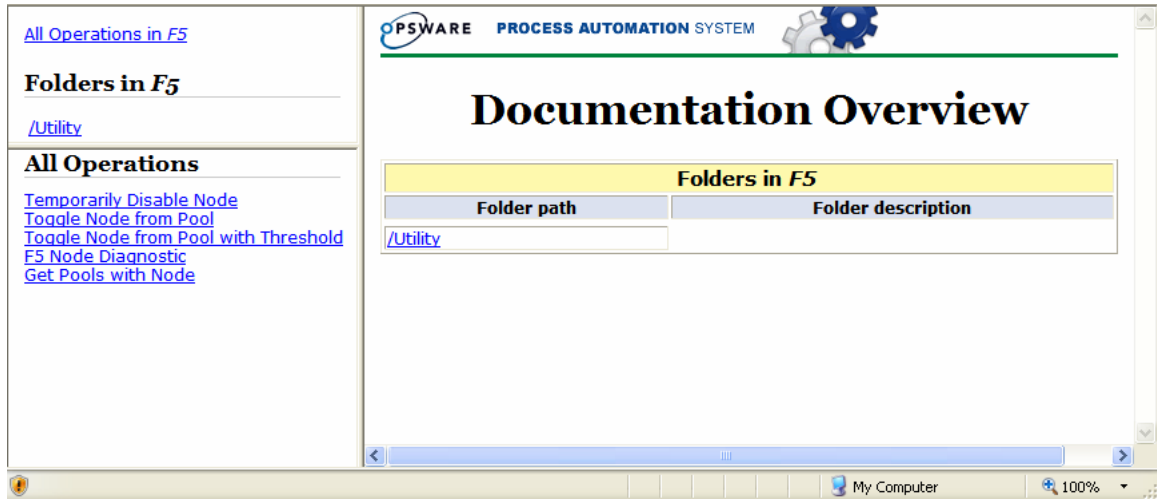


Figure 8 - Generated documentation for the F5 Accelerator Pack

4. To view the operations in a folder within the folder that you selected, click the folder.
5. To view the documentation for a single operation or flow, then under All Operations, click the operation you're interested in.

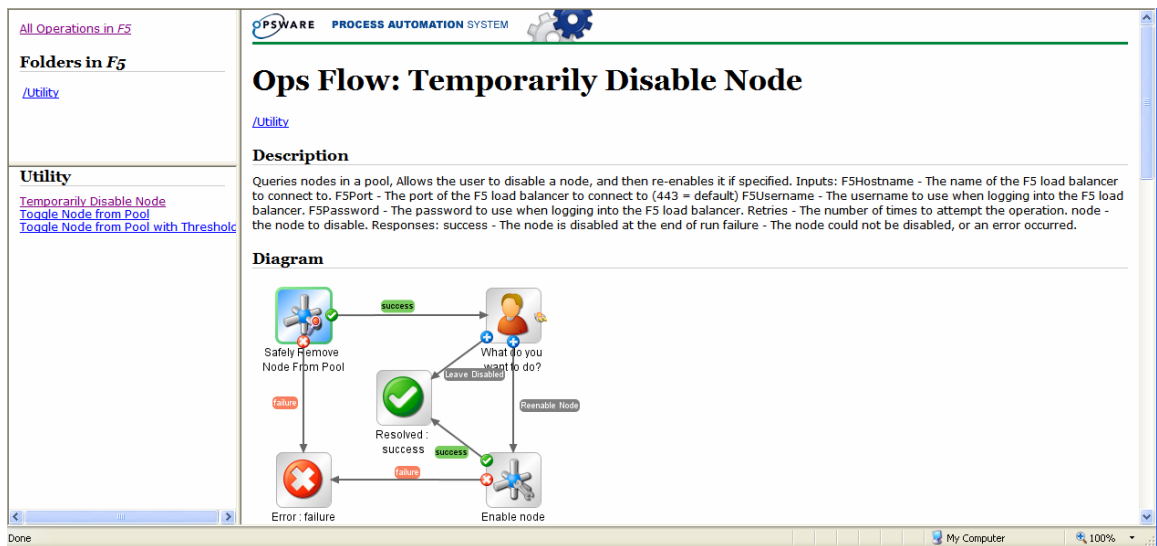


Figure 9 - Some of the information available for an operation

The illustration above shows Description and Diagram that Generate Documentation displays for an operation. In addition, you can scroll down for information on the operation's:

- Inputs
- Raw results
- Available outputs
- Responses
- Scriptlet
- If the operation is a subflow, information for each step in the subflow, including:

- The operation from which the step is created
- Values assigned to the step's inputs (these are called bindings in the **Generate Documentation** Web page)
- Transitions leading away from the step
- Outputs

Finding out which flows use an operation

You can learn more about ways to use and implement an operation or flow by looking at how it is used in existing flows.

Important: After you have created your own operations and flows, you should not change them until you have found out which other flows use them.

There are two kinds of references:

- **References to** the operation or flow. These are the flows that have a step created from the operation or flow.
- **References from** the operation or flow. These are the objects, such as selection lists, permissions assigned to groups, system filters, etc., that the 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.

These referenced flows and operations are valuable as samples that you can copy, paste, and modify.

To view an operation's references

1. In the Library, right-click the operation or flow.
2. To view the **references to** the operation or flow, click **What Uses This**.

OR

To view the **references from** the operation or flow, click **What Does This Use**.

Tip: To bring the last obtained references back up after you closed the References panel, click the **References** tab.

Copying flows and operations

When you've found the flow or operation that you want to use, best practice is to make a copy and then work on the copy. Flows and operations in the **Operations** folder of the Library are sealed and cannot be modified, so the only way to create modified versions of them is to create a copy, then make changes to the copy.

Note: If the changes you want to make are necessary only for a few uses and can be made in a step that was created from the operation, you may not need to modify the operation, but instead to make the necessary changes in the step. For more information on the differences between steps and operations, see [Advanced flow, step, and operation concepts](#).

For information on modifying steps, see the following topics:

- [Inputs: Providing data to operations](#)
- [Operation results and responses](#)
- [Changing which operation a step is based on](#)

To copy, duplicate, and/or rename flows or operations

1. In the Library, right-click the flow or operation.
2. Point to **Edit**, then point to either **Copy** or **Duplicate**.
 - If you duplicate the operation, the duplicate is automatically placed in the same folder as its original.
 - If you copy the operation, you can paste it (CTRL+V) in any folder that is not sealed.

You can copy, duplicate, and rename flows and operations with the context (right-click) menu.

Now you're ready to work with the flow.

Adding steps to the flow

There are several simple ways to create a step from an operation or flow (remember that when you create a step from a flow the flow is treated as a kind of operation).

When you create a step from an operation, the step is an instance of the operation and so inherits the operation's inputs, results, references, etc. You can change those elements of a step without affecting the operation from which the step was created.

Important! Changing inputs or results on the operation that underlies the step also changes the inputs or results for every step that is an instance of that operation. Making changes to operations can therefore break any step (and its flow) was created before you changed the inputs, etc. in the operation.

Operations should be generic enough to base many particular steps on them. Then you can make changes in the step to respond to particular situations.

To add a step to a flow

1. From the Library pane, from the Search results, or from the Bookmarks panel, drag an operation or flow onto the authoring canvas.

OR

If the operation is in the Bookmarks panel, you can:

- d. Right-click anywhere in the flow canvas.
- e. Point to **New** and **Operation**.
- f. Click the operation you're creating the step from.

You probably want to rename the step to reflect its function within the flow (operation names are more generic than their use in a particular step).

2. To rename the step, in the flow canvas, right-click the step, click **Edit Name**, and then type the new name and press ENTER.
3. Configure the step as needed:
 - Adding and defining data sources for inputs
For more information, see [Inputs: Providing data to operations](#).
 - Adding results, defining and filtering their data sources, and storing their values in flow variables, and adding responses that determine the next step in the flow
For more information on these items, see [Operation results and responses](#).
 - Adding transitions
For more information, see [Transitions: connecting responses to steps](#).

Copying steps between flows

You can save yourself authoring time by re-using a step that you created in another flow. Doing so, you might save yourself some definition of inputs, creation of filters, results, and other tasks.

To copy a step from one flow to another

1. Open the **Design** tab of the flow that has the step you want to copy.
2. Highlight the step, and then copy it using the right-click menu or the standard Windows accelerator keys for Copy (CTRL+C).
3. Open the **Design** tab of the flow where you want to use the step.
4. Paste the step where you want it.
5. Save your work.

Bookmarking flows and operations

When you add flows or operations to the **Bookmarks** panel, they are not only available in that panel but also from the right-click menu for flow canvases.

To add a flow or operation to the Bookmarks panel

- Drag the flow or operation from the Library or **Search** box to the **Bookmarks** panel.

Flow variables: Making data available wherever you need it

You can move data via flow variables by referencing a flow variable containing the data:

- In different steps in a flow or in a different flow from the one in which you created the flow variable.
Once you create a flow variable, you can then reference it (and the data that it stores) in another flow, step, or 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 the data you are testing with a response rule
To see whether an output string or error string contains a value that you have stored in a flow variable.
- In scriptlets
- In operation parameters
If an operation parameter takes a value, you can access that value by referencing a flow variable that contains it.

Creating a flow variable


Creating a step or flow input automatically creates a flow variable with the same name that has the input's value stored in it. However, you can specify that the input's value be saved to or that the input get its value from another flow variable. You can take advantage of this feature by adding several operations that act on a server and having the first operation prompt the user for the server. All the rest of the following operations that have inputs of that name will automatically use that server name.

You can create flow variables from operation, step, or flow inputs or results. The method for doing each is the same.

Note: The following procedures assume a basic knowledge of how to create inputs and step results.

- For information on creating inputs, see [Inputs: Providing data to operations](#).
- For information on creating results, see [Adding a result](#) and [Working with operation results](#).

To create a flow variable from an input

1. Open the operation's or flow's Properties sheet or the step Inspector.
 2. On the **Inputs** tab, select an input or create a new one.
 3. On the input's row, click the right-pointing arrow ().
- The input or result editor opens.
4. In the editor, select the **Assign to Flow Variable** box.
 5. Type a name for the flow variable in the text box.

OR

To name the flow variable the same as the input, leave the text box blank.

For instance, if you leave the text box blank and the name of the input is **list**, the name of the flow variable will also be **list**.

6. Save your work.

To create a flow variable from a result

1. Open the operation's or Properties sheet or the step Inspector.
2. On the **Results** tab, create a new result.
3. In the new result's row, under **Assign To**, select **Flow Variable**.
4. Under **Name**, specify the name of the new flow variable.
5. Under **From**, specify the source of the value for the flow variable.
6. If necessary, create one or more filters for the result.

For more information on creating a filter, see [Filtering an operation's raw results](#) and [Filter details](#).

7. Save your work.

Inputs: Providing data to operations

In the Network Check flow, the first step's operation pings a server, so needs the IP address of the server to ping. You provide the IP address in an input.

You can create an input to supply the IP address:

- On the flow (that is, in the Ops flow's properties).
The value of an input added to the flow can be used throughout the flow. This is recommended for data that is provided from outside the flow.
- On the step that is associated with the operation.
A step input supplies its value to the step's particular application of its operation. As a step input's value, you can assign data obtained from earlier steps in the flow.
- On the operation itself.
Data assigned to an operation input is used in each of the steps created from the operation.


Because operations are the classes from which steps are made, so steps have the same inputs (and data sources for those inputs) as the operations from which they were made. However, if you change a data assignment for a step input, then the step input's data assignment is used. Thus we can tailor the step's inputs to what you need for a particular flow.

For instance, let's return to the Network Check flow: The ping operation gets the target IP address from the **host** input (which is defined in the operation as a user prompt). When we create a step in a flow from the operation, we can change the step's host input data assignment to a specific value: the target IP address for that particular flow.

Creating an input

The procedure for creating an input is essentially the same, regardless of whether you create the input on a flow, a step, or an operation.

To create an input

1. Open the flow or operation's Properties sheet or the step's Inspector, click the **Inputs** tab.
2. Click **Add Input**, and then name the input.
3. To open the input editor for assigning a data source and defining other functionality for the input, click the right-pointing arrow at the end of the input's line ().

For information and procedures on defining how the input works, see the following:

- On assigning a data source for the input, see the following topic, [Assigning data sources to inputs](#).
 - On assigning the input's value to a flow variable, see [Creating a flow variable](#).
 - On recording the input's data for use in reporting charts in PAS Central, see [Recording values for reporting in Dashboard charts](#).
4. If you need to modify another input, scroll through the step's inputs by clicking the up or down arrow beside **Inputs Summary**.
 5. After making changes, save your work.

Assigning data sources to inputs

There are several types of data sources you assign to inputs:

- User prompt
This is the default data assignment when no other data assignment has been defined or is possible.
- From a flow variable
If there is a flow variable with the same name as the input and that flow variable stores a value, that value is assigned to the input. For example, if the input is named **list** and there is a flow variable named **list** that contains a value—say, a list—the **list** input is populated with the list flow variable’s list.
- A specific value
You can reference a flow variable as providing the specific value for the input. Doing so enables you to set up schedules that each run the flow with a different value for the input.
- No assignment
If you do not assign a data source for the input, PAS treats it as a user-prompt assignment in order to obtain a value when the flow is run.
- The previous step’s result
- A system account
- The logged-in user’s credentials

To assign a data source to a flow, step, or operation input

1. On the **Inputs** tab, click the right-pointing arrow on the input’s row to open the input’s editor.

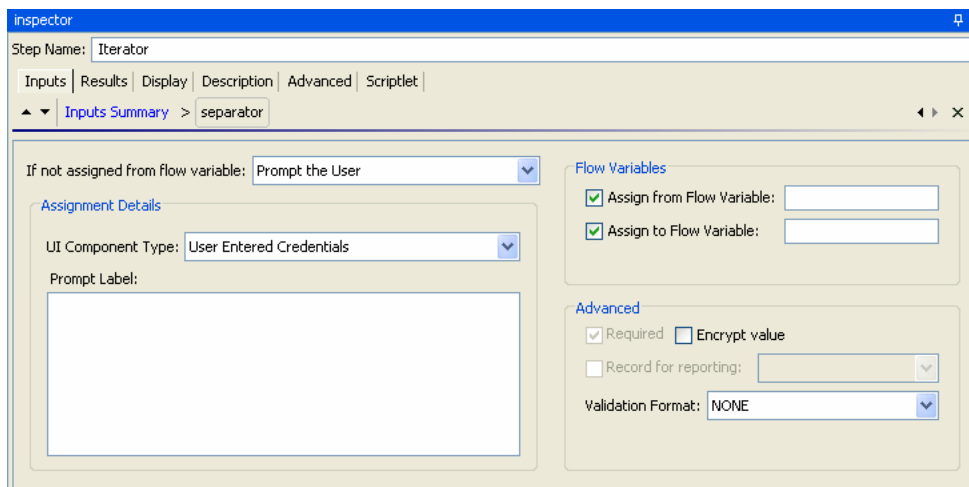


Figure 10 - Input editor for “separator” input

2. To assign the input its value from a flow variable, make sure the Assign from Flow Variable check box is selected, and then type the name of the flow variable in the text box.

OR

If the name of the source flow variable is the same as the name of the input, you can leave the text box blank.

3. To assign the input’s value from a source that is not a flow variable, unselect the **Assign from Flow Variable** box and then select a source from the **If not assigned from flow variable** box.

For more information on selecting a data source that is not a flow variable, see, in the Detailed Reference section of this Help system, [Inputs and flow variables: Providing the flow with data](#).

4. To encrypt the input's value, under **Advanced**, select the **Encrypt value** checkbox.
5. Save your work.

You can then work with another input or another step without closing the Inspector.

Inputs and scheduling Ops flow runs

To create a schedule that automatically starts runs of a flow, the flow must be able to run without needing user initiation or input. You need to change any inputs that are user prompts to specific values.

In addition, when you create schedules for a flow, the scheduling box in PAS Central enables you to store a different value in the flow variable for each schedule. For instance, you can create several schedules for the Network Check flow and point it at a different server for each schedule, by supplying a different server IP address to the host input on each schedule.

For information on creating a schedule for a flow, see Help for PAS Central.

To enable schedules to define different specific values for the input

- In the input editor, select **A Specific Value** as the data source for the input and, in the **Value** box, reference the flow variable with the syntax `${flowvariablename}`

For information on assigning a data source to an input, see [Assigning data sources to inputs](#).

Scriptlets

When you have obtained data from an operation such as ping, traceroute, or the http operation get, you probably want to test, format, manipulate, or isolate a particular piece of the results. Or perhaps you want to use a value from a flow variable to compare your results against. Using a scriptlet, you can perform these tasks and anything else that is possible with JavaScript or Perl.

You can use JavaScript or Perl scriptlets:

- In the body of an operation, step, or flow (on the **Scriptlet** tab of each).
- To filter a flow, step, or operation's results.
- As a rule for an operation's response.

For instance, in the Network Check flow, the TraceRoute operation counts how many network links, or hops, there are between the computer running the flow and the target computer. In addition being able to test the number of hops in the operation's rules to determine the operation's response, you could add the scriptlet that is used in the Evaluate Expression operation, which compares two values. With this script, you could test the number of hops with a threshold and pass the evaluation to another step or flow, for evaluation of the system's health or for communication to the user.

Following are some considerations to remember when using a scriptlet:

- As with other uses of operations, you create a scriptlet on an **operation** for a generic use that you can apply over and over.
- You create a scriptlet on a step when the scriptlet is specialized to the conditions of that step.

For instance, you might add a scriptlet to a step to evaluate and/or format data that is returned from the step's operation. If you then store the data in a flow variable and pass it to one of the fields in the flow's result and repeat this with other results, you can incrementally create a trail of the data that you've discovered.

- You might add a scriptlet to a flow (on the **Scriptlet** tab of the flow's **Properties** tab) when the flow will be used as a subflow within another flow and thus will function as an operation. After the subflow has obtained some data that you have passed to a field in the flow result, you might manipulate the data in the flow result field before passing it to the parent flow.

For more detailed information on creating and using scriptlets (including saving scriptlets for re-use), see [Using scriptlets in operations](#).

To filter step or flow results with a scriptlet

1. On the **Results** tab, open the filter editor of the result that you want to filter with a scriptlet by right-clicking the right-pointing arrow on the result's row.
2. In the filter editor, click **Add** to create a new filter.
3. In the **Select Filter** dialog, select **Scriptlet**.
4. Create the scriptlet, test the filter, and save your work.

To see the syntax for functions that help perform common tasks, click **Insert Template**. The scriptlet template provides commonly used functions and their syntax.

For more information on creating filters, see [Filtering an operation's raw results](#) and [Filter details](#).

To create a scriptlet rule for an operation response

1. On the **Responses** tab of the operation's Properties sheet, click **Add** to add a response.

OR

Right-click the right-pointing arrow on the row of the response of interest.

2. In the **Rule Type** drop-down list, select **Scriptlet**.
3. Create the scriptlet.

To see the syntax for functions that help perform common tasks, click **Insert Template**. The scriptlet template provides commonly used functions and their syntax.

4. Save your work.

To use a scriptlet from the Configuration folder

1. In the Library panel, open the Configuration\Scriptlets folder.
2. Drag the scriptlet you want to use from the folder into the scriptlet editor.
3. To change from a Configuration\Scriptlets scriptlet to one that you write yourself, click **Switch to a custom scriptlet**.

Operation results and responses

Operation results, when you apply rules called evaluators to them, determine which of several possible responses is the operation's actual response for the current run. In turn, each response is connected to another step through a transition. Thus the outcome of the evaluation of an operation's results determines which will be the next step in the flow run.


A result is part of the operation's output—success code, output string, error string, or failure message, for example. Besides evaluating them, you can pass them as data to other steps in the flow or to other flows. You can create filters to extract and modify parts of the result.

For example, suppose you only want the maximum, minimum, and average round-trip times for a ping operation to a certain server. You could extract all three pieces of information from the ping operation's raw results into several filtered results for the operation by filtering the raw results into three filtered results. Then you can use those filtered results to do the following:

- Create response rules (evaluators) that test one or more of the filtered data results to determine the next step of the flow.
- By passing the filtered data as values to flow variables, make them accessible to operations and transitions later in the flow, and through prompts that reference the flow variables, to the users of the flow.
- If the flow is a step in another flow (that is, a subflow of a parent flow), pass the data in the filtered results to fields in the flow's result, thus making the properties available to operations, steps, and transitions in the parent flow. For information on passing operation results to flow results, see [Passing operation results to flow result fields](#).

Adding a result

To add a result to an operation

1. On the **Outputs** tab of the operation, from the **Extract Primary Result From Field** drop-down list box, select one of the sources—**Code**, **Output String**, **Error String**, **Failure Message**, or **TimedOut**.
2. To create a filter or a series of filters to shape the result as needed, click **Edit Filters**.
For information on creating filters, see [Filtering an operation's results](#) and [Filter details](#).
3. Click **Add** and then, in the dialog that appears, type the name of the result.
Note: You can change the result's name or source field by changing those values in the **Name** and **Result Field** columns of the result's row on the **Outputs** tab.
4. To create filters for the result data, click the right-pointing arrow () at the end of the row.
The **Filters** column under **Available Results** tells how many filters have been created to obtain the filtered result.

Filtering an operation's results

If you're diagnosing a network connectivity problem, your goal could be to extract for use elsewhere, either in or outside of the flow, pieces of data such as the maximum, minimum, and average round-trip times, or the percent of packets that were lost.

Note: To work with a ping operation, you can either create your own ping operation or copy the existing Ping operation and work with the copy. If you are working with a copy of the existing Ping operation, you'll note that the filters you need to extract the average packet loss and maximum, minimum, and average round-trip (latency) times have already been created.

You'll probably filter an operation's raw results from the **Results** tab of the operation, although there are alternative locations in PAS Studio to do so. For instance, if you want to create a response with a rule that filters the raw result, you can do so.

To filter a raw result

1. To create filters for the result that you have added, click **Edit Filters**.

When the following filter editor appears, create one or more filters to obtain the desired value.

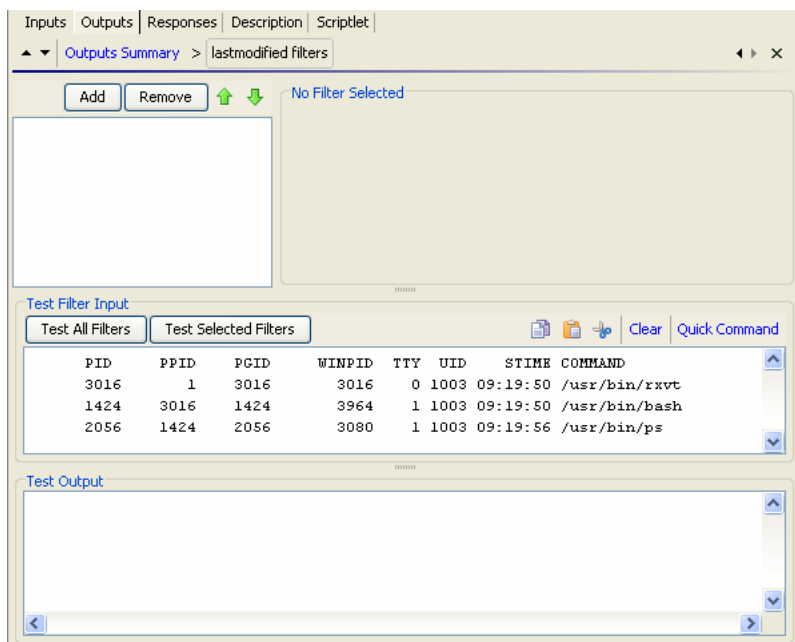


Figure 11 - Result Filter Editor

- The upper-left box builds a list of the filters as you create them.
 - When you create a filter and have selected a filter type, the upper-right box (labeled **No Filter Selected** in the above screen shot) contains controls for modifying filters depending on the kind of filter you select.
2. To add a filter, click **Add**, select the type of filter from the drop-down list in the box that appears, and define the filter's particulars in the **Details** for area at the upper-right of the filter editor.

When you create multiple filters for a result, they appear in a list. You can test all the filters in the list or just ones that you have selected. If you test multiple filters, they are run in top-to-bottom order.

Tip: While you have the filter editor open, you can click the upward- or downward-pointing arrows (▲ ▼) to change which result you're creating filters for.

3. To obtain realistic data for testing, click **Clear** to empty the **Test Filter Input** box, and then do one of the following:
 - If the data can be generated by a local command-line command, click **Quick Command** and then, in the text box that appears, type a command that generates the desired data.
 - If the data is produced by means that you cannot reproduce with a simple command-line command, you can debug the flow, highlight the relevant step, and then, from the results panel of the Studio debugger, copy the result that you want to filter and paste it into the Test Filter Input box.
4. To test one or more filters, put a sample of the data that you want to filter in the **Test Filter Input** box, then 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 (in top-to-bottom order) to the data in the **Test Filter Input** box, and the filtered results appear in the **Test Output** box.

For specifics on the kinds of filters you can create and on saving filters, see [Filter details](#).

Responses: Evaluating results




A response is one of an operation's possible outcomes. For example, an operation that runs an SQL query against a database should distinguish between its outcomes like this:


- **Failure** response if the database isn't running or can't be reached
- **No rows returned** response if the query ran but no data was returned
- **Rows returned** response if the query successfully retrieved data.

A particular response is selected when a rule or set of rules that describes a particular condition of the operation's result is true. A rule compares a value that you specify with a value in a field of an operation's raw results:


Response	Default	On-Fail	Response T...	Rules
port open	<input type="checkbox"/>	<input type="checkbox"/>	   	2 Rules. 
port listening	<input type="checkbox"/>	<input type="checkbox"/>	   	1 Rule [Source: returnCode, No Filters, No Filters] 
host not found	<input checked="" type="checkbox"/>	<input type="checkbox"/>	   	1 Rule [Source: returnCode, No Filters, No Filters] 

The above screenshot illustrates several faces of rules that you will want to remember:

- The response types are:
 - Success, or resolved: 
 - Diagnosed: 
 - No action: 

- Failure: 
- If you create more than one rule for a response (such as the **port open** 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 rules evaluate 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.
- If you specify a default response, that is the response chosen if none of the responses' rules evaluate to true.

To create a response

1. On the **Responses** tab of the operation, click **Add Response**.
2. To make a response the one chosen if an operation fails to execute, select the response's check box in the **On-Fail** column.
3. To create a rule for the response, at the right end of the response's row, click the right-pointing arrow ().
4. In the response rule editor, click **Add**.
5. In the **Apply Rule to Field** column, select the results field that has the content you want to test a rule against.
6. In the **Rule Type** column, pick the comparison or match that you want the rule to test.
7. In the **Rule Text** column, type the text to use in the test.

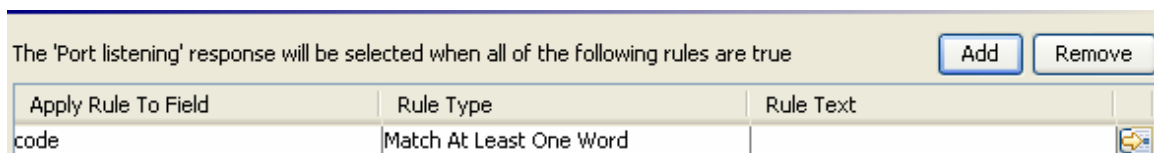



Figure 12 - Defining a rule

Using the rule editor, you can make the changes described above, or:

- Filter the operation result before applying the rule.
 - Test the rule.
 - Drag system evaluators (rules), filters, or a scriptlet into the rule.
8. To open the rule editor, at the right end of the rule's row, click the right-pointing arrow ().

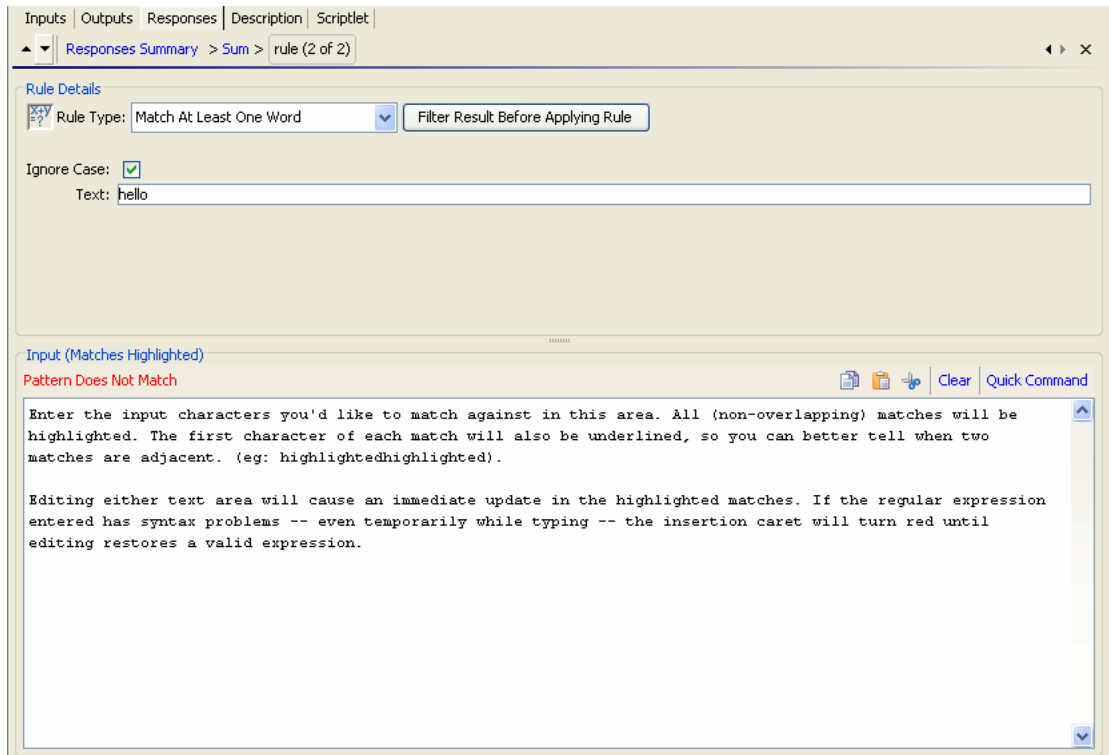


Figure 13 - Rule editor

Note: If you choose **Scriptlet** as the rule type, the rule editor changes to a scriptlet editor, as you use elsewhere. For more detailed information on creating and using scriptlets (including saving scriptlets for re-use), see [Using scriptlets in operations](#). For information on creating scriptlet filters, see [Filtering an operation's results](#).

Now you can test the rule and/or create a filter for filtering the results in the field before applying the rule.

9. To test or refine the rule, click the right-pointing arrow to open **Rule Details**. The rule type that you chose in the rule editor appears in **Rule Details** also, already selected in the **Rule Type** drop-down list.
10. In **Rule Details**, to choose another rule type, select a different one in the drop-down list.
11. 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.

OR

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 on creating Regular Expression filters, see [Filtering an operation's results](#), and for details on filter types, in Detailed Reference, [Filter details](#).

12. To work on a rule for another of the operation's responses, click the up or down arrow beside **Responses Summary**.

Tip: Response rules are evaluated in the order in which they appear in the operation's **Responses** tab. When you run an Ops flow, the response of the first rule

that evaluates to true is selected as the response of the step. So the order you provide to the responses can determine whether users obtain the desired results from the flow.

Transitions: connecting responses to steps

Transitions are the connections between each response and a subsequent step. More than one response can be connected to a given step, such as failure responses that are all connected to a single failure return step.

When a flow is run in PAS Central, descriptions for completed steps appear under **Results Summary** in the **Description** column.

Each of these descriptions is actually the description for the transition that was followed after the step completed. This way, the description can tell the user what actually happened in the completed step, rather than a generic description of what the step does. The description of what actually happened can also include data that was obtained from the step's operation (or other relevant data). You make this communication to the user possible by what you type for the transition description.

In addition to connecting steps, you can use transitions to:

- Provide information to flow users about what happened in the step that led to the transition.

To make dynamically changing data available to a transition description, you store it in a flow variable. For example, the Ping step places the host into a flow variable called "host". To use this value in the transition description you can reference it with the syntax `${host}`. A description from the success response might read "Successfully pinged `${host}`". When this is run in PAS Central against a host of "server1" the summary description will read "Successfully pinged server1".

- Control who can continue with the flow beyond the transition.

Gated transitions are transitions that require membership in a certain PAS role for the account that executes the flow. You might want to limit who can continue executing a flow beyond a certain transition for security reasons or because of who has the necessary knowledge to continue using the flow.

Gated transitions are colored red in the flow diagram in PAS Central.

- Track the value of the flow during a run.

When you assign a value to a transition, then if the transition is followed during a run of the flow, its value is added to the value of the flow for that run. The total value of the flow for that run is the sum of the values that have been assigned to the transitions that were followed.

- Quantify the value of a run of the flow.

This value, which is displayed in the PAS Central dashboard, is determined by the sum of the values for the transitions followed by an Ops flow run. The flow author sets the values of the transitions.

Adding a transition

To add a transition

1. On the flow diagram, click a response and drag to the step that that response should lead to.
2. To open the inspector for the transition, select the transition by single-clicking it and then click the Inspector tab at the bottom of the flow canvas (or double-click the transition).

The image shows a software interface window titled "inspector". It has a "Name" field containing the text "failure". Below this is a section titled "Gated Transition" which includes a checked checkbox labeled "Check user's roles before proceeding" and a dropdown menu for "Required Role" currently showing "LEVEL_ONE". Below the gated transition section is a "Flow Transition Value" field containing "\$0.00". At the bottom of the window is a section titled "Description" with a large, empty text area for input.

Figure 14 - Transition Inspector

3. In the **Description** text box, type a description.
Note: To show the flow user (in the **Summary Information** area of the **Ops Flows** tab) with a description of what happened in the preceding step that caused this transition to be followed, this is where you type a description of the step's action and outcome. You can use flow variables to store changeable information for using here.

For example, to identify a server whose name is stored in the servername flow variable, you could type, "Server {\$servername} is available for connection."
4. To limit who can run the step beyond the transition, in the **Gated Transition** area, select the **Check user's roles before proceeding** check box.
5. In the **Required Role** drop-down list, select the role that the user should be a member of in order to continue executing the flow.
6. To assign a value to the transition, in the **Flow Transition Value** box, type a value for the transition

By default, the transition name is the same value as the name of the response where it originates, but you can change the transition name to any text you want.
7. To change the transition's name, then in the **Name** text box, type the new name.

Re-arranging transitions

You might want to move and reshape transitions to tidy up your flow or to separate transitions that are in a stack. (By default, transitions that have the same origin and destination steps are stacked on each other.) You can move and reshape transitions and move transition names with two versions of clicking and dragging.

To add curve-defining points and change a transition's shape

- Position your mouse over the transition, hold down SHIFT, and drag until the transition curves the way you want it to.
If the cursor is not positioned on an existing curve-defining point, SHIFT+dragging the transition adds a new curve-defining point as well as moves the transition.

For instance, in the following flow, we'd like to move the second Iterator step's **failure** transition so that it doesn't cross any transitions.

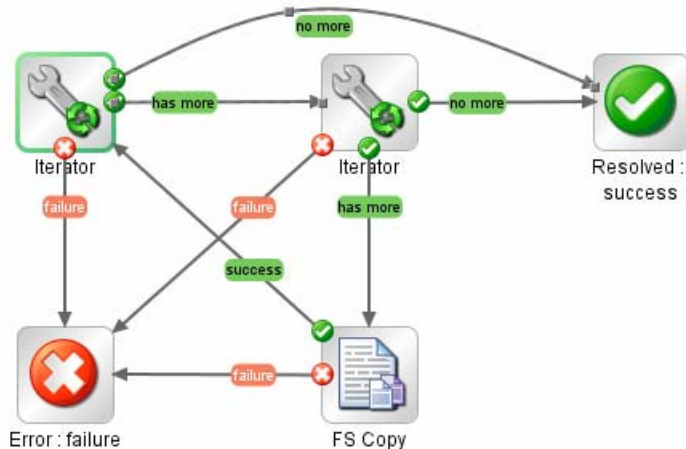


Figure 15 - Before SHIFT+dragging a transition

Using the procedure described above, we drag the transition to a new location. Note the curve-defining point that was added.

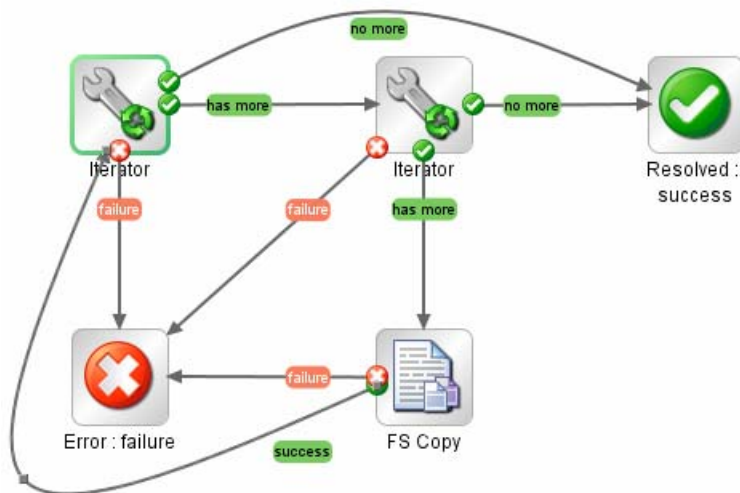


Figure 16 - After SHIFT+dragging the transition

To move a transition name

- Click the name and drag it where you want it to be.

Changing a step's icon

You can change a step's icon to one that gives you a clearer visual cue to what the step does.

To change a step's icon

1. To open the **Icons** panel, click the **Icons** tab on the right side of the flow canvas.

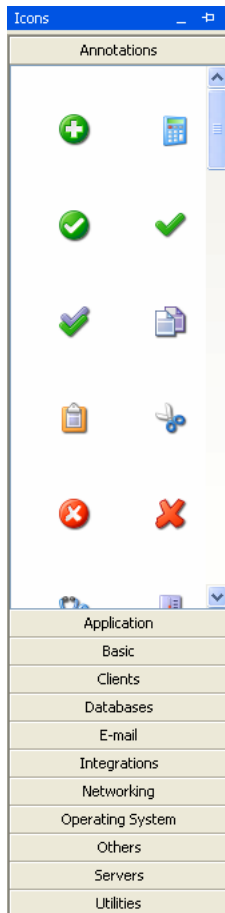


Figure 17 - Icons panel

2. In the **Icons** panel click the subpanel name that contains the icon you want, find the icon, and drag it onto the step.

Using subflows to simplify flow design

You can simplify a flow by creating steps that comprise subflows. This way, you can:

- Separate the programming tasks into smaller, more manageable pieces.
- Test them individually.
- Reuse the pieces that you create.

For example, suppose you want to copy a set of files from one server share to another. You could:

1. Create a step from the Iterator operation, providing something list inputs such as the following for the Iterator step (and specifying a semicolon [;] as the separator between the members of the list).

```
\\server1\share1\fileAAA.zip,\\server2\share2\fileAAA.zip;  
\\server1\share1\fileBBB.zip,\\server2\share2\fileBBB.zip;
```

On the first iteration, the Iterator step extracts

```
\\server1\share1\fileAAA.zip,\\server2\share2\fileAAA.zip.
```

- To use `\\server1\share1\fileAAA.zip` as the source pathname and `\\server2\share2\fileAAA.zip` as the destination pathname, you create two ListItemGrabber steps, one to extract the source pathname and one to extract the destination pathname.
Alternatively, you could simplify the flow and create a reusable piece by doing the following:
- Create a flow that consists only of the two ListItemGrabber steps and the success and failure return steps.
- Drag the flow onto the parent flow, to create a step from the subflow.

In this case and in most cases, you need to enable the parent flow to use data that have been created or modified by the subflow. Namely, a copy step in the parent flow needs each source pathname and destination pathname. See the following topic for how to pass information from a subflow to its parent flow.

Passing data from a subflow to a parent flow

Subflows usually generate data that steps in the parent flow will need to have access to. This reflects the difference between global flow variables and local flow variables. When you specify that a step input's value or a step result's value be assigned to a flow variable, they are assigned to local flow variables, which are not available outside of the subflow.

To make data from a step in the subflow available to steps in the parent flow

1. To open the subflow, double-click it in the Library pane.
2. On the subflow's authoring canvas, double-click the subflow step whose data you want to be available in the parent flow.
This opens the step Inspector for the step.
3. To keep the step Inspector open, click the pin icon (📌) in the upper-right corner of the Inspector.
4. To add a result, click the **Results** tab, and then click **Add Result**.
5. In the row that appears for configuring the result, do the following:
 - Under **Assign To**, select **Flow Output Field**.
This stores the value in an output field for the subflow, which makes the data available outside of the subflow.
 - Under **Name**, type a name for the flow output field.
 - Under **From**, select **Result Field:Result**.
6. In the parent flow's authoring canvas, open the Inspector for the step that was created from the subflow.
7. On the **Results** tab, click **Add Result** to add a result for the data that you want to make available to the parent flow.
8. In the row that appears for configuring the result, do the following:
 - Under **Assign To**, leave **Flow Variable** selected.
 - Under **Name**, type the name of the flow variable that you want the value assigned to.

- Under **From**, in the drop-down list, select **ResultField:<name>**, where **<name>** is the name of the flow output field that you typed in the result of the subflow step.

Changing which operation a step is based on

To switch the operation that the step is based on

1. Open the step's Inspector and, on the **Advanced** tab, under **operation**, click **Diff**.
2. Navigate to the operation that to base the step on, select it, and click **OK**.
3. Make any changes to the step's elements that are necessary.
You can change the step name on the step's right-click menu.

Using descriptions

A description that includes likely search words helps you or others find your flow. Suppose, for example, you modified a copy of the Network Check flow. The description "Performs a ping and a traceroute to the server," a Search tab search with the following:

```
description:ping traceroute
```

would find any flows and operations that have either or both of the words "ping" and "traceroute" in the description of the flow or one of its steps.

To create a description for the step or flow

1. On the flow's or step's **Properties** sheet, click the **Description** tab and type a description of the step.
2. Click **OK**, and then save your changes.

Enabling an Ops flow to run automatically

When a flow can run automatically, PAS Central users can schedule the flow's runs, and the flow can be started from a URL. Running automatically means that a flow does not require human intervention to start or complete, which means that it cannot have any inputs that get data from user prompts. More, if the flow's inputs get their data from flow variables, the Central user who schedules the runs can control the data that are assigned to the inputs for each run.

So, to enable a flow to run automatically:

- For any inputs that get their data from user prompts, change the data source to **A Specific Value**.
- To enable a Central user to assign inputs different data on different runs, create a flow variable and reference the flow variable for the value in the input's **A Specific Value** data source assignment.

For information on changing data sources for inputs, see [Inputs: Providing data to operations](#).

Debugging a flow in Studio

Before making a flow available in PAS Central, best practice is to test the flow in Studio.

To debug an Ops flow in Studio

1. Open the diagram of the flow you want to debug and then, in the authoring pane,

click the **Execute Flow** button ().

The Studio debugger starts.

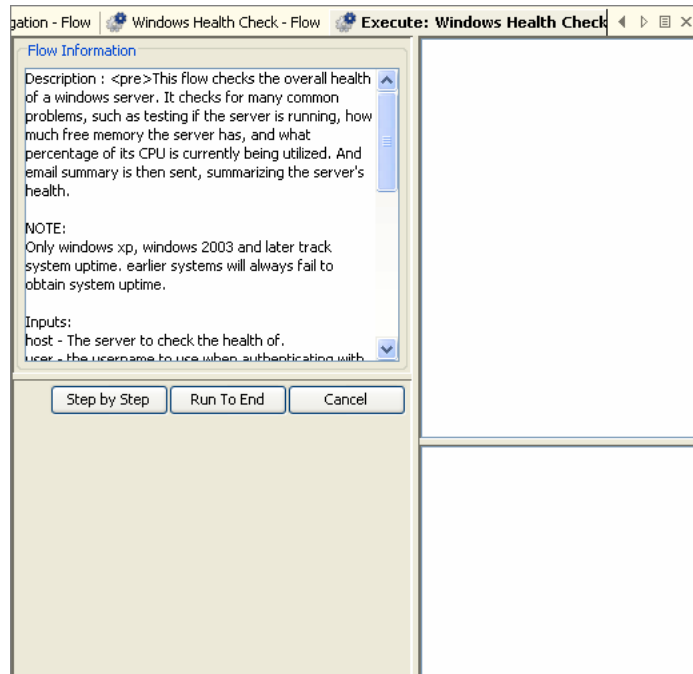


Figure 18 - Studio Flow Debugger

Tip: Note the Flow Information panel, which contains information such as the inputs for each step, which you might be prompted for. When you run the flow, this panel changes to show the current step details. However, because the debugger runs on a different tab from the flow tab, you can click the flow's tab to find a reminder of what the inputs need.

2. Click **Step By Step** or **Run to End**.

As the first step is pending or has run, the debugger changes to something like the following:

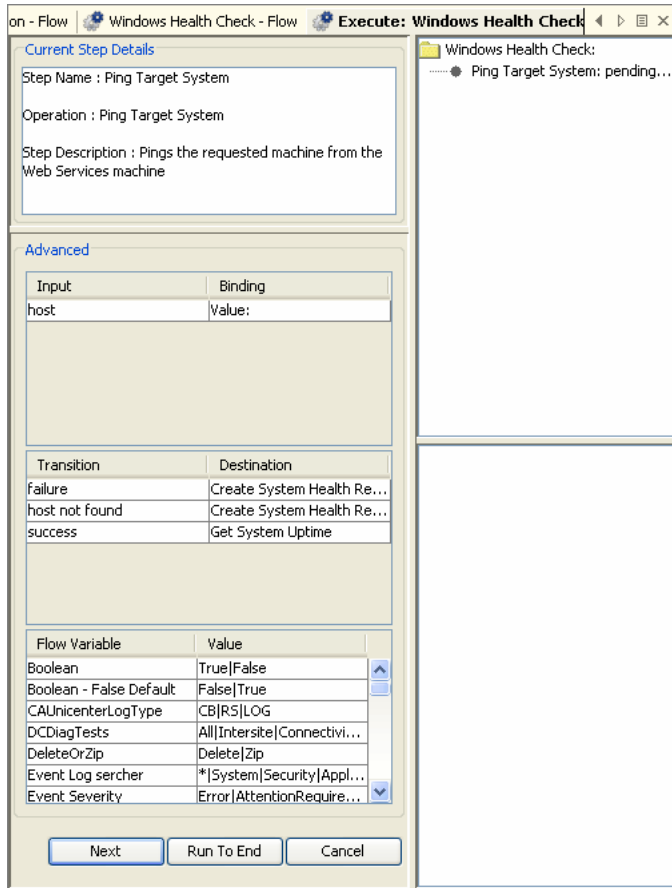


Figure 19 - Flow Debugger mid-flow

Note the classes of information whose current state the debugger reports for each step: inputs and their values, transitions, and flow variables, including system reserved flow variables, which are global flow variables that always exist and are populated, and which you can reference.

3. If you debug the flow step by step, click **Next** to test each step.
As each step in the flow and any subflows completes:
 - The flows and steps are represented hierarchically in the top-right panel.
 - Each step's results (the raw results and filtered results for the step's operations) appear in the lower-right panel.
4. To see the results of a step that has already completed, click on the step in the upper right hand pane.
If the step is a flow, the flow's description appears in the lower-right panel:

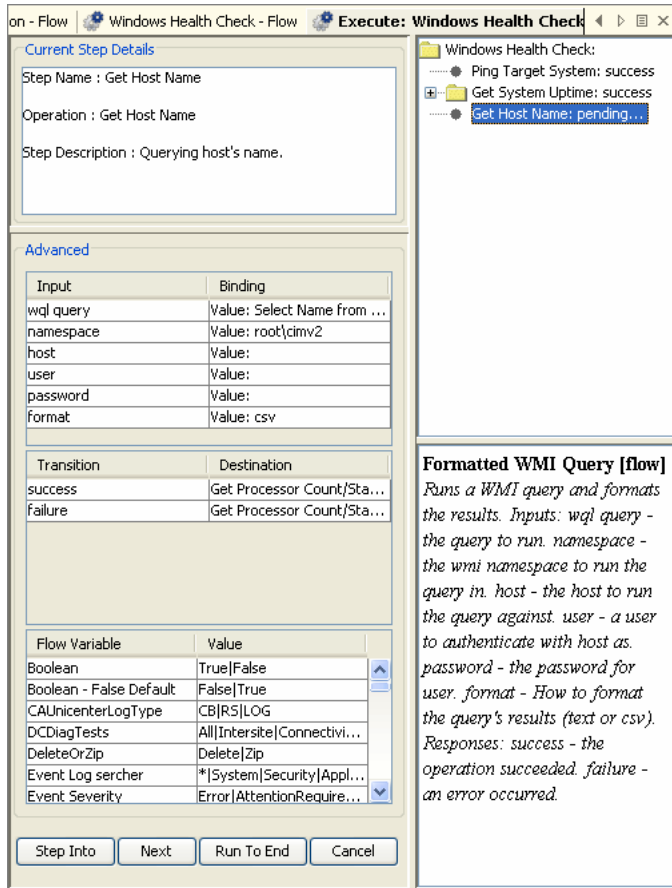


Figure 20 – Step results with description of a subflow

Repositories: Libraries for Ops flows and their objects

Ops flows, operations, and any PAS objects (domain terms, remote action services, scriptlets, selection lists, etc.) that they reference are stored in a *repository*, a structured set of XML files.

When Studio is installed, it is by default connected to the Central *public* repository. You can, however, work in a *local* repository—that is, a private repository that is local to Studio and external to the Central server. You develop flows in this local repository, then publish them to the public repository on the PAS Central server.

You can exchange work with the other flow authors on your team by publishing to and updating from the Central public repository on a staging server. Or, when one of you is offline, you can export and import selected portions of a repository.

Only the author who creates a private repository has access to it. No one can either update or publish to a private repository that you created except you.

Adding and opening a repository for authoring

You might want to work in another repository besides your default public repository in order to:

- Work in a multi-author environment.
- Keep work on one version of a flow separate from the rest of your flows.

For discussion of the scenarios in which you might want to use multiple repositories, see [Repositories: Libraries for Ops flows and their objects](#).

To work in another repository, you add it and then open it.

To add a repository

1. On the **Repository** menu, click **Add Repository**.

The following dialog appears:

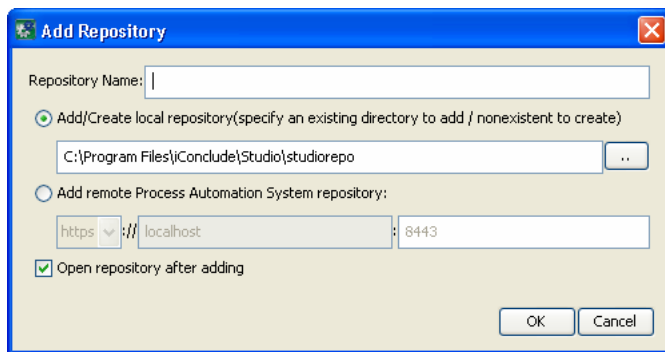


Figure 21 - Adding a repository

2. In the **Repository Name** text box, type a name for the repository
3. To add a local repository, select **Add/Create local repository** and specify a location for the repository folder.

OR

To add a repository on the PAS Central server, select **Add remote Process Automation System repository** and then:

- a. In the drop-down list of communication protocols, select the protocol that Central uses for communications.
By default, Central uses the **https** protocol.
- b. In the text boxes to the right of the list, type the Central server's name and the port that it uses for PAS communications.
By default, the port for Central when it uses the https protocol is **8443**.

4. Click **OK**.

The next time you choose a repository to open, this one will be included in the list.

To open a repository for authoring

1. On the **Repository** menu, click **Open Repository**.
2. From the list of available repositories, select the one you want to work in.

Moving flows and their elements among repositories

There are two ways to exchange flows, operations, and other objects that flows use:

- Exporting and importing repositories or parts of them is intended for exchanging flows and related objects with authors who are outside your company.
Exporting a repository creates a standalone repository. When an author imports an exported flow, he or she must manually resolve any conflicts in his repository.
When importing a repository, you can select which of its objects you want to import.
- Updating and publishing repositories or parts of them is for making flows and related objects available to other authors who share access to the same public (PAS Central) repository.
When you publish or update a folder, all the items within the folder and all the objects that are used or referenced by a flow within the folder are published or updated. That is to say, you can't be selective about which objects within a folder that you publish or update.

For procedures for importing, exporting, updating, and publishing, see the following topics in this section:

- [Setting a target repository](#)
- [Publishing a repository](#)
- [Updating from a repository](#)
- [Rolling back a publish or update](#)
- [Exporting a repository](#)
- [Importing a repository](#)

To publish or update flows, operations, and other PAS objects from one repository to another, you must first set a target repository. The repository that is not the target repository is the source repository.

Note: You cannot set the open repository as the target repository.

Because storage of the repositories resides outside of Studio, you do not need to set a target repository for exporting or importing

- When you **publish**, you are publishing **from the source** to the **target**.
- When you **update**, you are updating **from the target** to the **source**.

Setting a target repository

To set a target repository

1. If the repository that is open is the one that you want to set as the target repository, open another repository.
2. On the **Repository** menu, point to **Set Target Repository**, and then click the repository that you want to be the target.

OR

To set no target repository, click **None**.

Publishing a repository

Publishing is the copying of objects that are new or have changed from a source repository to a target repository. The source repository is the repository from which you initiate a publish or update. It must be a local repository, and the target must be the public repository of the Central that Studio is connected to.

Note: When you publish, changes that you previously published to the target that are not present in the source are not overwritten by the local (source) state of the repository. Suppose that you have two local (source) repositories, A and B, publishing to a public (target) repository, in the following sequence of events.

1. The versions of Testflow1 on local (source) repositories A and B are in sync, identical.
2. Author A adds a step to Testflow1 and publishes to the public (target) repository.
3. The public repository now has the new step.
4. The two authors agree that adding the new step was a mistake.
5. To correct the mistake in the public repository, author B tries to publish the original version of Testflow1 to the public repository.

When author B does the publish, he gets the message that there are no changes, and the Testflow1 in the public repository retains the new step.

The way to get the undesired step out of the public repository's version of Testflow1 is to do either an update preview or a publish preview. The Update/Publish Preview panel provides the option of selecting which objects you want to publish or update.

Important: When you initiate a publish, you cannot stop it or choose which objects you want to copy to the target Public repository. However:

- There is an alternative to simply publishing: You can preview the publishing action. In a publish preview, you can look at the conflicts between versions of objects and open either or both versions to examine them before selecting whether to publish the object to the target, to update the local repository from the target, or to do neither.
- You can in effect undo a publish by importing the target repository's backup. For information on how to do so, see [Rolling back a publish or update](#).

This section includes procedures for both publishing and previewing a publish.

Reminders:

- Before you try to publish or update a repository, make sure you have set a target repository and that the source repository is open. For information on setting a target repository, see [Setting a target repository](#).
- When you publish, you are publishing **from the source** to the **target**.

To publish to a repository

1. With the local repository open, set the public repository as the target repository.
2. On the **Repository** menu, click **Publish Source to Target**.
Any flows, operations, and configurable PAS objects that are new or have changed are copied to the target directory.

To preview the effects of publishing

1. With the local repository open, set the public repository as the target repository.

2. On the **Repository** menu, click **Publish Source to Target - Preview**.
OR
 - a. Click the **Publish/Update** tab at the bottom of the Studio window, and then click the pin icon (📌) to keep the **Publish/Update** panel open.
 - b. At the left end of the **Publish/Update** panel's toolbar, click either the Publish Preview icon (🌐) or the Publish & Update Preview icon (🌐🔄).

The **Publish/Update** panel lists any conflicts between the source and target versions of the two repositories' objects.

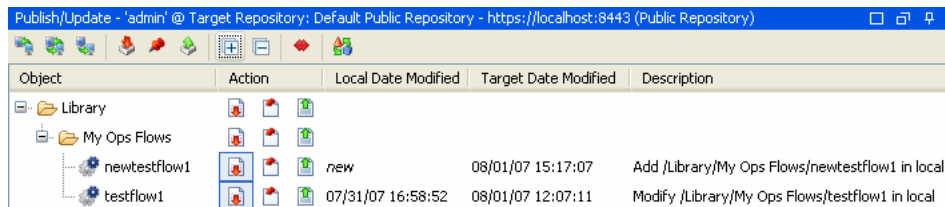


Figure 22 - Publish/Update panel

3. To completely expand each folder in the dialog box, click the plus sign (⊕).
If there is a conflict between versions of objects, you can examine the two versions of the object individually, or side by side. Note that when you preview publishing, a conflict is reported only if there have been changes to both objects of the same name in the target and the source repositories. Suppose, for instance, the following:
 - a. Local repository X and the public repository are in synch.
 - b. You import a new version of testflow1 to local repository X.
 - c. From a second local repository Y, you publish another version of testflow1 to the public repository.

Now testflow1 has changed in both local repository X and in the public repository. A conflict will be reported for testflow1 when you preview publishing from local repository X to the public repository.

In the illustration above, there are conflicting versions of testflow1 on the local and target repositories.




4. To look at either of the versions by itself, right-click in the row for the object (testflow1), and then, in the menu that appears, click **Open Local** or **Open Target**, depending on which you want to examine. The flow diagram of version that you select opens in the authoring pane of Studio, above the Publish/Update panel.
OR

To look at both versions side by side, , right-click in the row for the object (testflow1), and then click **Compare**.

In the flow diagram(s), you can make and save changes.

5. After comparing the objects and possibly making changes, choose an action for each object or folder:


Tip: Note that in the Publish & Update Preview, you can choose to publish some objects and update others.

- To change the object in the target repository, click the upward-pointing arrow .
- To delete the object or folder from the source, click the downward-pointing arrow ()
- To take no action, click the No Change icon ()

OR

Right-click the object and choose the corresponding command:

- To update the object from the target, **Modify <objectname> in local**
- To publish the object to the target, **Modify <objectname> in target**
- To take no action, **No Change**

6. To apply the changes that you've specified, click the Apply icon ()
7. Save your work.

Updating from a repository

Updating from a repository is the opposite of publishing to a repository. But the source repository is still the repository from which you initiate a publish or update. It must be a local repository, and the target must be the public repository of the Central that Studio is connected to. So when you **update**, you are updating **from the target** to the **source**.

When you initiate an update, you cannot stop it or choose which objects you want to copy from the target Public repository to your local source repository. However, you can preview the update, and you can, in effect, undo it after the fact.

- When you preview an update, you can select which PAS objects that you want to update to the local repository, and update them.
- To undo an update, you need to import the source repository's backup. For information on how to do so, see [Rolling back a publish or update](#).




To update from a repository

1. Set the repository that you want to **update from** as the target repository. For information on setting a target repository, see [Setting a target repository](#).
2. With the local (source) repository open, from the **Repository** menu, select **Update Source from Target**.

To preview the effects of updating

1. With the local repository open, set the public repository as the target repository.
2. On the **Repository** menu, click **Update Source from Target - Preview**.

OR

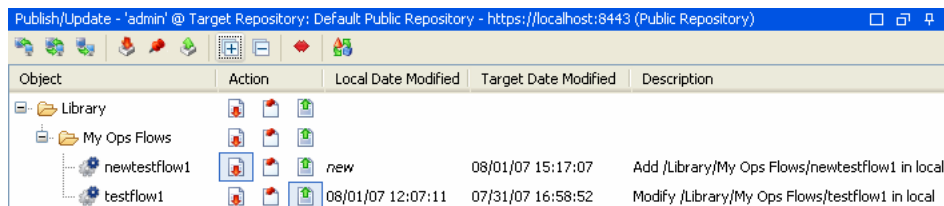
- a. Click the **Publish/Update** tab at the bottom of the Studio window, and then click the pin icon () to keep the **Publish/Update** panel open.
- b. At the left end of the **Publish/Update** panel's toolbar, click either the Update Preview icon () or the Publish & Update Preview icon ()

The **Publish/Update** panel lists any conflicts between the source and target versions of the two repositories' objects. When updating, conflicts are reported for objects that have the same name in the same location in the target and source repositories, but that have two different IDs.

Suppose, for instance, the following:


- The local repository and the public repository are in synch, and both contain testflow1.
- In the public repository, you delete testflow1 and then add another version or instance of testflow1.

Now testflow1's ID in the public repository is different from the ID it has in local repository. A conflict will be reported for testflow1 when you preview updating from the public repository to the local repository.



Object	Action	Local Date Modified	Target Date Modified	Description
Library				
My Ops Flows				
newtestflow1			08/01/07 15:17:07	Add /Library/My Ops Flows/newtestflow1 in local
testflow1		08/01/07 12:07:11	07/31/07 16:58:52	Modify /Library/My Ops Flows/testflow1 in local

Figure 23 - Publish/Update panel

3. To completely expand each folder in the dialog box, click the plus sign () .

Important: If there is a conflict between versions of objects, you can examine the two versions of the object individually, or side by side.

In the illustration above, there are conflicting versions of testflow1 on the local and target repositories.

4. To look at either of the versions by itself, right-click in the row for the object (testflow1), and then, in the menu that appears, click **Open Local** or **Open Target**, depending on which you want to examine. The flow diagram of version that you select opens in the authoring pane of Studio, above the Publish/Update panel.




OR

To look at both versions side by side, , right-click in the row for the object (testflow1), and then click **Compare**.

In the flow diagram(s), you can make and save changes.

5. After comparing the objects and possibly making changes, choose an action for each object or folder:


Tip: Note that in the Publish & Update Preview, you can choose to publish some objects and update others.

- If you are updating, click () to update the local repository with the object from the public repository.
- If you are updating, click the downward-pointing arrow () to delete object from the public repository.
- To take no action, click the No Change icon () .

OR

Right-click the object and choose the corresponding command:

- To update the object from the target, **Modify <objectname> in local**
- To publish the object to the target, **Modify <objectname> in target**
- To take no action, **No Change**

6. To apply the changes that you've specified, click the Apply icon ()
7. Save your work.

Rolling back a publish or update

If you publish to or update a repository and then realize that you did not want to make those changes to that repository, you can effectively undo or roll back the changes by opening the repository and importing its backup.

A repository is automatically backed up to a .jar file 10 minutes after a change occurs, so:

To reverse the effects of a publish or update

1. Unpack the .jar and then import it.
2. Save your work.

Exporting a repository

To make flows and PAS objects available to another author with whom you do not share a public repository, you can export the flow and its dependent objects as a repository, which other PAS Studio authors can then import.

In addition, you can back up your Ops flows and their dependent objects by exporting them to a safe backup location.

To export a repository

1. In the Library, right-click the folder that contains all the PAS objects you want to export, point to **Repository**, and then click **Export as New Repository**.
2. In the **Select Repository Directory** dialog, navigate either:
 - To an existing folder and click **Save**.
 - OR
 - Where you want a new folder to reside, type the folder name in the File name box, and then click **Save**.

In the **Exclude Items** dialog, you can choose whether to keep the following out of the export. You might want to exclude these if they are specific to one environment and you're exporting the repository for use in a different environment.

- Selection lists
Stored lists that are presented to the user.
- System properties
Global flow variables with values that never change and so can be used in many flows, saving you the time of recreating the flow variable each time you need to use it.
- System accounts
These are user credentials that are hidden behind the system account name by which they can be referenced.
- Remote Action Services (RASs)

An RAS is an instance of either the Java RAS (JRAS) or .NET RAS (NRAS) that enables an Ops flow to carry out commands outside PAS or on a remote computer, or to integrate with other application programming interfaces

3. Make sure that only the items you want to exclude are selected, then click **OK**.

Importing a repository

To obtain additional existing Ops flows and operations – say, a flow that was created by someone else or is in a new Accelerator Pack – you can import the flow as a repository. When you import a flow, you also import the flow's configurable PAS objects, such as domain terms and filters that have been saved as system filters, that are used by the flow or its operations.

Remember: When you import an Ops flow or operation, you're importing the object, not a copy or instance of it. Further, the object can only exist in one place within the repository. Therefore, an operation can reside anywhere in the library relative to the flow or flows that use it.

Note: To import Ops flows that were created or last updated in PAS Studio version 2.0, use the procedure To import a legacy Ops flow, below.

To import an Ops flow, Accelerator Pack, or repository

1. Either select or create a folder in the Library.
2. From the **Repository** menu, choose **Import Repository**.
3. In the **Select Repository Directory** dialog box that opens, navigate to the directory that contains the repository (which may be an Accelerator Pack) that you want to import, and then click **Open**.

After Studio checks for differences between the source and target repositories, the **Import Source to Target** dialog box opens.

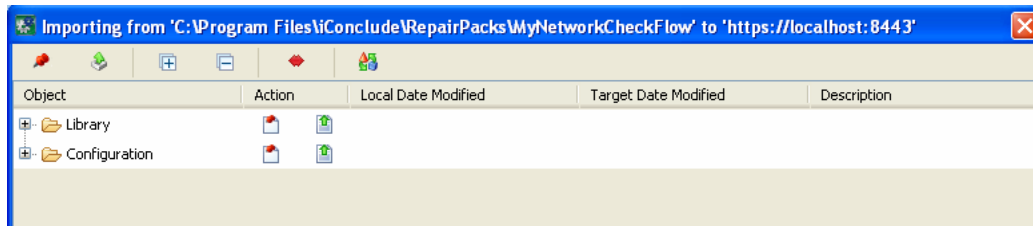



Figure 24 - Importing dialog box

This dialog lists the folders whose contents you will import.

The **Library** folder contains the flows and operations that you're importing.

The **Configuration** folder contains domain terms and system objects.

4. To completely expand each folder in the dialog box, click the plus sign () . When you expand the folders, each PAS object is described.

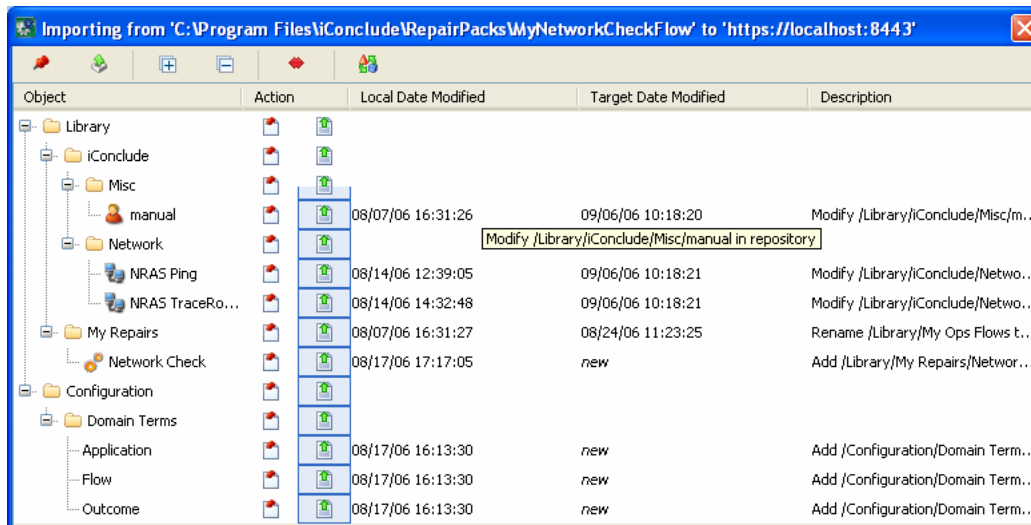



Figure 25 - Selecting objects to import

Note particularly that the dialog shows the last date of modification in the target repository for PAS objects that exist in both the target and the source.

Note: You can view only the conflicts by clicking the **View All Changes** button



- To import a folder and its contents or a PAS object to the target repository, under **Action**, click the upward-pointing arrow () for the appropriate row.

OR

To leave an object as it is in the source, click the no change icon ()

Tip: You can take the same action for all the objects within a folder by clicking the upward-pointing arrow or the no-change icon for the parent folder.

- In the toolbar of the dialog, click the apply icon ()

By default, two RASs are configured when you install PAS Studio:

- **JRAS_Operator_Path**
If you accepted the default settings when you installed PAS Studio, the URL for JRAS_Operator_Path is:
<http://localhost:8080/JRAS/services/RCAgentService>
- **NRAS_Operator_Path**
If you accepted the default settings when you installed PAS Studio, the URL for NRAS_Operator_Path is:
<http://localhost:4080/NRAS/services/RCAgentService.asmx>

For information on configuring an RAS, see [Adding or configuring a new RAS](#).

Because storage of the repositories resides outside of Studio, you do not need to set a target repository for exporting or importing

Validating repositories

For a flow to run, the flow itself, its operations, and any system accounts used in the flow must be valid. You can check an individual flow or operation for problems (by highlighting the flow or operation in the Library and then clicking the **Problems** tab). (See also [What an Ops flow needs to be valid.](#))

Or, to discover and correct in one sweep any problems that there might be, you can validate an entire repository. This validates all the flows, operations, and system accounts in the repository.

To validate a repository

- With the repository open, from the **Tools** menu, click **Validate Repository**.
A list of any problems, with their location and description, appears, as it does when you use the **Problems** tab, to guide you in repairing the problems.

Encrypting repositories

When you encrypt a repository, you are making a copy of the repository and encrypting the copy. The reason to create an encrypted copy is to protect a snapshot of your work from tampering or theft. To modify in anyway, publish to, update from, import to, or export an encrypted repository, you must type the correct password.

Notes:

- When you encrypt a repository, the original, unencrypted repository remains open in Studio.
- Once you have created an encrypted repository copy in a given location, you cannot re-encrypt a repository or encrypt another repository to the same location.

Encrypting a repository

To encrypt a repository

1. On the **Repository** menu, click **Encrypt Repository**.

The following dialog appears:

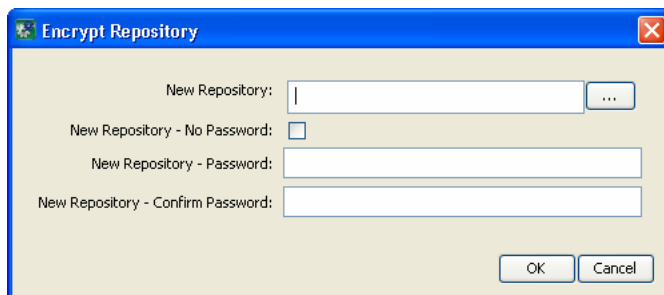


Figure 26 - Encrypting a repository

2. In **New Repository**, type a path and name for the new copy of the repository.
OR
Click the ... button to navigate to the desired location or repository and name the encrypted copy.
3. Type and confirm a password for the encrypted copy, and then click **OK**.

Opening an encrypted repository

To open an encrypted repository

1. To open the encrypted copy of a repository for the first time, first add the repository: on the **Repository** menu, click **Add Repository** and navigate to the encrypted copy.
2. When prompted for the password, supply the password that you specified when you encrypted the repository.

Decrypting a repository

To decrypt a repository

1. Open the encrypted repository.
2. From the **Repository** menu, click **Decrypt Repository**.

Suppose you want to create a second encrypted copy of the repository, with a different password. You do so by re-encrypting the repository.

Creating a second encrypted copy of a repository

To create a second encrypted copy of the repository

1. Add (if necessary) and open the encrypted repository.
2. From the **Repository** menu, click **Re-encrypt Repository**.
3. Complete the following dialog, and then click **OK**.

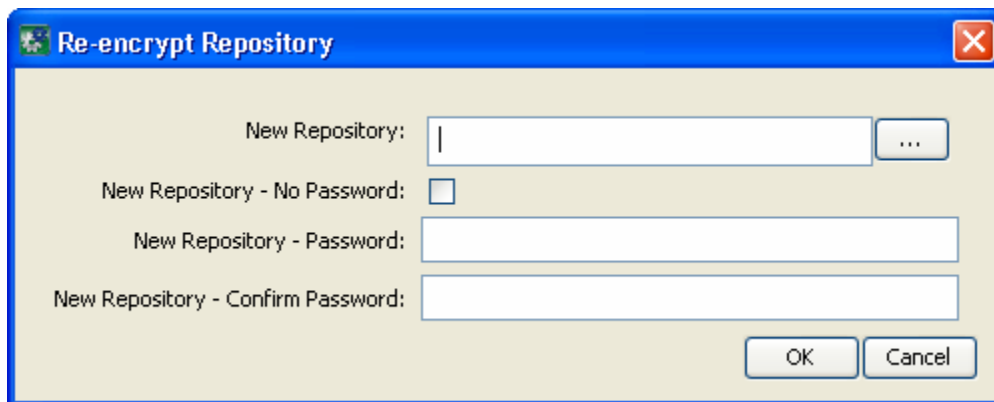


Figure 27 - Re-encrypting a repository

Detailed reference for Ops flow authoring

The Detailed Reference section of this Help system introduces you to the PAS Studio user interface with a [Visual Overview](#) and some of the [Keyboard shortcuts in Studio](#), then follows up with greater depth on how you can add function and depth to the work that your flows accomplish and the information that they make available.

Visual overview of PAS Studio

The main elements of PAS Studio are:

- The [Library pane](#), which shows the repository you're working in.
- The [Authoring pane](#), where you do the substance of your work on the Ops flow diagram; the Properties sheets for flows, steps, operations, and transitions; and editors for a number of system objects and flow objects such as selection lists, domain terms, filters, and scriptlets.
- The [Bookmarks pane](#), where you can store shortcuts to favorite operations and flows.
- The [Icons pane](#), which contains several collections of icons you can drag new icons from onto operations or steps

Library pane

Expanding the Library folder and then the Ops flow folders reveals Ops flows and operations, with symbols that characterize them.

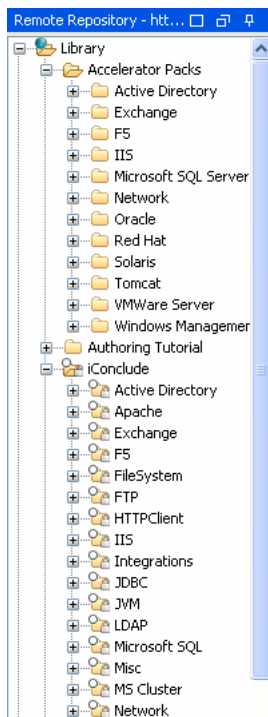


Figure 28 - Library, in Library pane

In the list of folders and Ops flows, subflows (such as **Get User Account (Auto-logon)**) are indented from their parent Ops flows (such as **Fix Auto-logon Account**).

As you explore the Library, the following table explains some of the icons you'll encounter.



This is an Ops flow.



The **white circle** means that this folder and all the flows that it contains are hidden from PAS Central users, thus do not appear in PAS Central (no operations ever appear in Central). You can hide individual flows within a folder.

The **lock** means that the folder and all the flows and operations that it contains are sealed: You cannot modify the flows and operations in this folder. You can, however, duplicate them and then make changes to the copy.



This Warning symbol is superimposed on the symbol for an Ops flow or operation that is incomplete or invalid.

In the **Library**, you can identify operation types by the icons that represent them.

Authoring pane

The authoring pane is the large, right-hand area in the Studio where you work on flow diagrams, adding steps and the connections between them to an Ops flow's diagram and setting properties that determine how flows and their parts work.

The authoring pane toolbar buttons provide shortcuts for a number of tasks. The following illustration labels the buttons that are not standard application buttons (such as cut, copy, paste, undo and redo). See the list below the diagram for more information on some of the labeled buttons.

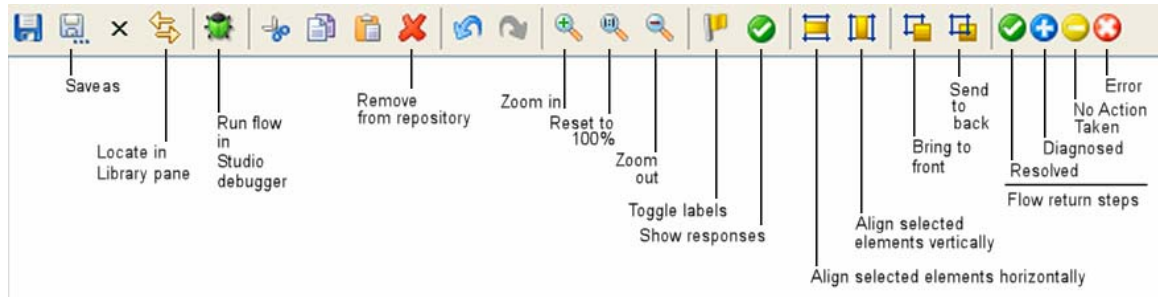


Figure 29 - Ops flow authoring canvas toolbar

The following describes what the less familiar icons do:

- **Locate in Library pane** expands the Library to show the location of the flow or operation that you're working on.
- **Run flow in Studio Debugger** opens the Studio Debugger and starts a run of the current flow in it.
- **Toggle labels** shows or hides labels of objects such as responses. An Ops flow appears in PAS Central with labels showing or hidden according to whether this button had toggled them on or off when you last saved the flow.
- **Show responses** toggles to represent each of a response's inputs with an icon like those of the flow return steps.
- **Align...horizontally** and **Align...vertically** respectively align flow elements that you select.
- If you have overlapping steps as you work on the canvas, the **Send to back** and **Bring to front** icons move the step that has the focus to the back or front of the stack, respectively.
- **Resolved**, **Diagnosed**, **No Action Taken**, and **Error** add those flow return steps respectively to the flow.

Bookmarks pane

The **Bookmarks** pane (expandable from the **Bookmarks** tab in the upper-right of the window) contains operations and flows that you use frequently, making them more readily available.

Icons pane

The **Icons** pane (expandable from the **Bookmarks** tab in the upper-right of the window) contains numerous libraries of operation icons that you can use to make it clear more quickly what a step's operation does. These icons replace the default operation icon on a step that you drag the icon to.

Keyboard shortcuts in Studio

The following sections list some of the keyboard shortcuts that are available in Studio:

Library pane keyboard shortcuts

SHIFT + SPACE

Expand everything below selected

ENTER

Open editor on selected

DEL

Delete selected

Authoring pane keyboard shortcuts

DEL

Delete selected step

CTRL + x

Cut

CTRL + c

Copy

CTRL + v

Paste

CTRL + z

Undo

CTRL + y

Redo

CTRL + t

Insert callout

ALT + F6

Zoom in

ALT + SHIFT + F6

Zoom out

Properties Editors / Inspector keyboard shortcuts

CTRL + <right-arrow>
Edit selected row

Scriptlet panel keyboard shortcuts

CTRL + f
Find

Icons pane keyboard shortcuts

DEL
Remove selected bookmarks

Locking a repository

Locking a repository means that only the person who has locked it can make changes or publish to it. Locking a repository is useful for preserving work as it has been tested on a QA (testing) server, when you are ready to publish the flows from the QA server to a production server.

To lock a repository

1. Open a different repository from the one you want to lock.
2. Set the repository you want to lock as the target repository.
For more information on setting the target repository, see [Publishing to and updating from the public repository](#).
3. On the **Repository** menu, click **Lock/Unlock Target Repository**.

Publishing to and updating from the public repository

Whether on a staging server for testing purposes or in a production environment, you exchange flows with the PAS Central repository by publishing to and updating from its repository. The recommended scenario for working with other Ops flow authors to supply flows to your organization's production installation of PAS Central is that you have Central installed on a staging server and both authors working an installation of PAS Studio on his or her own computer. With this arrangement:

- Both authors make their work to each other by publishing to and updating from the staging server's installation of Central.
- When an Ops flow has been sufficiently tested on the staging server, then one of the authors locks the staging server's repository and publishes the flow (or flows) to the repository that servers the production installation of Central.

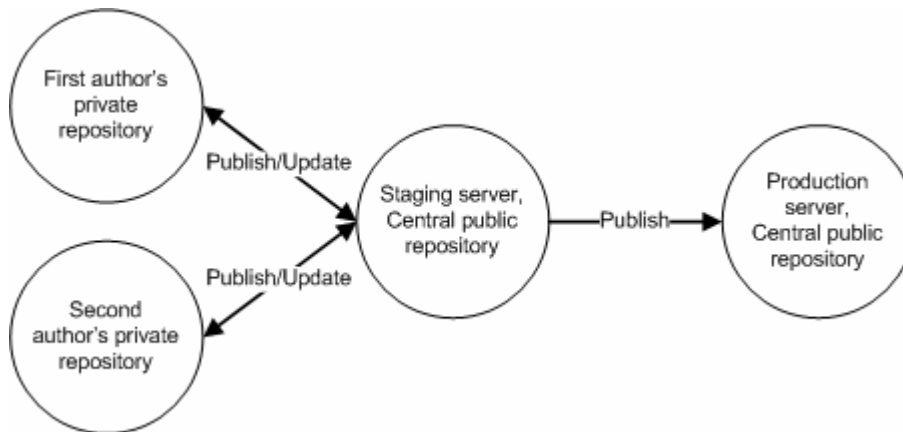


Figure 30 - Recommended arrangement for two authors or more creating Ops flows

Updating your repository from the public repository becomes important when there are multiple authors publishing to a single PAS Central repository. (If there's a single author, the direction of work is one way, from the author to PAS Central. If you are a single author and need to acquire an Ops flow or Accelerator Pack that that exists elsewhere, you can import the flow or Accelerator Pack.)

When you publish or update, you can pick and choose which groups of objects you publish, with the constraint that when an Ops flow is published, all of its dependent objects – that is, the operations and system objects that that Ops flow uses – are published or updated with the Ops flow.

For information on publishing and updating flows, operations, and system objects, see [Moving flows and their elements among repositories](#).

Creating operations from Web services

The Web Services Wizard creates PAS operations based on the API in the Web Service Definition Language (WSDL) of the Web service that you identify in the wizard. (A Web service is a piece of business logic that resides somewhere on the Internet.)

The Web Services Wizard is installed when you install PAS Central. When you run the wizard, you provide it with the WSDL for a given Web service. The WSDL string you provide as a pointer can be a file's location and name or a URL.

For instance, suppose you have an application called AlertAlert that creates a ticket through a Web service and API, and you want to tell AlertAlert to create a ticket. The Web Services Wizard extracts, from the Web service's WSDL, the application's APIs for the actions that can be performed with the application, such as creating or changing a ticket. The WSDL defines the Web service's methods, the inputs that each method needs, and the required format for each input.

When you provide the wizard with the WSDL (in our example, for AlertAlert) and run the wizard, it generates operations that can run against the Web service. The operations appear in the Studio Library, with the required inputs created. You can use the operations in flows. To run a flow that uses one of these operations, in Studio you'll need to create a remote action service (RAS) that points to the Web service.

To create operations for a Web service with the Web Services Wizard

1. In the PAS home directory, in \Studio\tools\ double-click **Wswizard.exe**. The WebService Wizard starts.

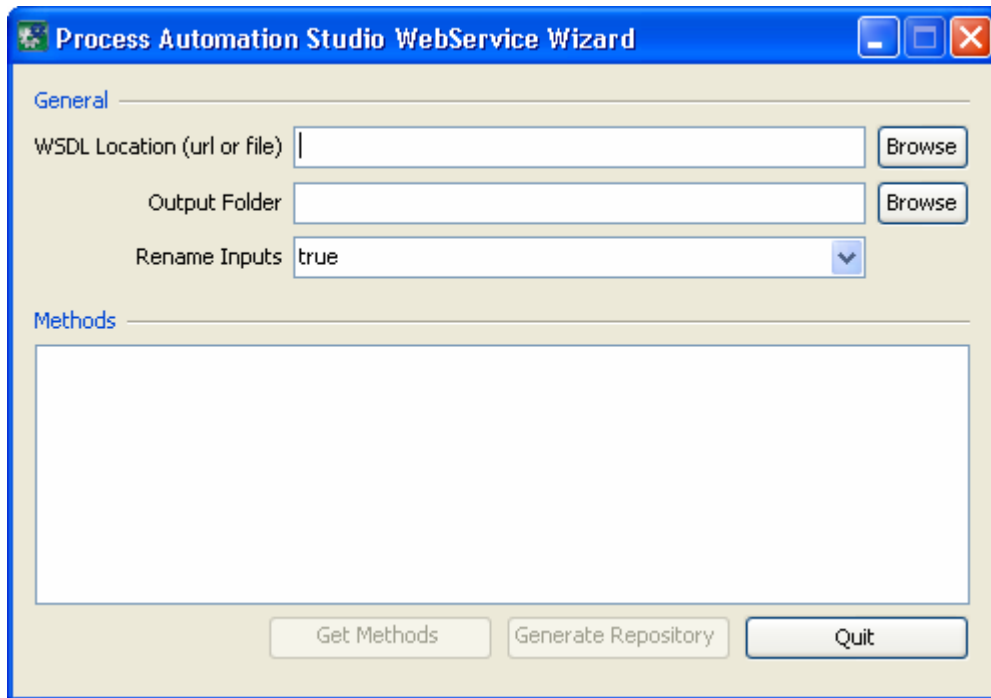


Figure 31 - WebService Wizard

2. In the **WSDL Location** box, type the location of the WSDL.
OR
Click **Browse** and navigate to the location where you want to store the files.
The WSDL location can be a RAS reference that you created in PAS Studio. The RAS is a Web service.
3. In the **Output Folder** box, type a path and filename for the operations to be saved as.
OR
Click **Browse** and navigate to the location where you want to store the files.
You can store them anywhere you like; you will import these XML files into a repository (probably your local private repository).
4. In the **Rename Inputs** box, select either **true** or **false**, depending on whether you want the wizard to rename inputs to a more user-friendly form.
You might want to select **true**, because each input is named for the entire branch of XML tags in the WSDL, such as authenticationInfo|credentials|userName. If you allow the wizard to rename this input, it will shorten the name to something like userName, to make the input easier to use and reference. Then, if you want to open the operation that is generated and rename this input, you can. Note: If you do rename the input, you must change the mapping to reflect the new name. You do so in
5. Below the **Methods** box, click **Get Methods**.
The operations created from the WSDL's methods are listed in the **Methods** box.

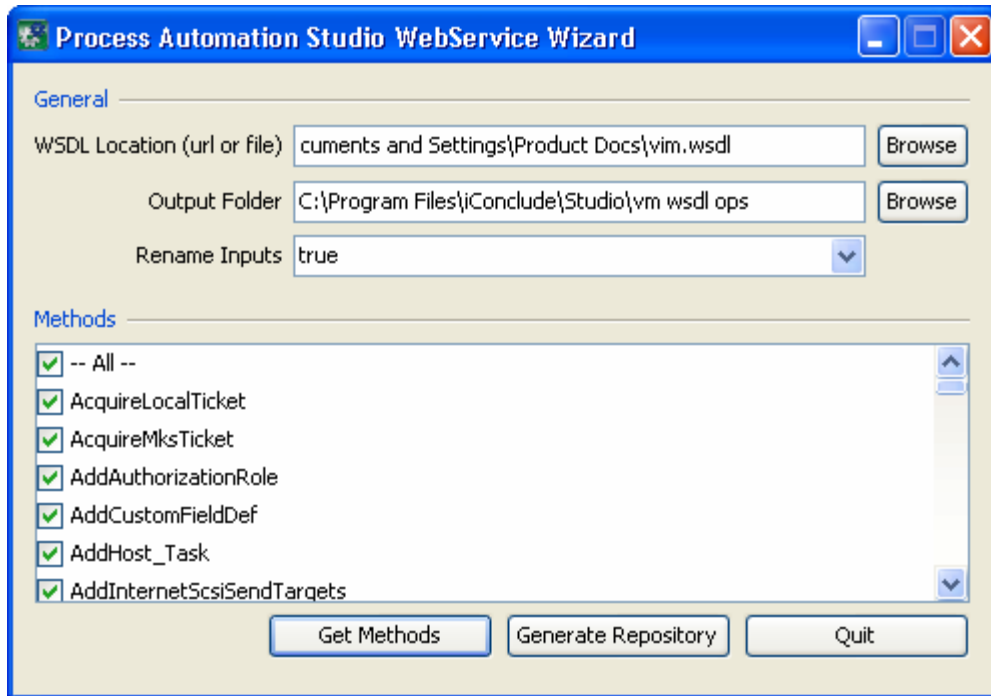


Figure 32 - Selecting methods to get

6. In the list of methods, select those from which you want to create operations, then click **Generate Repository**.
7. Open Studio and import the repository that you just created.

Copying Ops flows and operations

To make changes to an Ops flow or operation that is in the sealed part of the Library, you must make a copy of the flow or operation, then make your changes to the copy.

Similarly, if you want to change an operation from which you have already created a step that you are using in an Ops flow, you should copy the operation and then make changes to the copy and use the copy to create any new steps.

Here's why: Suppose that the operation you want to change has an input, InputA. And suppose that the step that you have already made from the operation is used in a flow, FlowAlpha, that has a flow input that supplies InputA with a value. Now suppose you add InputB to the operation. The step that you already created from the operation doesn't have any way of obtaining a value for InputB. When a way of getting a value is not defined for an input, the input presents a prompt to the user to get its value. However, if an PAS Central user has created a schedule for automatic runs of FlowAlpha, FlowAlpha will break during those runs, because a requirement of running fully automatically is that a run not require any user inputs.

To make a copy of an Ops flow or operation

1. In the **Library**, navigate to and right-click the Ops flow or operation.
2. In the context menu that appears:
 - To create a copy in the same directory, click **Duplicate**.
 - To place the copy in memory, click **Copy**.

- If you clicked Copy, navigate to the location where you want to place the copy and then click CTL+V.

You can rename the copy.

Remote action services

A remote action service (RAS) is an instance of either the Java RAS (JRAS) or .NET RAS (NRAS) that enables an Ops flow to carry out commands outside PAS or on a remote computer, or to integrate with other application programming interfaces (APIs). The JRAS or NRAS must be installed on the computer where it extends an Ops flow's operations, and the flow must have access to an RAS on the PAS Central server that points to the installed JRAS or NRAS. The RAS on the PAS Central server by must have a name and a URL that accesses either the JRAS or NRAS. So you may need to configure a RAS for your flow.

For example:

- To ping a remote computer, you would need a RAS to access the target computer.
- A flow that integrates with the Exchange API would need a RAS that accesses NRAS to complete the integration.

When you installed PAS Central and Studio, the JRAS_Operator_Path and NRAS_Operator_Path RASs were created on Central. If your Ops flow can use either of these RASs, you only need to confirm the availability of the service. However, if your flow uses a RAS that has a different path from the two existing RASs, you would need to add and configure a new RAS service.

Checking the availability of an RAS

To check the availability of an existing RAS

- In the **Library**, right-click the **Configuration\Remote Action Services** folder and click **Open**.
- In the **Remote Action Services** sheet, highlight the RAS whose availability you want to check, and then click **Check Availability**.

Name	Description	URL	Availability
JRAS_Operator_Path		https://red-JTHOMAS-001.opsware.com:9004/JRAS/services/RCAgentService	AVAILABLE
NRAS_Operator_Path		https://red-JTHOMAS-001.opsware.com:9005/NRAS/services/RCAgentService...	AVAILABLE

Figure 33 - Remote Action Services sheet

- If the RAS is not available, click the right-pointing arrow at the end of the RAS's line, and then correct the location of the RAS in the **URL** box.

Note: If you installed NRAS to run on IIS and the service that you just configured is not available, make sure that the IIS default Web site is running on the machine.

Adding an existing RAS

When you open the **Configuration\Remote Action Services** folder, any RASs that are installed somewhere on your network are listed under **Discovered RAS**. Use the

following procedure to make them available to operations that require them for operations outside PAS.

To add an existing RAS

1. Open the **Configuration\Remote Action Services** sheet.

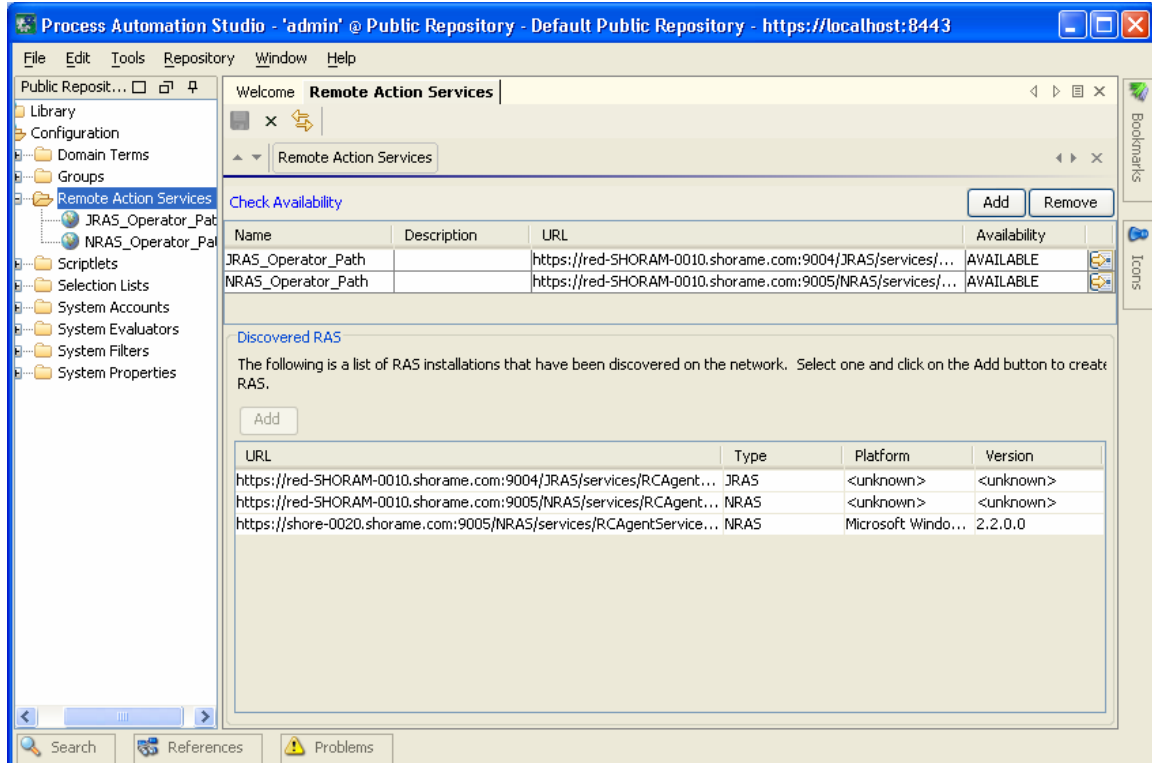


Figure 34 - Discovering RASs on the network

2. Under **Discovered RAS**, select the RAS you want to make available to your operations, and then click the **Add** button (the one under **Discovered RAS**).
3. In the dialog that appears, name the new RAS.
The new RAS appears in the list of RASs above.

Adding or configuring a new RAS

To add a new RAS

1. If you want only to create a new RAS, right-click the **Remote Action Services** folder and then click **New**.

The **Remote Action Services** sheet opens in the authoring pane and a dialog appears in which you name the new RAS.

OR

If the **Remote Action Services** sheet is already open, click the upper **Add** button.

2. In the box that appears, type a name for the new RAS and then click **OK**.
3. In the **URL** column, type a URL with one of the following syntaxes:
 - When configuring a JRAS:
`http://<yourHostname>:<yourPort>/JRAS/services/RCAgentService`

If you accepted the default port number in the JRAS installation program, the port number is 4085.

- When configuring an NRAS:

<http://<yourHostname>:<yourPort>/NRAS/services/RCAgentService.aspx>

If you installed NRAS to run on IIS, the default port number is the IIS port number, 80.

If you installed NRAS to run standalone and accepted the default port number in the NRAS installation program, the port number is 4080.

4. Click **Check Availability**.

The availability or lack thereof appears in the **Availability** column.

If you installed NRAS to run on IIS and the service that you just configured is not available, make sure that the IIS default Web site is running on the machine.

5. When the service is available, click **OK** at the bottom of the pane, and then save your changes.

To configure a RAS

1. In the **Library**, open the **Configuration\Remote Action Services** folder and either double-click the RAS you want to configure or right-click the RAS and then click **Open**.

The **Remote Action Services** sheet opens in the authoring pane.

2. In the **URL** column, type a URL with one of the following syntaxes:

- When configuring a JRAS:

<http://<yourHostname>:<yourPort>/JRAS/services/RCAgentService>

If you accepted the default port number in the JRAS installation program, the port number is 4085.

- When configuring an NRAS:

<http://<yourHostname>:<yourPort>/NRAS/services/RCAgentService.aspx>

If you installed NRAS to run on IIS, the default port number is the IIS port number, 80.

If you installed NRAS to run standalone and accepted the default port number in the NRAS installation program, the port number is 4080.

3. Click **Check Availability**.

The availability or lack thereof appears in the **Availability** column.

If you installed NRAS to run on IIS and the service that you just configured is not available, make sure that the IIS default Web site is running on the machine.

Domain terms for Dashboard charting analysis

Domain terms are attributes that you can assign to flows and inputs, usually to be associated with specific data that the flows obtain, the kind of IT component and the specific IT component that the flow ran against, the type of trouble ticket that prompted the running of the flow, the action that the flow took, etc. Once associated

with such specific data, domain terms add to the capability of PAS Central as a diagnostic tool for your IT infrastructure.

Suppose that you run a flow that takes an IP address as that of the server that it will run against and you assign the domain term Configuration Item to that input. After the flow has run in Central, the Central user could view charts that report other factors against Configuration Items, such as the **Flows per Configuration Item for the Last 7 Days chart**. If the flow ran against 192.118.55.109, there would be a row in the chart labeled "192.118.55.109," with bars that break down the flows that ran against that server.

In PAS Central, on the **Dashboard** tab, users can view charts that collate different combinations of domain terms and the values of the inputs associated with them to bring to the surface analyses that reveal trends in your IT infrastructure, such as:

- Overall usage and results of Ops flows
- What kinds of and how many alerts or events have occurred
- Configuration items (servers, applications, etc.) that have been affected.

The domain terms that you can use for reporting data to Dashboard charts include the following. Some of them have values by default; others obtain their values from the flow's inputs; yet others have the values that you create for them.

- **Action**
The action that the flow performed. You add these values to the list, then enter them as values of inputs that are associated with this term.
- **Alert**
The identity of the alert that you want to associate with the flow. The value associated with Incident can be supplied by a flow input.
- **Category**
The category of the flow. A number of categories are supplied by default; you can add to the list of categories.
- **CI Minor Type**
These terms could be the divisions of CI Types (see next domain term). For instance, if "network hardware" is a CI Type term, then CI Minor Type terms might include "router," "server," "gateway," etc. You add these values to the list, then enter them as values of inputs that are associated with this term.
- **CI Type**
These terms could be the major divisions of IT infrastructure elements, such as "network hardware." You add these values to the list, then enter them as values of inputs that are associated with this term.
- **Configuration Item**
Specific servers, routers, switchers, applications or other elements in your IT establishment. The value associated with Configuration Item can be supplied by a flow input.
- **Incident**
The incident that triggered the running of the flow. The value associated with Incident can be supplied by a flow input.
- **Problem**

The issue that the running of the flow is meant to correct. You add these values to the list, then enter as values of inputs that are associated with this term.

- **Severity**
The severity of the problem, incident, or alert that triggered the flow.

Adding domain terms and domain term categories

To add a domain-term category

1. In the **Library** pane, expand the **Configuration** folder, right-click the **Domain Terms** folder, and click **New**.
2. Name the new domain-term category and save your work.

To add a domain term

1. In the **Library** pane, expand the **Configuration** and **Domain Terms** folders, and double-click the domain term category that you want to add the term to.
2. In the domain term editor that appears in the authoring pane, click **Add**, then type the term and its description in the **Name** and **Description** columns, respectively.

Creating Ops flows

The main steps of creating an Ops flow, once you have created or imported the operations that you will use in it, are:

- Creating the Ops flow, as described in this section.
- Adding operations as steps.
- Creating connections (known as “transitions”) between steps so each response for a step takes the flow to another step.
- Creating one or more return steps to end the Ops flow and assigning each return step an Ops flow response.
- Saving the Ops flow.
- Debugging the flow.

For information on debugging an Ops flow in Studio, see [Debugging a flow in Studio](#).

Tip: To help with understanding the steps, open the Restart Service – Tutorial Flow.

To open the Restart Service - Tutorial Ops flow’s Properties sheet and authoring canvas

1. Find the Restart Service – Tutorial Flow in the Library.
2. To open the Restart Service – Tutorial Flow **Properties** sheet, right-click the flow name and then click **Properties**.
3. To open the Restart Service – Tutorial Flow canvas in the authoring pane, double-click the flow name.


To create an Ops flow

1. Highlight the folder in which you want to create the Ops flow.
2. From the **File** menu, point to **New** and then click **Flow**.
3. Name the Ops flow, using standard characters.

- In PAS Studio, you cannot give two Ops flows the same name.
- Naming in Studio is not case-sensitive.
- Names can be a maximum of 128 characters long.

A new Ops flow diagram appears in the authoring pane.

You can also create a flow from one of several templates that are designed specifically for completely some frequently undertaken tasks. These templates give you a head start by providing the steps needed to complete those common tasks. For information on creating a flow from a template, see [Creating a new Ops flow](#).

In the Library, the new Ops flow is marked with the Warning symbol () and its name appears in red as long as it is incomplete. Moving the cursor over the name of an incomplete Ops flow displays a tool tip that specifies how the flow is incomplete.

Creating an Ops flow step

In addition to creating a step by dragging an operation or flow onto an Ops flow's authoring canvas, you can create a step by copying an existing one.

To create a step

- With the Ops flow diagram open in the authoring pane, locate the desired operation or flow in the Library pane and drag it to the diagram.

OR

If the operation that you want to use to make a step is in the **Operations** folder and you plan to make changes to it, make a copy of the operation (select the operation, click Ctrl+C, then Ctrl+V in the folder where you want it to reside), and then drag the copy to the authoring canvas. Operations in the **Operations** folder are sealed and so cannot be modified.

Note: When you create a step in an Ops flow by dragging a flow from the Library to the current flow canvas, the flow that you drag becomes a subflow of the flow into which you drag it.

The first step that you drag to the flow's canvas automatically becomes the Start Step of the flow. The Start Step is the step that uses the data provided by the flow's inputs. The current Start Step always has a green outline.

Important: The step that you create should not be confused with the operation or subflow that the step is associated with. Steps are particular instances of operations or flows. If you make changes to make the effect of the operation specific to your flow's current needs, you should make these changes on the step.

For instance, to specify a host machine for the operation to run against, you should specify the host in an input to the step rather than to the operation. If you were to specify a particular machine in one of the operation's inputs, the operation would break any flow that the operation was part of and that could not properly run against that machine.

To change which step is the Start Step

- Right-click the step that you want to start the flow and, from the context menu that appears, select **Set As Start Step**.

Prompting the user before running the step

Even if the step does not require user input, you can create a prompt that requires the user's consent before running the step.

Tip: For any step that requires information of the Central user, you can supplement the Ops flow description and help the user by creating a prompt that tells what he or she will need to know for this step.

To create a user prompt for the step

1. Open the step's **Properties** sheet, and then click the **Display** tab.
2. To prompt the user, select the **Always prompt user before executing this step** checkbox.
3. Create the prompt in the following boxes:
 - Create a label for the prompt in the **Prompt Title** box.
 - Specify the size of the prompt in pixels in the **Prompt Width** and **Height** boxes.
 - Provide a message to the user in the **Prompt Text** box.
4. Click **OK**, and then save your changes.

Evaluating data for correct format

Evaluators are string formats used to validate inputs for any data sources except system accounts. They can be particularly useful for validating data obtained from the following data assignments for inputs:

- User prompt
- Specific value
 - If the specific value contains a flow variable (such as `${domain}`), the evaluator is applied to the specific value with the variable's value (rather than the variable reference) included in the specific value.
- Previous step's result

Studio has by default system evaluators for validating the following:

- Alphanumeric
- Email
- Filename
- IP address
- No white space
- Numeric
- Phone number

Evaluators can use standard evaluators such as `=`, `!=`, `Begins with`, `Contains`, `Match All Words`, `Match At Least One Word`, etc., or they can use evaluation tools such as the following:


- Regular expressions

For information on creating regular expressions, see [Working with regular expressions](#).

- Scriptlets
- References to shared evaluators

Creating an evaluator

To create an input-validation evaluator

1. In the Library, right-click the **Configuration\Evaluators** folder and click **New**.
OR
Click **Open**, and in the **System Evaluators** sheet that appears, click **Add**.
2. In the **Select Evaluator** drop-down list that appears, select an evaluator or evaluation tool and click **OK**.
3. In the next box, name the evaluator and click **OK**.
The new evaluator appears in the list of evaluators in the **System Evaluators** sheet.
4. To complete the evaluator, click the right-pointing arrow () at the right end of the row.
The system evaluator editor opens. Its appearance depends on the kind of evaluator you
5. In the **Compare To** box, type the string that you want to use the evaluator to test the input.
6. To clear the contents of the **Test Filter Input** box, click **Clear**.
7. Test the evaluator by either:
 - Typing a sample of the text you want to validate
 - Click **Quick Command**; in the **Input** box that appears, type the command and arguments that obtain output that you want to validate or test; and then click **OK**.
8. Save your work.

For instance, to create an evaluator that validated input for URL format, you could select the evaluator **Begins With** and then, in the **Test** text box, type **http://www**. Or you could describe the start of a URL with a Regular Expression.

Editing an evaluator

To modify an evaluator

1. In the Library, open the **Configuration\Evaluators** folder.
2. Right-click the evaluator you want to modify and click **Open**.
3. Make changes as you did in the procedure for creating an evaluator.
4. Save your work.

To edit another evaluator without closing the editor

- Click **Previous Item** or **Next Item**.

Deleting an evaluator

To delete an evaluator

1. Open the **System Evaluators** sheet.

- In the list of evaluators, select the one you want to delete, and then click **Remove**.
- Save your work.

Inputs and flow variables: Providing the flow with data

Step inputs are created from the inputs of the operation from which the step was created. The step input value determines the value of the operation input.

If a flow input's value is assigned to a flow variable and the step input is configured to get its value from that flow variable, then the flow input determines step input. Otherwise, the step input gets its value from the assignment (user prompt, assignment from another flow variable, etc.) that is defined for it.

Note: If you don't supply a flow or a step with an input that it needs, the input is by default converted to a user prompt. This requires, of course, that the user have the information necessary for the flow to do its work.

For information on creating an input with a fixed, specific value or a specific value provided by a flow variable, see [Inputs: Providing data to operations](#).

To add an input to a flow, step, or operation

- In the **Repository** pane, in the **Library** folder, navigate to and double-click the flow name and then click the **Properties** tab at the bottom of the authoring pane.

OR

To add an input to a step, with the flow diagram open, click the step and then, at the bottom of the diagram, click **Inspector**. The step's Inspector opens.

OR

To add an input to an operation, in the **Library** folder, navigate to and double-click the operation.

- On the **Inputs** tab, click **Add Input**.
- In the text box that appears, type the name of the input and then click **OK**.

The new input appears as a row on the **Inputs** tab.

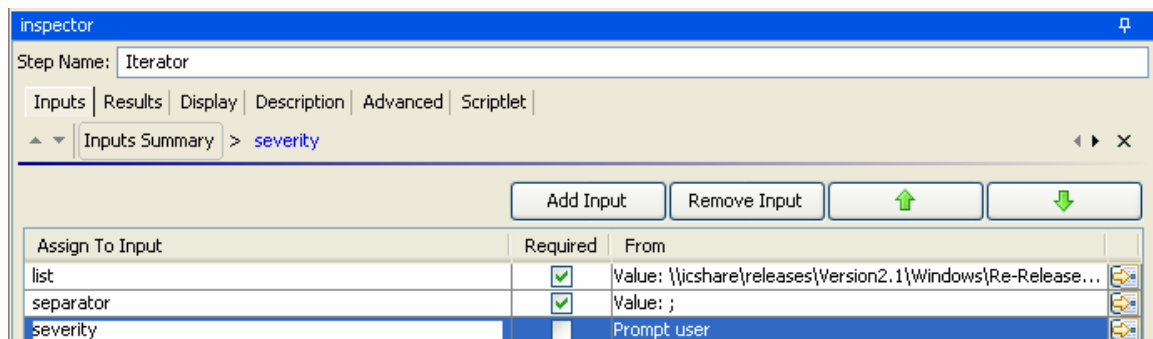



Figure 35 - Inputs tab with new input

- To set the input as a requirement for the flow to run, select the **Required** checkbox.

- To specify how the input gets its value, click the right-pointing arrow (). The tab changes to the input editor

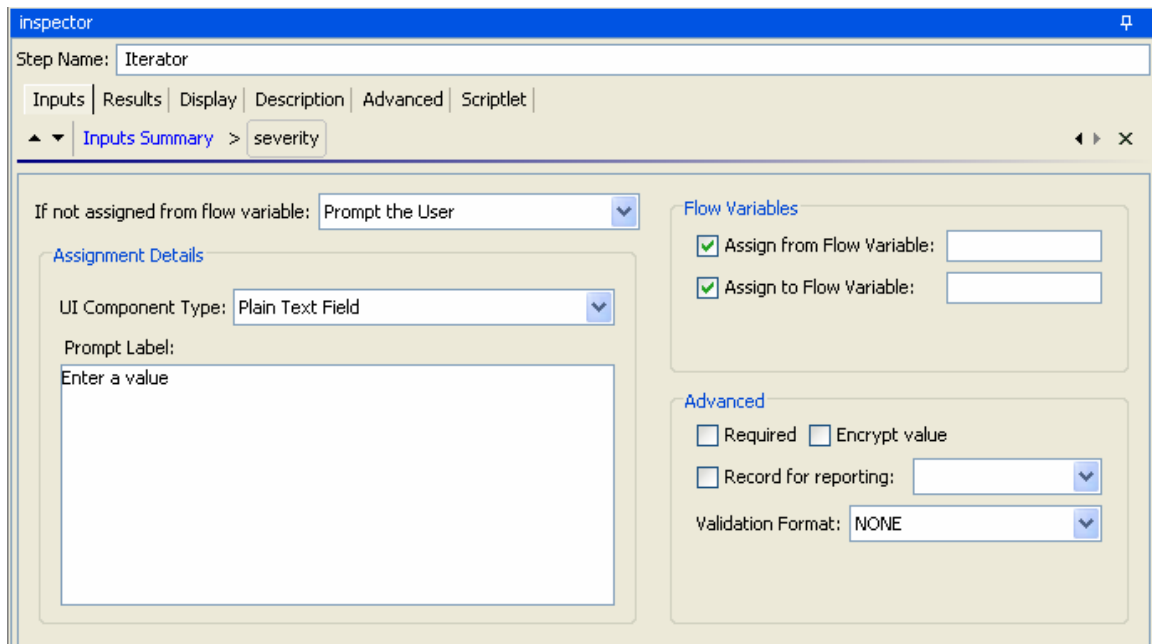


Figure 36 – Assigning a data source to the input

- To specify that the input get its value from a flow variable, make sure that the check box beside **1 (If checked, first attempt to assign the input from a flow variable)** is checked, and type the flow variable's name in the text box.

Note: Among the flow variables whose values you can assign to the input are reserved flow variables, whose values are always available for referencing anywhere in the flow. For more information, see [Flow variables](#).

Creating an input always creates a flow variable of the same name.

- To provide the input's value from a flow variable, under **Flow Variables**, select the **Assign from Flow Variable** check box.

If you select the **Assign from Flow Variable** check box and leave the text box blank, then the input's value is assigned from the flow variable of the input's name.

- To get the input's value from a different source, under **Assignment Details**, select one of the following sources from the **If not assigned from flow variable** drop-down list:

- **Prompt the User**

A value that the user supplies to the flow in response to a prompt.

To complete this assignment of data to the input, see the following procedure.

- **A Specific Value**

An unchanging value that the Ops flow always uses for its first step to act on. You determine what this unchanging value will be.

In the **Assignment Details** area, in the **Value** box, type the text of the specific value.

- **The Previous Step's Result**

This is the result of the previous step as modified by any filters you have created on that step.

- **A System Account**

This type of input source enables the flow to perform tasks that require account credentials, while protecting the credentials from exposure by keeping them hidden behind the system account name. The system accounts that are available in the list are those that you have created. For information on creating system accounts, see [System accounts: secure credentials](#).

In the **Assignment Details** area, in the **System Account** drop-down list, select the name of the system account that you want to use for the input.

- **The Logged-in User's Credentials**

This data source provides credentials when the log-in account on which the flow is being run has the required access permissions to perform the tasks required by the flow.

9. To store the input value in a flow variable, under **Flow Variables**, select the **Assign to Flow Variable** check box and type a name in the text box
10. To assign the input value to a flow variable with a different name from the input, type the name of the flow variable in the text box.

OR

To assign the input's value to a flow variable with the same name as the input, leave the text box blank.

11. To make the input required, select the **Required** check box.
12. To encrypt the value, select the **Encrypt value** check box.
13. To record the input's value for charting in the PAS Central dashboard, select the Record for reporting check box and then, from the drop-down list beside the check box, select the domain term (Action, Alert, etc.) to record the value as.
 - For information on domain terms used for recording input information for PAS Central for Dashboard reporting, see [Recording values for reporting in Dashboard charts](#).
 - For information on domain terms used in Dashboard reporting and on creating new domain terms and domain-term categories, see [Domain terms for Dashboard charting analysis](#).
14. Save your work.

To complete assignment of the "Prompt the User" type data source

1. In the **UI Component Type** drop-down list, select the type of user interface to present the user:

- **Plain Text Field**
- **Selection List**

In the **List Name** drop-down list, choose a selection list. These lists are stored in the **Library** pane, in the **Configuration\Selection Lists** folder.

Note: You can add more selection lists to the **Configuration\Selection Lists** folder or add items to existing lists. For information on how to do so, see [Creating and adding to selection lists](#).

- **Selection List From Flow Variable** – a selection list that is provided by a flow variable.

In the **Variable Name** text box, type the name of the flow variable.

Note: In the flow variable, the selection list elements are separated by line breaks.

- **Domain Term**

From the Domain Term drop-down list, select a domain term from which to get the value.

- **User-entered Credentials**

These are a name and password that that the user supplies to the flow in response to the prompt.

2. In the **Prompt Label** box, type a label to show the user.
3. Save your work.

To remove an input from a step

1. Open the Inspector for the step you're interested in.
2. On the **Inputs** tab, on the list of inputs, highlight the input that you want to remove, and then click **Remove Input**.

Flow variables

To pass a value to an input from somewhere else in the flow, you assign the value to a flow variable, then specify that this input get its value from that variable. Similarly, to make an input's value available elsewhere, such as to other steps or to transitions for displaying to the flow user, you assign this input's value to a flow variable, which another flow element can then reference.

In addition to flow variables that you create, the following reserved flow variables are always available for use in any flow or run. These flow variables can be particularly valuable for using within a scriptlet.

Reserved flow variables

Reserved flow variables	Possible values, initial values	Notes
Yes-No	Yes No	
ScriptLanguages	vbscript jscript	
WindowsScriptLanguages	VBScript Jscript	
SiteScopeAckType	Enable Disable	
RemedyCaseStatuses	New Assigned Work In Progress Pending Resolved Closed	
IISSiteStatus	Running Stopped	
VMWareActions	Start Stop Enumerate Status	
WindowsServiceStatus	Running Stopped Paused	
ServiceStatus	Running Stopped Paused	

run_id	Initial value: -1	
SqlAuthentication	Windows Sql	
execution_userid	The value is the user name of the user who is logged in and running the flow.	
WMIQueryFormat	csv text	
SQLReportingServerFormat	Boot System Auto Manual Disabled	
Boolean	True False	
WindowsClusteringStates	Online Offline	
MailBodyType	html text	
SQLDBType	Initial value: Oracle	
RemedyUrgency	High Medium Low Urgent	
FailureMessage	Initial value: null	
TimedOut	Initial value: null	
Result=Yes-No	Yes No	
RS_Previous_Response		The response of the previous step
RS_Previous_Translation		The transition from the previous step
RS_Previous_Translation_Annotation		The annotation of the last transition
execution_userid		The user id of the person running the flow
run_id		The id of the run

Recording values for reporting in Dashboard charts

Recording data from a flow for use in PAS Central Dashboard charts takes place in the inputs for a single step in the flow. On the step, you create inputs that get their data from the flow variables where you stored the data you want to record. Because all the values must be recorded on one step, that step must follow the last step in which one of the values you want to report is generated. This is often the Success return step.

You specify a domain term to record the input's value as. An PAS Central Dashboard bar chart then has the information it needs to record the input's value in this flow. The domain term is either the vertical or the horizontal axis of the bar chart, and the input value is charted on that axis. The chart might depict the names of flows on the horizontal axis and the servers that the flows were run against on the vertical axis.

The data that you want to correlate for reporting purposes must all be input values on a single step. Therefore, the step must follow the last of the steps that obtained the data that you want to record for reporting and have correlated with each other. This means that return steps and steps that contain remedies for a problem are frequently the steps in which all the necessary inputs for Dashboard reporting are both defined and recorded. You can create inputs on these steps that don't do anything except record values.

For example, suppose that an IT manager wants to see, in PAS Central, a Dashboard chart that shows which servers the Restart Windows Server was run against and what the actions were. For either the Ping or the TraceRoute step, you could record the host input as a domain term that one of the Dashboard charts in PAS Central uses. One such domain term would be "Configuration Item," which obtains the names or IP addresses of applications, and hardware components, including servers. The IT manager could choose the **Alerts Per Configuration Item** chart, which would show the IP addresses or domain names of the servers that the flow was run against. If the manager adds the **Flows Per Configuration Item** chart to the Dashboard, he or she can see instantly which flows have been run against those servers. Further, by double-clicking a bar representing Restart Windows Server flow when it ran against a certain server, he can see the results of the flow.

There are many other possible uses for which you can use reporting. For instance, if you have a step that takes an action on the target server such as restarting the server, you could record the appropriate input as an Action. In order to see how many times a given server was restarted, in PAS Central you could then view the **All CI's Organized by Action** chart.

Let's see how we're going to record this information for reporting. We'll record this input with the domain term Configuration Item.

In PAS Central, you can then add a chart that charts that domain term. For information on viewing and creating Dashboard charts, see Help for PAS Central.

To record an input for Dashboard reporting

- With the **Inputs** tab open for the input you want to record, select **Record for reporting** and in the drop-down list select the domain term under which you want it recorded.

The value of the input for this flow will be reported to PAS Central. Any Dashboard chart that has an axis that presents data recorded for the domain term that you chose will show this inputs value on that axis.

Displaying messages to users

You can display, by itself or at the top of an input prompt to the user, a message that provides information to the flow user.

To display a message to the flow user

1. On the **Display** tab of the step's **Properties** sheet, select **Always prompt user before executing this step**.
2. To provide a title for the message, in the **Prompt Title** box, type the title.
3. To specify a width and height for the display, type the width and height, in pixels, of the message or prompt of the appropriate text box.
4. In the **Prompt Text** box, type the text to be displayed.
5. Click **OK**, and then save your changes.

Working with operation results

You can achieve fine control of Ops flows and operations by modifying outputs:

- Specifying the source of the result
For information on specifying the sources of results, see [Adding a result](#).
- Filtering results in order to fine tune the data available for testing or for passing to another operation
For information on creating filters, see Quick View, [Filtering an operation's results](#). For more detailed information on results filters, see [Filter details](#).
- Specifying rules for testing results to determine responses

In addition, you can pass operation results to the flow as fields in the flow result.

Passing results to flow output fields

Flow variables that you create within a flow cannot be referenced outside that flow. However, you can pass values outside the flow to a parent flow (a flow of which this flow is a subflow).

To pass a result to the flow output field

1. On the **Outputs** tab of a step add a result, specifying the following:
 - In the **Assign To** column, select **Flow Output Field** from the drop-down list.
 - In the **From** column, pick the desired operation result.
2. Open the **Properties** sheet of the flow.
3. On the **Outputs** tab, click Add Result and name the new flow result.
4. In the **Result Field** column, pick the step result.

For more on using this technique to pass information from a subflow to its parent flow, see [Using subflows to simplify flow design](#).

Filter details

The following are the kinds of filters you can create and how you specify filter specifics of each kind. For information on creating filters for results, see Quick View, [Filtering an operation's results](#).

You can create the following types of filters. Depending on the type of filter that you create, specify the filter particulars in the **Field Filters** dialog box, under **Details For** as described below:

- **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 unchecked, the filter changes all the characters to lower case.
- **Format**
A Format filter attaches text that you type in the **Text** box to the result or replaces the results with the text. In the **Place Input At** list, select **Beginning** or **End** to prepend or append the text or select **Replace** to replace the output with the text that you typed.

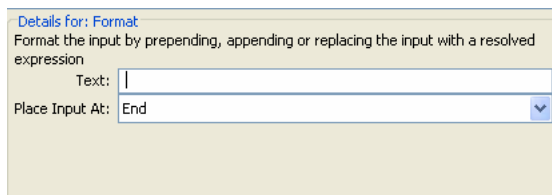


Figure 37 – Format filter definition

- **Line Count**
A Line Count filter outputs the total number of lines of the result. There are no settings to edit for this filter.
- **Regular Expression**
Filters the raw results using a regular expression (regex). For more information on regular expressions, see [Working with regular expressions](#).

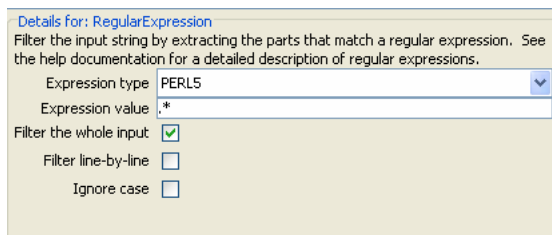


Figure 38 - Regular Expression filter definition

- In **Expression type**, select the type of regex to apply.
- In **Expression value**, type the regex.
- Select **Filter the whole input** or **Filter line-by-line**, according to how you want the filter applied to the raw results.
- To make the regex not case-sensitive, select **Ignore case**.
- **Replace All**
Replaces all instances of one string with another string.

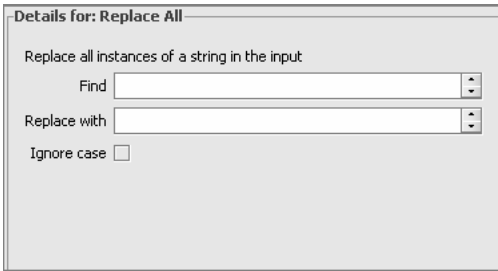


Figure 39 - Replace All filter definition

In the **Find** and **Replace with** boxes, type the string to search for and the string to replace that string with, respectively. To make the search not case-sensitive, select the **Ignore case** checkbox.

- Scriptlet
Filters a result with a scriptlet that you create

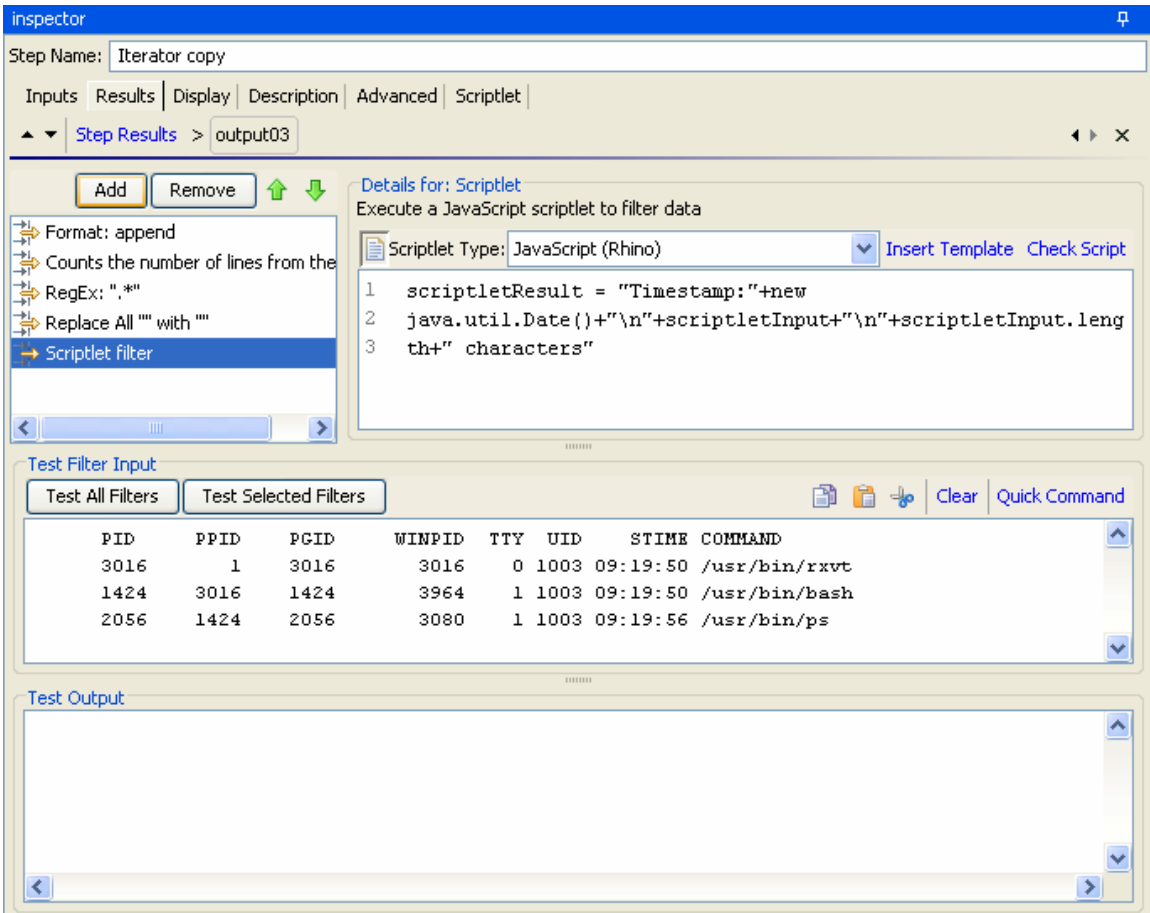


Figure 40 - Scriptlet filter definition

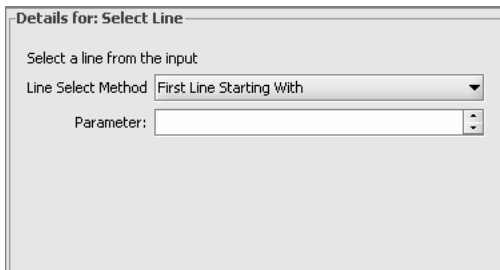
Choose either the JavaScript (Rhino) or the Sleep scriptlet language. If you choose JavaScript (Rhino) as the type of script, the text box starts you out with lines that you will need for the scriptlet to work as a filter.

- To debug the script, click **Check Script**.
- To start your script using a template that includes the most commonly used commands for accessing flow variables (whose values are *context data*),

operation results, and inputs and setting and manipulating flow variable values and results, click **Insert Template**.

- **Select Line**

Defines a line that you want to extract from the raw results.



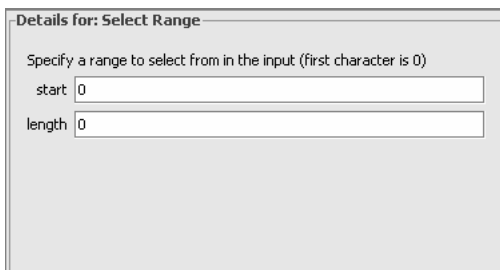
The screenshot shows a dialog box titled "Details for: Select Line". Inside, there is a label "Select a line from the input". Below this is a dropdown menu labeled "Line Select Method" with the selected option "First Line Starting With". Underneath the dropdown is a text box labeled "Parameter:".

Figure 41 - Select Line filter definition

From the **Line Select Method** list, select a criterion for the line that you're interested in and then, in the **Parameter** text box, type a string that the string contains.

- **Select Range**

Defines a string that you want to extract from the raw results.



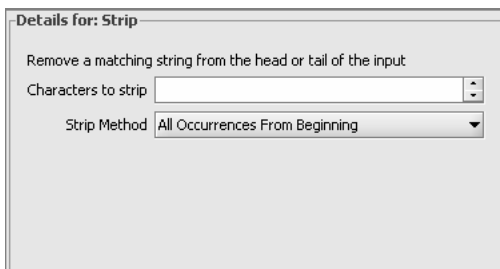
The screenshot shows a dialog box titled "Details for: Select Range". It contains the instruction "Specify a range to select from in the input (first character is 0)". Below this are two text boxes: "start" with the value "0" and "length" with the value "0".

Figure 42 - Select Range filter definition

In the **start** and **length** boxes, type the zero-based start position and the character length of the string that you want extract from the raw results.

- **Strip**

Strips characters from the raw results.



The screenshot shows a dialog box titled "Details for: Strip". It contains the instruction "Remove a matching string from the head or tail of the input". Below this is a text box labeled "Characters to strip" and a dropdown menu labeled "Strip Method" with the selected option "All Occurrences From Beginning".

Figure 43 - Strip filter definition

In the **Characters to strip** text box, type the string to find.

From the **Strip Method** list, select how you want the filter to strip the raw results. You can specify stripping all characters up to or up to and including, or all characters after, or after and including the string that you specified in the **Characters to strip** text box.

- Strip Whitespace
Removes all the whitespace characters from the front and the end of the raw results. There are no settings to specify for this filter.
- 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: Row Delimiter:

First Row is Header: Strip First Row of Result:

Sort On Column: Ascending:

Select Row: Select Col:

Select Width: Select Height:

Figure 44 - Table Sort filter definition

Note: Row numbering is 0-based, and column numbering is 1-based.

- In the **Column Delimiter** list, choose the character that will serve to divide the data into columns in a meaningful way.
- 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 white spaces count as a single white space, so a column may be occupied by data that you expected to find in a column to the right. For an example, apply this filter to the output of a "dir" command-line command with whitespace specified as the column delimiter.

- To treat the members of the first row as column headers, select **First Row is Header**.
- To remove the first row, select **Strip First Row of Result**.
- To sort on a column, type the column number in the **Sort On Column** box. Column numbering is 0-based.
- To sort on a row, type the row number in the **Sort On Row** box.
- To specify ascending order, select the **Ascending** box. By default, the sort order is descending.
- To select a row you want the filter to extract, type the row number in **Select Row** and in **Select Width** type the number of columns in that row that you want extracted. Remember that row numbering starts with 0.
- To select a column you want the filter to extract, type the column number in **Select Col** and in **Select Height** type the number of rows in that column that you want extracted. Remember that column numbering starts with 1.

For example, to extract the first 5 rows of the 2nd through 4th 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**

Connecting steps

You connect any two steps with one or more *transitions*. A transition starts from one of a step's responses and goes to another step. Every response in an Ops flow must have a transition either to another step or to a return step that ends the Ops flow.

The lines in the following diagram represent transitions. The names of a response by default becomes the name of the transition that connects it to a succeeding step.



Figure 45 - Transitions in a flow

In addition to simply connecting steps, transitions are valuable for:

- Controlling access not only to flows but to different parts of a flow.
Gated transitions are transitions that require membership in a certain PAS role for the account that executes the flow. Gated transitions are colored red in both Studio and PAS Central.
- Providing a basis for calculating the value of a flow.
The value that you assign a transition is counted toward the value of a flow as the transition is followed during a run.
- Providing flow users with information on the outcome of a step.
The descriptions that you wrote in the **Properties** sheets for the transitions that a flow run follows appear in the **Results Summary** area of Central as the **Description** for each step, as the step is completed and the transition followed to the next step.

The following screen shot from Central shows the transition descriptions' appearing in the **Results Summary** area.

▼ Results Summary		
Step	Response	Message
Get Stopped Services	✓	Retrieved a list of services which are currently stopped.
Select a Service	✓	Selected Adobe LM Service to restart.
Restart Service	✓	Restarted service Adobe LM Service
Resolved : success	✓	Return step - Restart Service - Tutorial Flow

Figure 46 - PAS Central Results Summary

To add a transition between two steps

1. On the step that you want to connect to the next step, click the **x** that represents one of the responses, and drag to the step that that response should lead to. When you move the mouse over the **x**, the response is labeled with the response. The transition that you create is also labeled with the response from which it originated.
2. In the flow diagram, select the transition (click its name) and click **Inspector**. The transition's Inspector appears. The following example is the Inspector for the **success** transition that connects the **Select a Service** step's **success** response to the next step.

inspector
🔍

Name:

Gated Transition

Check user's groups before proceeding

Required Group:

Flow Transition Value:

Description

Selected \${service} to restart.

Figure 47 - Transition inspector

3. To limit execution of the Ops flow beyond this transition to a specific role or to assign a value to this transition, under **Gated transition**, select **Check user's groups before proceeding**, and then select a group in the list box beside **Required Group**.

In the example above, flow user's must be a member of the LEVEL_ONE group.

4. To count completion of the transition in the Ops flow's value, type a value (in American dollars, with two decimal places) beside **Flow Transition Value**.
5. Type a description in the **Description** box, then click **OK** and save your work.
Note: The description in the above example refers to the flow variable **\${service}**. For information on using flow variables, see [Flow variables: Making data available wherever you need it](#).

Adding checkpoints to and removing them from a flow

A checkpoint in a flow step gathers the state of the flow at that step and saves it to the Central database. The flow's state comprises all the data necessary to completely reconstruct the run in case of a system crash. If you have configured a failover/run recovery cluster for Central and the Central server running a flow fails, other nodes in the cluster resume the flow runs that were taking place on the failed Central server. If a flow has no checkpoints, the run is resumed at the start of the flow. But if there are checkpointed steps in the flow, the run is resumed from the last checkpoint that was reached when the server failed. By default, checkpoints are created for a step when you create the step.

Note: When a user pauses or interrupts a run, the state of the run at the time of the interruption is saved to the database, so checkpointing is not necessary for resuming the run at the point at which the user paused or interrupted it. Checkpointing is for resuming runs that were lost due to a system failure.

Because a checkpoint is set by default for each step that you create, the amount of information in the flow state can become quite large (such as in loops). Frequent writing of such large amounts of data to the database can affect the flow's performance, so you might want to reduce the number of checkpoints in the flow. For instance, you might remove checkpointing from some steps in loops. If you remove a checkpoint from a step, a run of the flow will be resumed from the last checkpoint before the one you removed.

If you remove a checkpoint from a step that is a subflow, none of the steps in the subflow (or the steps in the subflow's steps' subflows) are checkpointed, regardless of whether they have had checkpoints set for them or not.

To create a checkpoint in a step

1. Open the flow diagram in Design view, and open the Inspector for the step to which you want to add the checkpoint.
2. On the **Advanced** tab, select the check box by **This is a checkpoint step**.

To remove a checkpoint from a step

1. Open the flow diagram in Design view, and open the Inspector for the step to which you want to add the checkpoint.
2. On the **Advanced** tab, remove the selection from the check box by **This is a checkpoint step**.

Using categories to add information about flows

Categories are a kind of domain term that you assign to flows rather than to inputs. Assigning a category to a flow adds another factor that Central users can use in creating Dashboard charts. A number of categories are installed with Studio, but you can also create your own categories.

Central users might use categories to create reports that indicate the health of key infrastructure components. For instance, 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.

To assign a category to an Ops flow

1. To open the flow **Properties** sheet, click the **Properties** tab at the bottom of the flow diagram.
2. Click the **Assign Categories** button, then in the following dialog, select one or more categories, and then click **OK**.

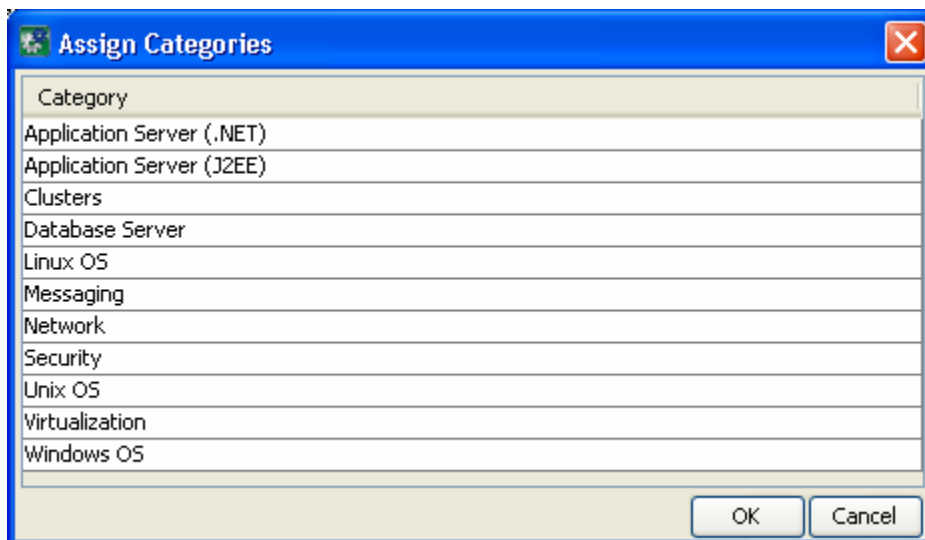


Figure 48 - Assigning a flow to a category

You can create new categories as you can any other domain term. For information on adding new domain terms, see [Domain terms for Dashboard charting analysis](#).

Changing which operation a step is based on

To switch the operation that the step is based on

1. Open the step's Inspector and, on the **Advanced** tab, under **Operation**, click **Select**.
2. In the Select Operation dialog, navigate to the operation you want the step to be based on, select it, and click **OK**.
3. Make any necessary changes to the step's elements, such as input data assignments or the name of the step.

4. Save your work.

Moving steps and transitions in the flow diagram

In addition to dragging steps, you can move steps as a group by using SHIFT + click (then dragging) or CONTROL + click (then dragging).

Copying a step

You can use an existing step as the basis for creating a new step, by copying the step, then changing the step that you have

To copy a step

1. On the canvas, right-click the step and then click **Copy**.
2. Right-click anywhere on the canvas and click **Paste**.

Creating and adding to selection lists

Selection lists are lists of items that you can provide in Ops flow user prompts. There are a broad variety of selection lists provided by default, many of them provided specific for working with various technologies. For instance, 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 ServiceStatus selection list (whose members are Running, Stopped, and Paused). Or to capture the SQL Server authentication type, the input's data-source selection list could be SqlAuthentication (Windows, Sql).

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

To create a selection list

1. In the Library pane, right-click the **Configuration\Selection Lists** folder and click **New**.
2. In the box that appears, name the new selection list.

The selection list editor appears.

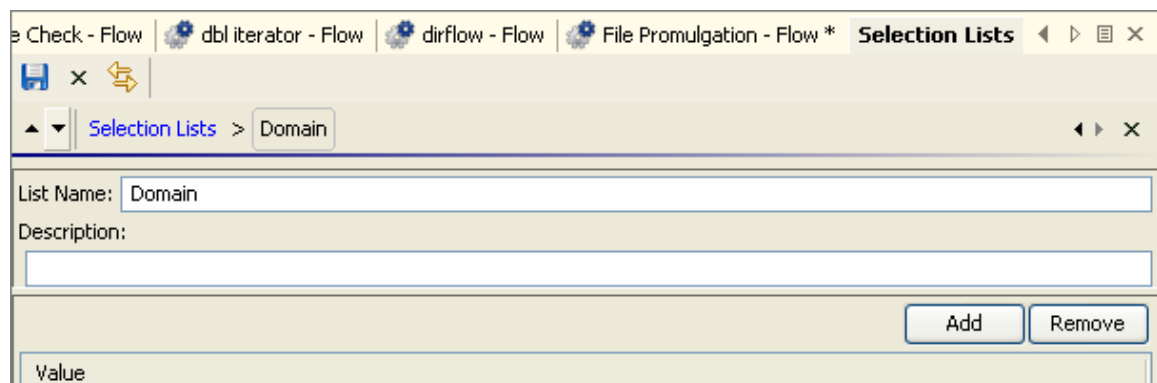


Figure 49 - Selection list editor


3. Give the selection list a description, if you wish.
4. To add an item to the list, click **Add** and name the item.

5. To remove an item, click **Remove**.
6. To close the selection list editor, click the **X** that is on the same level with **Selection Lists**.

To edit a selection list

1. In the **Library** pane, right-click the **Configuration\Selection Lists** folder and click **Open**.

The list of selection lists appears.

2. To edit the list, click the right-pointing arrow () at the right end of the list's row.
3. To add an item to the list, click **Add** and name the item.
4. To remove an item, click **Remove**.
5. To close the selection list editor, click the **X** that is on the same level with **Selection Lists**.

To remove a selection list

1. In the Library pane, click the plus sign to the **Configuration\Selection Lists** folder.
2. Right-click the selection list that you want to remove, then click **Remove**.
3. Click **OK**, then save your work.

System accounts: secure credentials

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.

The flow author provides the system account name to the input when creating the flow; the user never sees the system account name that provides an Ops flow with user account credentials for access to a remote machine. Also, the user cannot enter a system account name in response to a user prompt. Thus the credentials are protected from decryption, and the system account name is hidden from the user.

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

To create a system account

1. In the Library pane, right-click the **Configuration\System Accounts** folder and click **New**.
2. In the box that appears, name the new system account.
The system account editor appears.

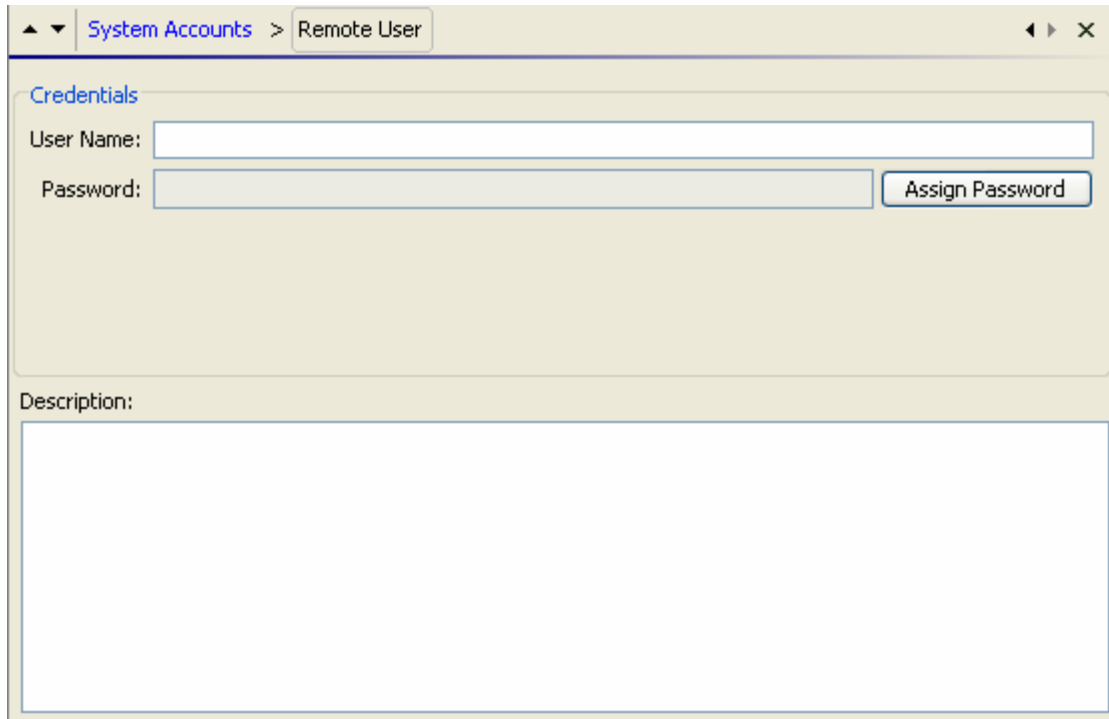


Figure 50 - System account editor

3. Type the user name of the account that the system account represents, using the following syntax:
`<domain>\<username>`
4. Click **Assign Password** and in the box that appears, type the password, then type it again when prompted.
5. Give the system account a description, if you wish.
6. Click the Save icon and then, to close the system account editor, click the **X** that is on the same level with **System Account**.

Editing a system account

To edit a system account

1. In the Library pane, open the **Configuration\System Accounts** folder, highlight the system account that you want to edit, and click **Open**.
2. Make any changes necessary and save your work.

Deleting a system account

To delete a system account

1. In the Library pane, open the **Configuration\System Accounts** folder, right-click the system account that you want to remove and click **Remove**.
2. Save your work.

Controlling access to PAS objects

You can assign permissions that control the kinds of access that groups have to PAS objects such as flows, operations, and system accounts. Following are the permissions and what they are required for:

Read

Required for an author to see an object

Write

Required for an author to delete an object

Execute

Enables an author or Central user to use an object

Link To

Allows an author to use an object in a flow

Following is how the permission editor first appears.

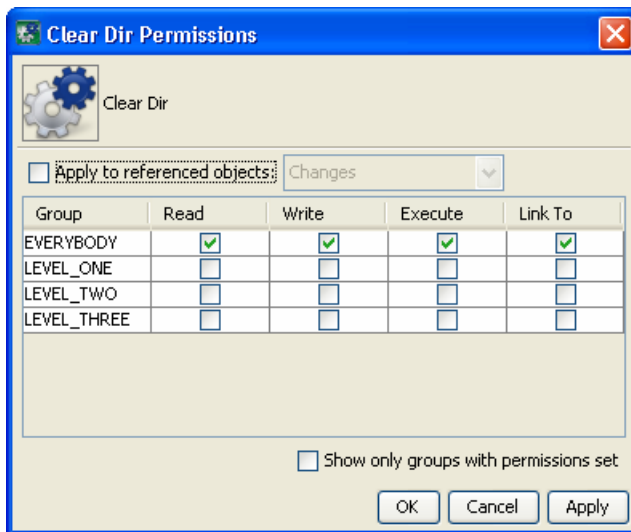


Figure 51 - Setting permissions for the Clear Dir flow

- Changes to permissions appear in black in the relevant check box.
- The **Apply to referenced objects** check box applies permissions to the objects that are referenced by the object for which you're setting permissions. For instance, if you're setting group permissions for a flow, selecting the check box applies the permissions that you specify to the operations and (sub)flows used in that flow, and any objects such as system accounts that the flow or its referenced objects reference.
- When you select the **Apply to referenced objects** check box, you can select either **Changes** or **All** from the check box's drop-down list.
 - Selecting **All** attaches to the object's referenced objects all the permissions that are specified in for the object.
 - Selecting **Changes** applies to the object's referenced objects only changes that you make to the object's permissions.

For example, suppose the Clear Dir flow has permissions that let everyone view the flow but only let members of the LEVEL_ONE group change or run the flow, or reference it in other flows:

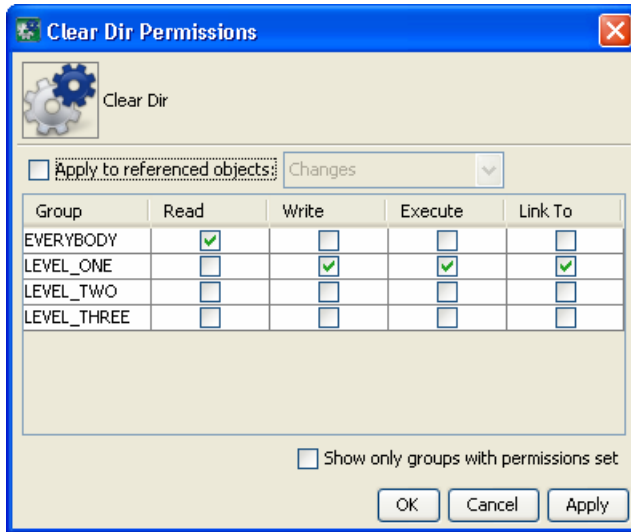
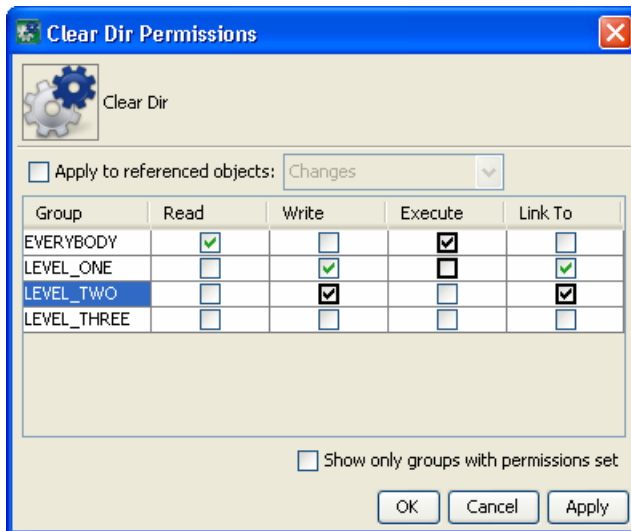


Figure 52 - Group permissions for the Clear Dir flow

Now suppose you want to set permissions the following permissions for the Clear Dir flow:



- EVERYBODY can run the flow
- You allow the LEVEL_TWO group to make changes to the Clear Dir flow and to reference the flow in other PAS objects.

If you were to select the **Apply to referenced objects** check box, then:

- If you select Changes from the drop-down list, only changes to permissions, which appear in black, are applied to the referenced objects, while the referenced objects retain any permission settings that were not removed as changes in this object's permissions editor.

Suppose an operation used in the flow had the default permissions, in which everyone is allowed to do everything. The changes shown above would not affect the permissions for that operation, since you did not select the checkbox for EVERYBODY for any of the permissions.

- If you select **All** from the drop-down list, the current state of all the permissions is applied to all the referenced objects.

This means that a referenced object's default permissions would be changed to those shown above.

To open the permissions grid for PAS objects

1. In the PAS Library, right-click the operation, flow, or system account and, on the right-click menu, click **Permissions**.
2. Select and unselect permissions according to the access that you want different sets of users to have.
3. Click **OK** and save your work.

Saving and re-using filters

You might want to save filters that you create for one operation's result and re-use them in another operation. For instance, the filters in NRAS Ping might be useful for other ping operations. You can save them as system filters, which are stored in the **Configuration\System Filters** folder.

Note that by saving a scriptlet filter as a system filter, you can save a scriptlet specifically for use in a filter.

To save a filter in the System Filters folder

1. Open the appropriate operation and, in the filter editor, select the filter you want to save.
2. In the **Repository** pane, expand the **Configuration** folder.
3. Drag the filter from the operation's filter editor to the **System Filters** folder.
4. To rename the new system filter, right-click it, click **Rename**, and give it a more descriptive name.
5. Save your work.

To use a filter in the System Filters folder

1. Open the editor in which you want to use the system filter.
2. In the Library, open the **Configuration\System Filters** folder.
3. Drag the filter that you want to use from the folder to the filter list box in the result editor.

Using scriptlets in operations

A scriptlet is a Sleep or JavaScript (Rhino) script that is contained in an operation. Scriptlets provide more control than operation output filters and response, for evaluating output, determining results and responses, or adding variables to the flow.

Scriptlets are also uniquely suited to obtain various side effects subsequent to the operation's core actions and to otherwise extend the operation's capabilities.

Assuming that you know how to script in either the two scripting languages used in PAS, JavaScript (Rhino) or Sleep, you can use the PAS scriptlet templates for those languages to learn the syntax and objects that PAS requires for exchanging information with the scripting language. The scriptlet templates are available on the scriptlet tab of an operation's or step's **Properties** sheet.

Note: PAS automatically creates from each input a variable of the same name. Therefore, a scriptlet can access values by using variables with the same name as the inputs.

Creating a scriptlet


To view a scriptlet or the scriptlet template

1. On an operation's or step's **Properties** sheet, click the **Scriptlet** tab.
2. If the operation does not already contain a scriptlet, then to view the scriptlet template, click **Insert Template** and select the language in which you will write your script.

For examples of existing scriptlets, look in operations in the content that ships with PAS (such as the operations under the iConclude/Red Hat folder).


Debugging a scriptlet

To debug a scriptlet

1. Before starting to debug the Ops flow, from the **Tools** menu, click **JavaScript Debugger**.
The JavaScript Debugger starts.
2. To start debugging the flow, ALT+TAB back to PAS Studio and click the Execute Flow button ().

Saving a scriptlet for use elsewhere

To save a scriptlet in the Scriptlets filter

1. In the scriptlet editor, select the scriptlet, then click the Scriptlet icon () and drag to the **Configuration\Scriptlets** folder.
2. In the **Configuration\Scriptlets** folder, name the scriptlet.
3. Save your work.

Creating a new operation

Depending on the kind of operation you need, creating the operation involves some of the following, which are treated elsewhere in this Help system:

- Defining [Inputs: Providing data to operations](#) and [Operation results and responses](#) for steps and operations
- Writing a [scriptlet](#)
- Specifying [responses](#) that determine transitions and mapping results to the responses
- Assigning the operation's [access permissions](#) to PAS roles as appropriate
- Storing output data in [flow variables](#)

A valid operation requires:

- One or more inputs, with a defined data source for each input.
- Responses that are mapped to valid expressions describing outcomes of the operation.

- Properties specified as necessary for the type of operation.

After you create an operation, you complete the operation's **Properties** sheet. The following procedure tells you about the Properties sheet items that are common to all the operation types. After you finish this procedure, specify the properties that are particular to the operation type that you have created. Operation-specific properties are described in the list in [Types of operations: setting properties](#), following this procedure.

To create an operation

1. Right-click the folder in which you want to create the operation, point to **New**, then to **Operation**, and then select the desired type of operation from the list that appears.
2. In the **PAS** dialog box that appears, type a name for the new operation in the text box and then click **OK**.

Notes:

- Naming in Studio is not case-sensitive, and you cannot give two operations the same name.
 - Names can be a maximum of 128 characters long.
 - The **Properties** sheet for the new operation appears in the authoring pane.
3. To assign the operation to a category for search purposes, click **Assign Categories** and then select a category from the list.

Note: Because PAS Central users never see operations, it is not necessary to select **Mark as Hidden**.

4. If the operation requires an input, click the **Inputs** tab and then click **Add Input**.

You can obtain input values from flow variables.

5. Add any necessary inputs and assign them their data sources.
6. In the dialog that appears, type the input name and then click **OK**.

For information on adding inputs, see [Inputs: Providing data to operations](#) and, in Detailed Reference, [Inputs and flow variables: Providing the flow with data](#). For information on using flow variables when assigning a data source to an input, see [Flow variables: Making data available wherever you need it](#).

7. Add and define any output data.

For information on adding output data (results), see [Operation results and responses](#) and, in Detailed Reference, [Working with operation results](#).

8. Create any responses needed.

For information on defining rules that govern which responses are chosen for the operation, see [Responses: Evaluating results](#) and [Adding responses to the flow](#).

For information on the response definitions that are particular to each type of operation, see [Types of operations: setting properties](#), following this procedure.

9. To document the operation for the PAS Central user, click the **Description** tab and write the description in the text box.

10. Finish specifying properties as described in the following section ([Types of operations: setting properties](#)), and then click **OK**.

In the Library, if the new Ops flow is invalid or incomplete, its name is displayed in **red** type. Moving the cursor over the name of an incomplete operation displays a tool

tip that specifies how the operation is incomplete. (To review the requirements for a valid operation, see the list that comes before this procedure.)

Types of operations: setting properties

The following list of the classes of operations that you can create includes:

- Guidelines for the using the operation in PAS Central and Studio.
- A screenshot of each operation type's **Properties** sheet and notes on the properties that you must set for the operation.

For the command-line, http, secure shell, and telnet operations, you can add the op-timeout input, which is optional, and specify a time value for the input, in milliseconds. The default value is 2 minutes, or 120000 milliseconds. You add the op-timeout input as you do any other. For the steps to add an input, see the procedure "To create an operation."

Note: RAS operations cannot take op-timeout inputs, because they only act as proxies for IActions. Any operation timeouts must be programmed in the IAction.

For some of these operation types, there are various ways to create the operation. Where relevant, discussion of the operation type includes recommendations on which method to use.

Note, in the following screen shots, that the Properties sheet for an operation includes the operation's universally unique identifier (**UUID**) and last modified date.

Cmd (command-line), or shell operation

Use for existing commands and scripts that, outside of a flow, you would run from the command line.

Programming that you carry out with command operations sometimes can also be carried out with a scriptlet operation or with IAction programming. For more information on using a scriptlet, see [Scriptlet operation](#). For information on using IAction programming, see [Creating operations from RASs](#), [Creating IActions for Ops Flow Operations](#), and [RAS operation: IAction programming for remote action service \(RAS\)](#).

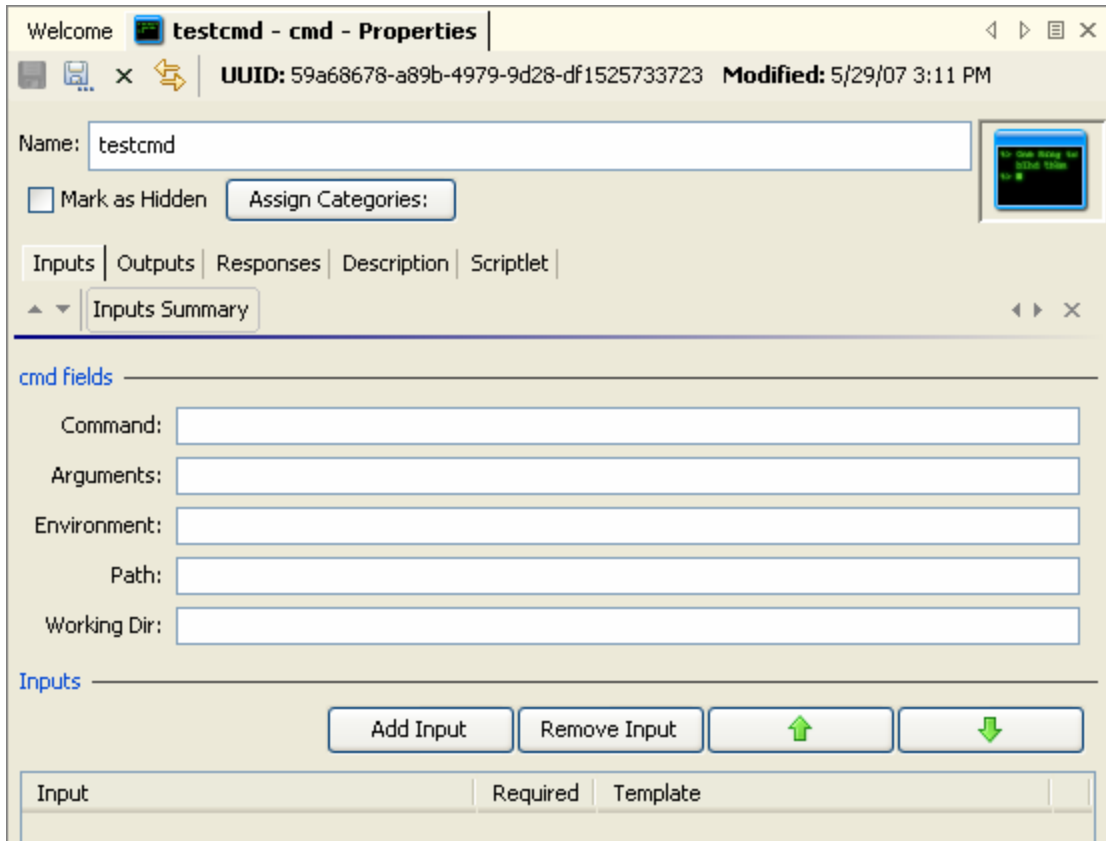


Figure 53 – Cmd (shell) operation properties

Command – The shell command that the operation executes

Arguments – The command-line arguments (switches)

Environment – Additions to the existing environment variables

Path – The path to the shell command

In the **response definition** dialog box, the following are the terms you can use to build statements to which a response is bound:

- **Code**
The numeric return code of the operation. If the operation succeeds, this value is typically 0.
- **Output String**
The standard output of the operation (what the operation writes to stdout).
- **Error String**
The error output of the operation (what the operation writes to stderr).
- **FailureMessage**
The message that you receive on failure.
- **TimedOut**
Compare **TimedOut** to "true" (without the quotation marks).

Get Stored Flow Variable operation

Use for accessing the values stored in the flow's flow variables (also known as *context keys*) in previous runs of the flow.

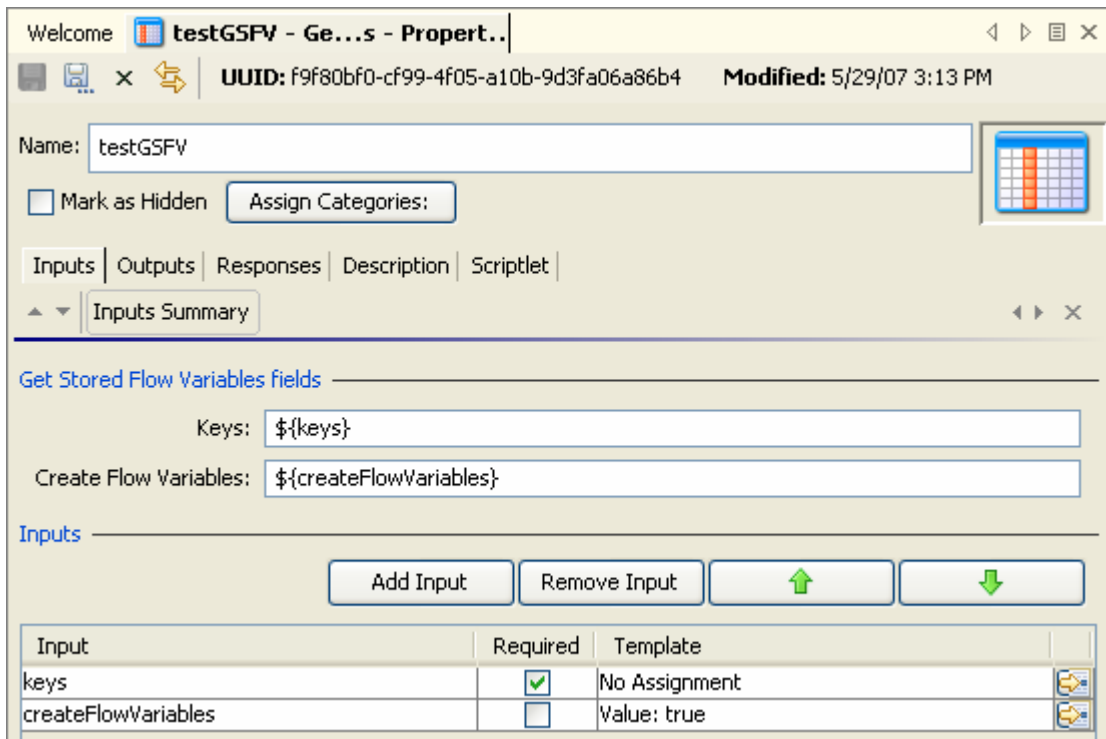


Figure 54 - Get Stored Flow Variable operation properties

For description of the fields **Keys** and **Create Flow Variables**, see the **Description** tab on an instance of the operation.

Http operation

Use for simple put or get operations, retrieving docs such as log files from an intranet or extranet.

Because IAction programming would not accomplish these tasks in a significantly different way or offer advantages over a standard http operation, this is the method of choice.

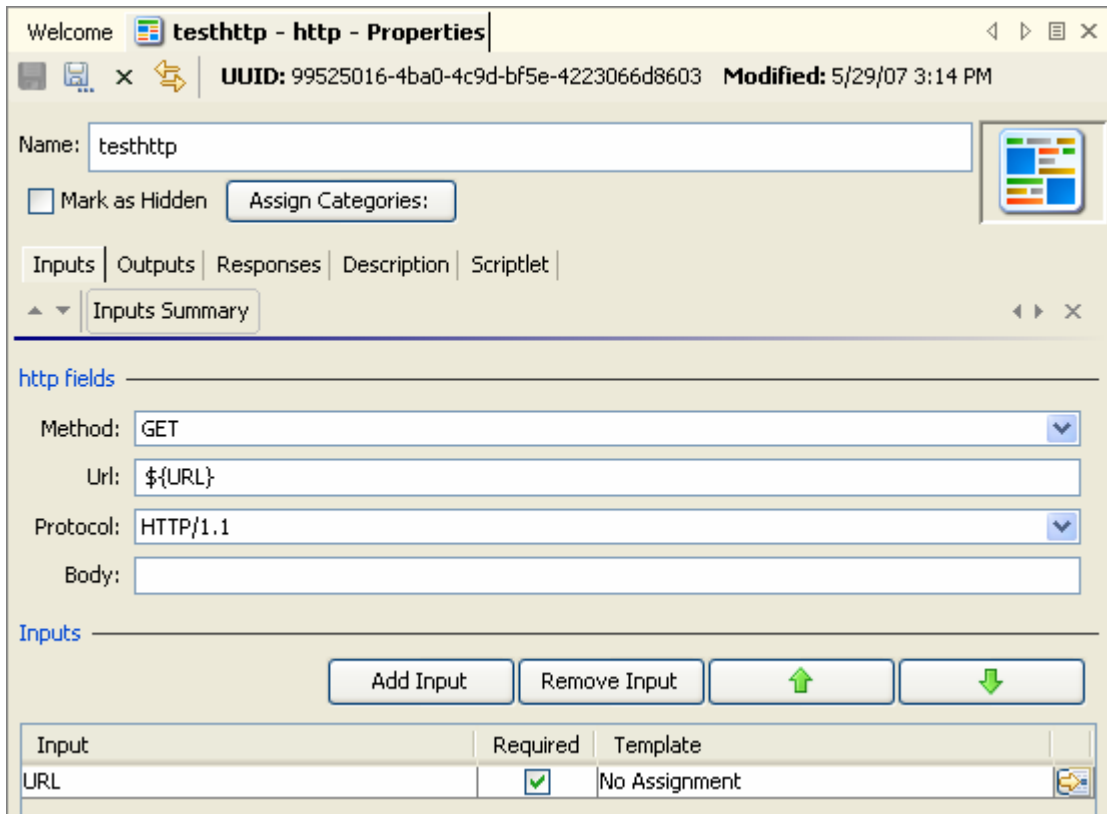


Figure 55 - Http operation properties

Method – Http protocol methods. Pick one from the drop-down list.

Url – The URL of the target Web page.

Protocol – The version of the http protocol. Pick the version that the target server supports.

Body – If the request has a body, the data of the body, as defined in RFC 2616.

In the **response definition** dialog box, the following are the terms you can use to build statements to which a response is bound:

- code =
Http return code as described in RFC 2616.
- reason =
A text string for the return code as described in RFC 2616.
- headers =
Response headers as defined in RFC 2616.
- document =
The actual document that was returned by the server.
- FailureMessage =
The message that you receive on failure.
- TimedOut =
Compare **TimedOut** to "true" (without the quotation marks).

Lock Acquire operation

Use for creating a lock in a flow with which you can limit access to the flow beyond the step that you create from this operation. The lock remains in force until the run that obtains the lock either reaches one of the flow's return steps or completes a Lock Release further along in the flow.

- You can use a Lock Acquire step for traffic or load control, to prevent more than one run of the flow from simultaneously executing the steps between the Lock Acquire step and either the end of the flow or the Lock Release step.
- To prevent undesirable repetition of a flow. For instance, you might use a Lock Acquire step in a flow that installs a security patch so that more than one Central user does not try to install the patch on a given server.

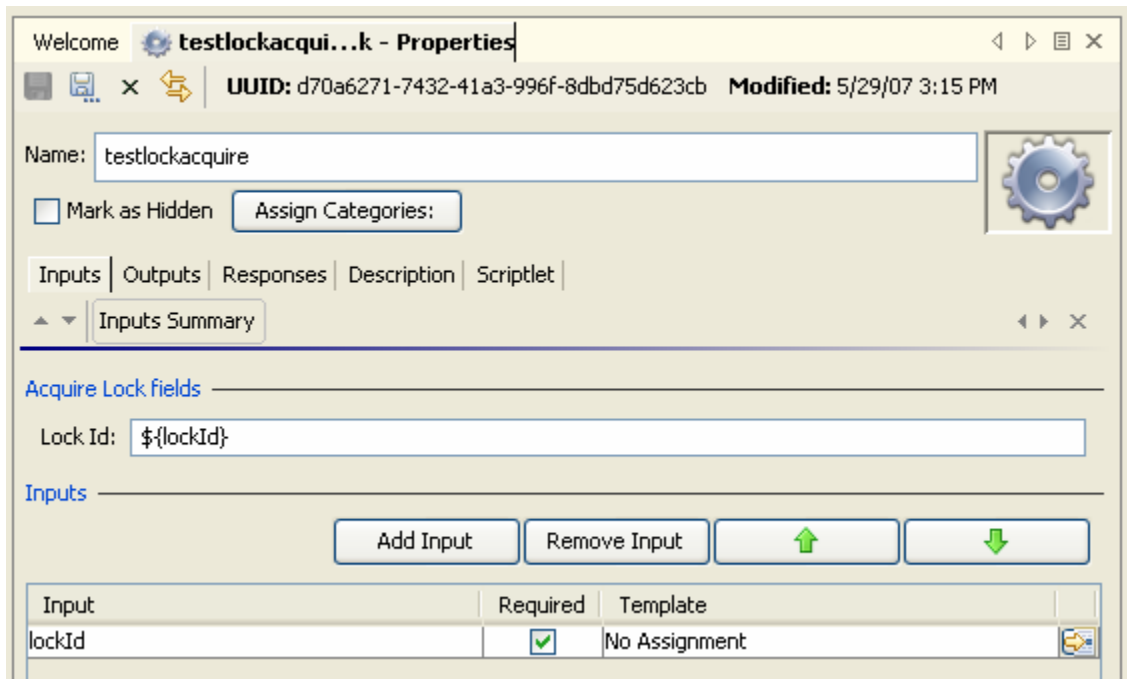


Figure 56 - Lock Acquire operation properties

For description of the **Lock Id** input, operation responses, result, and notes, see the **Description** tab on an instance of the operation.

Lock Release operation

Use for releasing the lock that you inserted into the flow with a step made from the Lock Acquire operation.

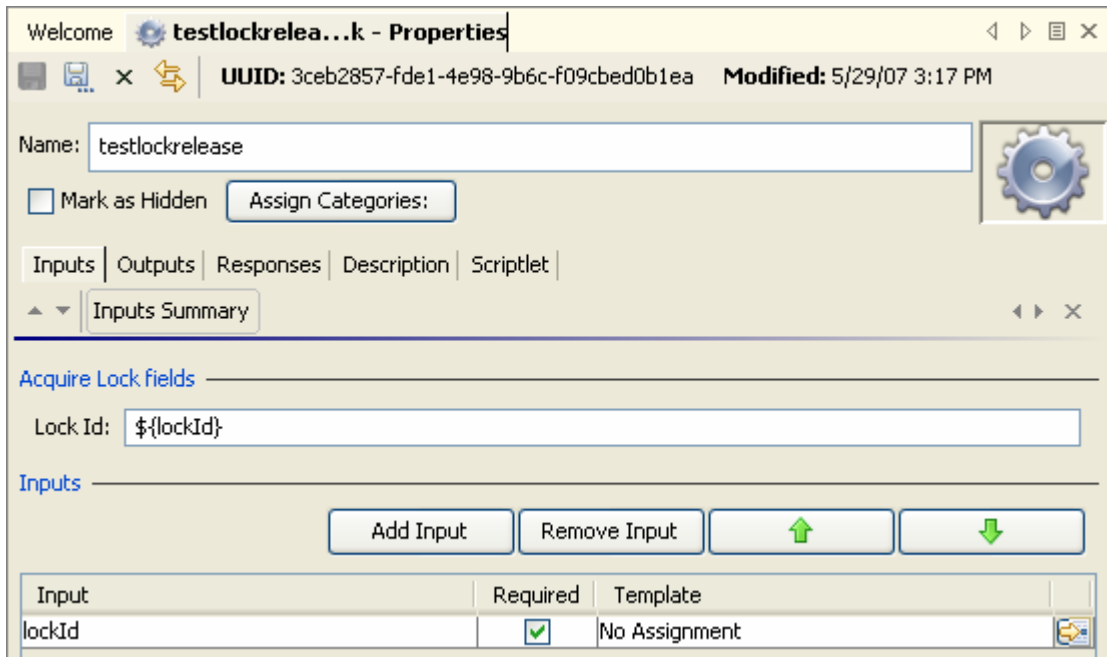


Figure 57 - Lock Release operation properties

For description of the **Lock Id** input, operation responses, result, and notes, see the **Description** tab on an instance of the operation.

Perl script operation

Use for calling a Perl script that executes your Perl scriptlet. You can pass variables to and modify their values with a Perl scriptlet within the flow, then pass the modified values back to the Perl script.

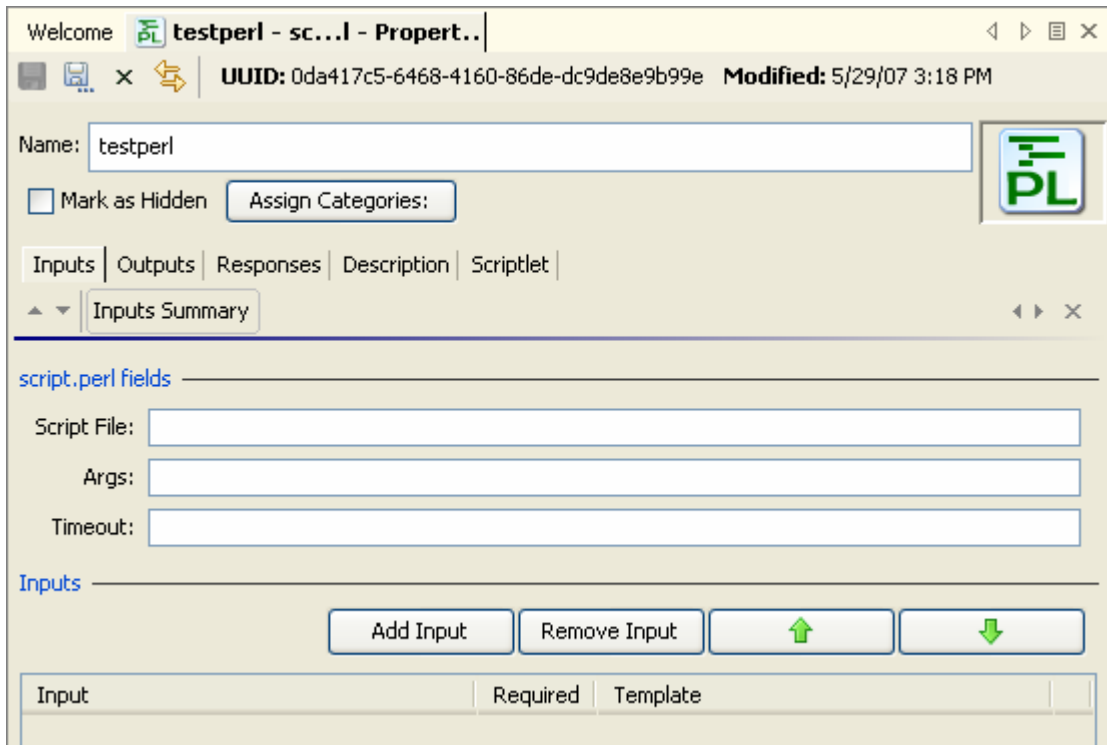


Figure 58 - Perl script operation properties

Script File – The Perl script to run

Args – The arguments that you pass to the script

Timeout – The maximum time allowed for execution of the script.

For Perl script operations, in the **response definition** dialog box, the following are the terms you can use to build statements to which a response is bound:

- **Code** =
The numeric return code of the operation. If the operation succeeds, this value is typically 0.
- **Output String** =
The standard output of the operation (what the operation writes to stdout).
- **Error String** =
The error output of the operation (what the operation writes to stderr).
- **Script Response** =
For script operations, the string that is returned by the script.
- **Script Result** =
For script operations, the output that is returned by the script.
- **FailureMessage** =
The message that you receive on failure.
- **TimedOut** =
Compare **TimedOut** to "true" (without the quotation marks).

RAS operation: IAction programming for remote action service (RAS)

Use for the bulk of work in the flow, assuming that your organization has the resources for programming IAction objects.

The advantages of IAction programming include:

- More capacity for passing large amounts of data.
- Greater speed.
- Less overhead when launching processes.
- Improved performance.
- Greater scalability.
- Greater stability.
- Ease of reuse.

Programming that you carry out with scriptlet operations can also be carried out with command or IAction programming operations. For more information on when command or IAction programming operations are indicated, see their descriptions in this list.

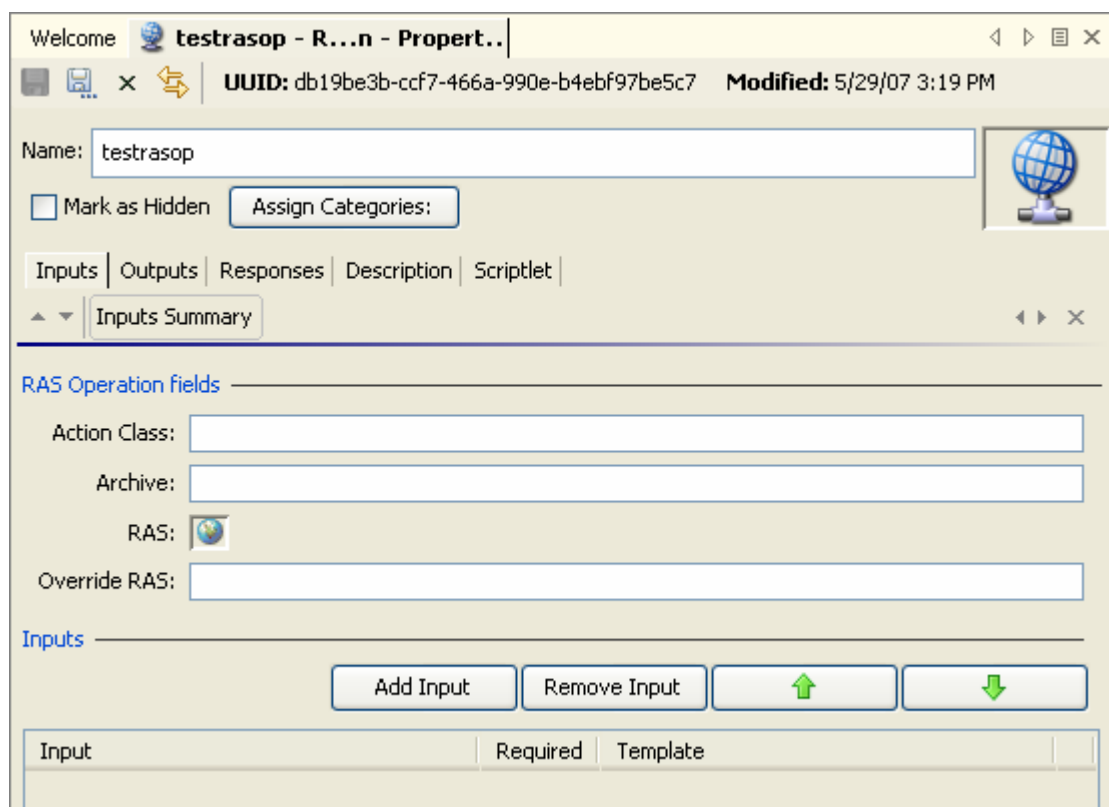


Figure 59 – RAS operation properties

Action Class – The name of the action class from which the operation is to be created.

Archive – The name of the Java Archive (JAR) file or dynamic-link library (DLL) file in which the action class is contained

Service Name – The name of the RAS reference that you configured in PAS Studio. The reference is to a JRAS, NRAS, or custom RAS that you installed, on which the operation executes the action class.

In the **response definition** dialog box, the following are the terms you can use to build statements to which a response is bound:

- FailureMessage =
The message that you receive on failure.
- TimedOut =
Compare **TimedOut** to “true” (without the quotation marks).

For more information on RASs, JARs, and DLLs and how you can use them to extend the functionality of PAS Ops flows, see [Operating outside PAS Central with remote action services](#) and [Creating IActions for Ops Flow Operations](#).

Scriptlet operation

Use for creating a scriptlet that you want to have available for reuse in multiple flows.

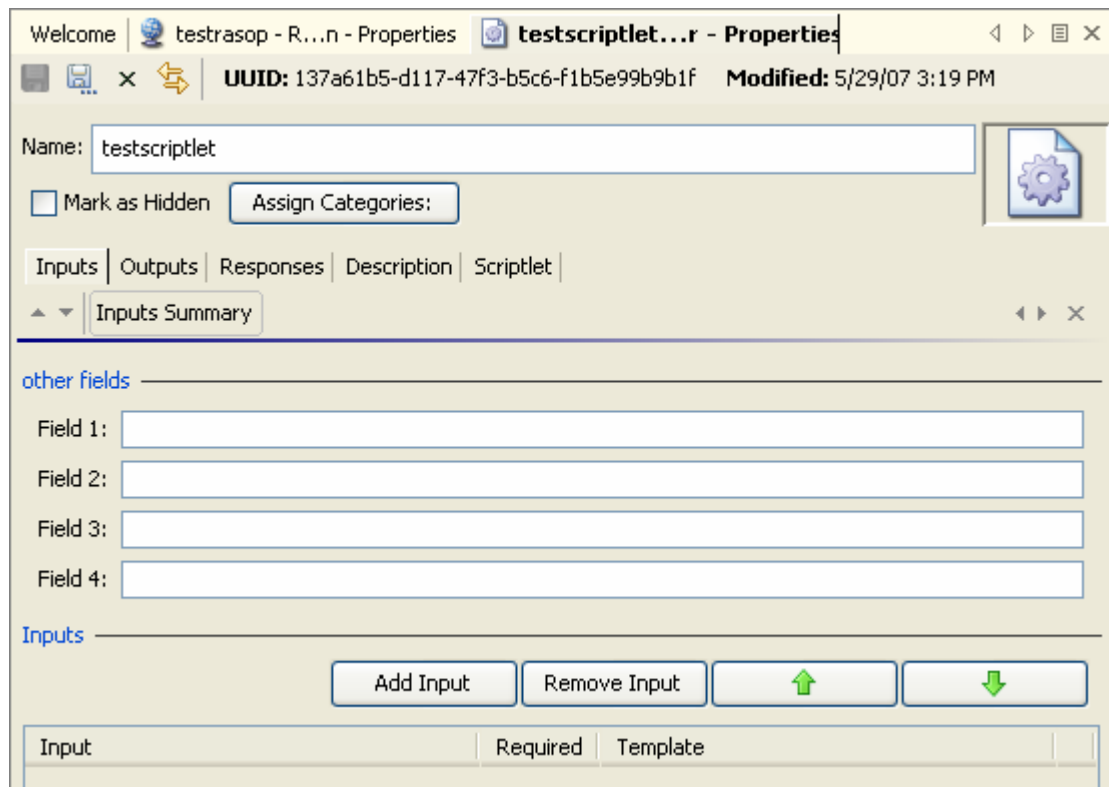


Figure 60 - Scriptlet operation properties

Use for gathering up to four pieces of data and passing them to flow variables for use elsewhere in the flow.

One of the uses of a Scriptlet operation is to provide user prompts when, in order for the flow to run without human intervention, the step must be able to alternatively obtain the prompt's input value from a flow variable.

Fields 1 through 4 – The four fields that you can populate using the inputs that you define for this operation.

For scriptlet operations, in the **response definition** dialog box, the following are the terms you can use to build statements to which a response is bound:

- **Fields 1 through 4**
The contents of the four fields in which you have stored the data from the inputs.
- **FailureMessage**
The message that you receive on failure.
- **TimedOut**
Compare **TimedOut** to "true" (without the quotation marks).
- **Result**
The output that is returned by the script.

Secure shell (ssh) operation

Use for standard http operations that use already-existing secure communications.

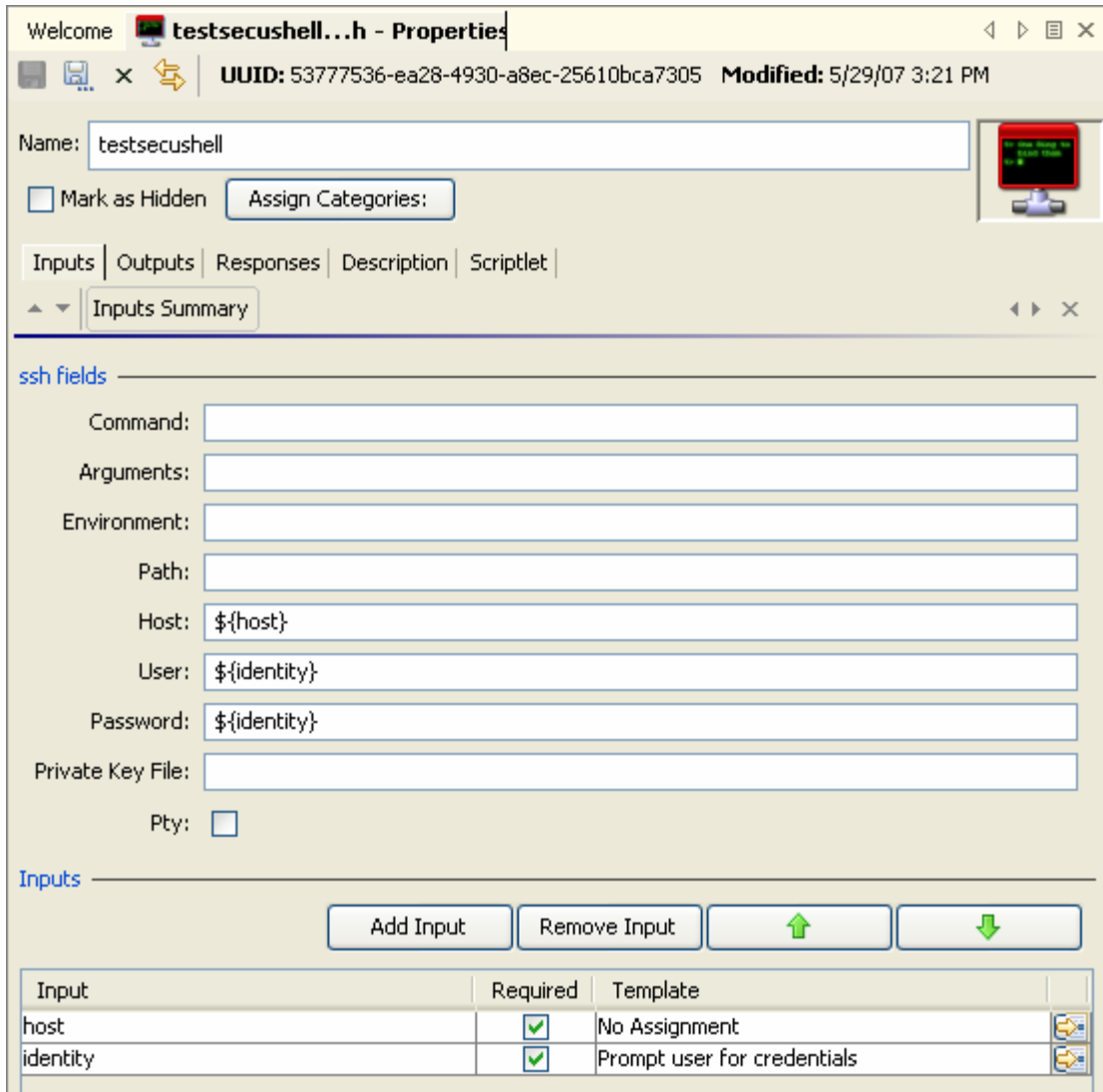


Figure 61 - Secure shell operation properties

Command – The shell command that the operation executes

Arguments – The command-line arguments (switches)

Environment – Additions to the existing environment variables

Path – The path to the shell command

Host – The target computer for the secure shell command

Tip: To specify a particular port for the host, specify the host and port with the following syntax:

`<hostname>:<portname>`

User – The user account for the command on the target machine. By default, the user account is extracted from the system account (which is stored in the flow variable "identity") that the user is prompted to enter when the flow is run.

Password – The password for the **User** login. By default, the password is extracted from the system account (which is stored in the flow variable “identity”) that the user is prompted to enter when the flow is run.

Private Key File – The path and name of the file. You supply the path and file name only if you have a private key that you use to authenticate with the target machine using a certificate.

Pty (checkbox) – The checkbox creates a pseudoterminal so that Unix operations that require a terminal can be run from a secure shell operation that you create.

Warning: A pseudoterminal does not distinguish between the standard output (or output string) and standard error (or error string) but takes both. As a result, the error string is always empty.

In the **response definition** dialog box, the following are the terms you can use to build statements to which a response is bound:

- Code =
The numeric return code of the operation. If the operation succeeds, this value is typically 0.
- Output String =
The standard output of the operation (what the operation writes to stdout).
- Error String =
The error output of the operation (what the operation writes to stderr).
- FailureMessage =
The message that you receive on failure.
- TimedOut =
Compare **TimedOut** to “true” (without the quotation marks).

Store Flow Variable operation

Use for storing values in the flow variables that you define with **keyn** inputs.

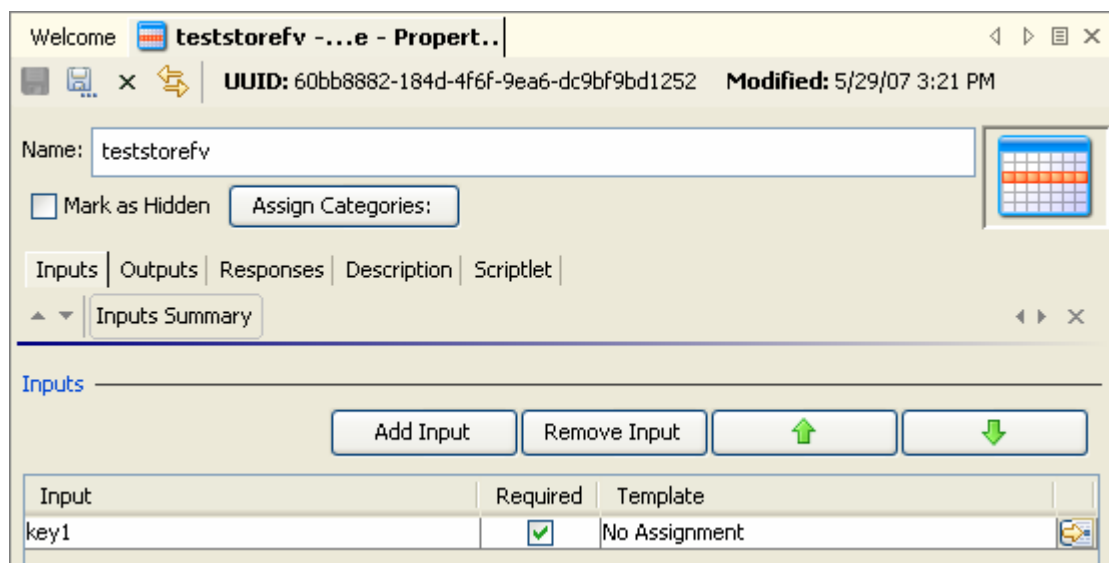


Figure 62 - Store Flow Variable operation properties

As you create each **key** input (you can, of course, rename the input), you click the right-pointing arrow at the end of the input's row. In the input editor that appears, define the flow variable that the input will store the value in. This is the same input editor that you have encountered elsewhere; for more information on using its controls, see [Inputs and flow variables: Providing the flow with data](#).

Telnet operation

Use for sending messages or commands to a server using the telnet protocol.

Welcome **testtelnet - ...t - Propert..**

UUID: d3320b19-f3d8-4e61-8723-76704feb0b68 Modified: 5/29/07 3:23 PM

Name: testtelnet

Mark as Hidden Assign Categories:

Inputs | Outputs | Responses | Description | Scriptlet

Inputs Summary

telnet fields

Command:

Arguments:

Environment:

Path:

Host:

Port:

Session Name:

Terminal Expression:

Inputs

Add Input Remove Input

Input	Required	Template
-------	----------	----------

Figure 63 - Telnet operation properties

Command – The shell command that the operation executes

Arguments – The command-line arguments (switches)

Environment – Additions to the existing environment variables

Path – The path to the shell command

Tip: To specify a particular port for the host, specify the host and port with the following syntax:

`<hostname> <portname>`

Note that for telnet operations, the separator between `<hostname>` and `<portname>` is a whitespace.

Host – The target computer for the secure shell command

Port – The port through which the operation connects to the target computer

Session Name – The name for the session that you want to store in the context

In the **Response Definition** dialog box, the following are the terms you can use to build statements to which a response is bound:

- **Code** =
The numeric return code of the operation. If the operation succeeds, this value is typically 0.
- **Output String** =
The standard output of the operation (what the operation writes to stdout).
- **Error String** =
The error output of the operation (what the operation writes to stderr).
- **FailureMessage** =
The message that you receive on failure.
- **TimedOut** =
Compare **TimedOut** to “true” (without the quotation marks).

Adding responses to the flow

Flow responses are the possible outcomes of the flow run—such as, whether the system that the flow appraised needed fixing or whether the action was successful.

For example, the Restart Service – Tutorial Flow has three responses. To see how the following responses are obtained, in Studio, find and double-click the Restart Service – Tutorial Flow in the Library.

- **Failure**—The flow was unable either to get a list of services, or the service that it was running against was stopped and the flow could not restart it.
- **Service Already Running**—The flow found the service and tried to restart it, but the service was already running.
- **Started Service**—The flow successfully started the service.

To add a response to the Ops flow

1. On the flow **Properties** sheet (with the flow diagram open, click the Properties tab at the bottom of the window), click **Responses**.
2. Click **Add Response** and then, in the text box that appears, type the name of the response.




Tip: To delete a flow response, on the **Outputs** tab, click the response you want to delete and then click **Remove Response**.

3. To close the **Properties** sheet and save changes, click **OK** and then press CTL+S.

Return steps

Return steps are the step representation of flow responses. There are four kinds of return operations, which indicate four primary possible end states to the flow.

- **Resolved:** 

- Diagnosed: 
- No Action Taken: 
- Error: Failure: 

These return steps are sealed, so you cannot modify them.

To add a return step to the Ops flow

1. From the flow diagram toolbar, drag the appropriate return operation to the Ops flow canvas.



Figure 64 - Return step icons

You can change the response of a return step to more accurately reflect the outcome that led to the return step. For example, if the outcome that led to an Error:Failure return step were 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.

2. To change the response of a return step, in the flow diagram, right-click the return step, point to **Select Response**, and click the desired response.

OR

To create and assign a new response to the return step, right-click the return step, point to **Select Response**, and click **Add New Response**, then name the new response.

Associating a step with a different operation

Suppose that you discover that you need a different operation for some step in your Ops flow, but you want to keep the existing transitions to and from that step. The step Inspector helps you accomplish this.

To associate a step with a different operation

1. In the flow's design view open and the step selected, open the step's Inspector.
 - Click the **Advanced** tab of the Inspector.

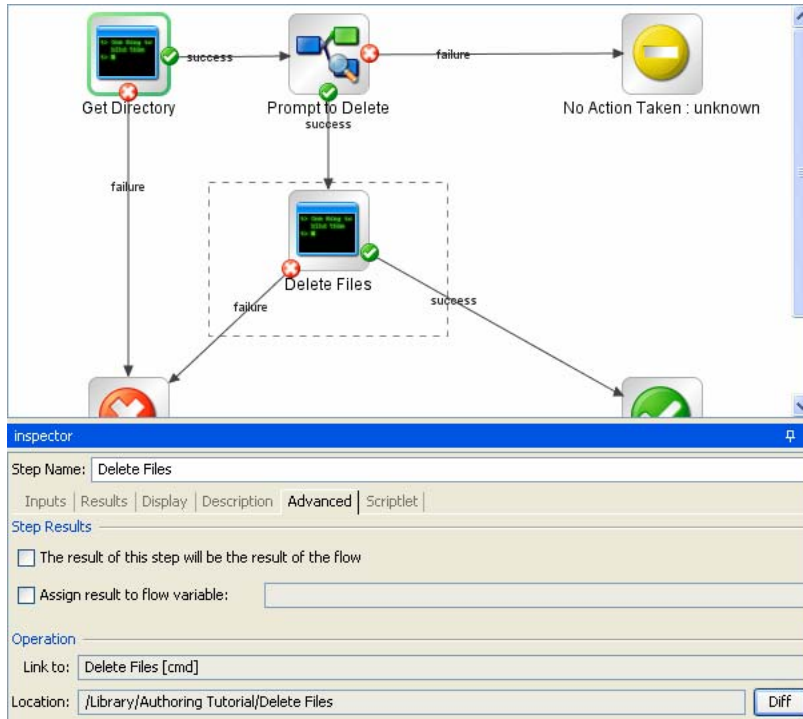


Figure 65 - Step Inspector Advanced tab

- Under **Operation**, click **Diff**.

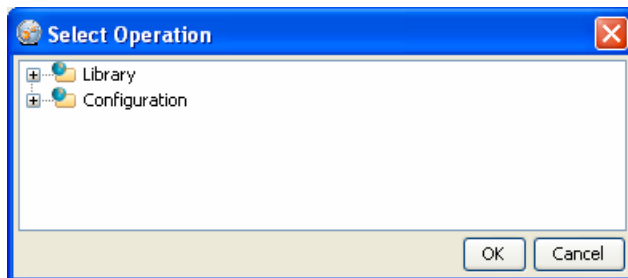


Figure 66 - Select Operation box

- If you need to change any step connections, do so, and then save your work.

Debugging Ops flows

PAS Studio includes three important tools for solving problems in operations and Ops flows:

- **Execute** flow sheet, which tracks the execution of an Ops flow in real time and provides the raw results of the flows operations, including error strings.
- **Problems** panel, which displays problems in the currently selected flow and operations
- **Validate Repository** command (on the Tools menu), which displays, in the Problems panel, problems for all the Ops flows and operations in the entire Library

The **Problems** panel reflects any problems that it encounters in real time, as you work.

To view problems from the entire repository

- On the **Tools** menu, choose **Validate Repository**.

As you execute a flow in order to debug it, the **Execute** sheet of the authoring pane looks like the following:

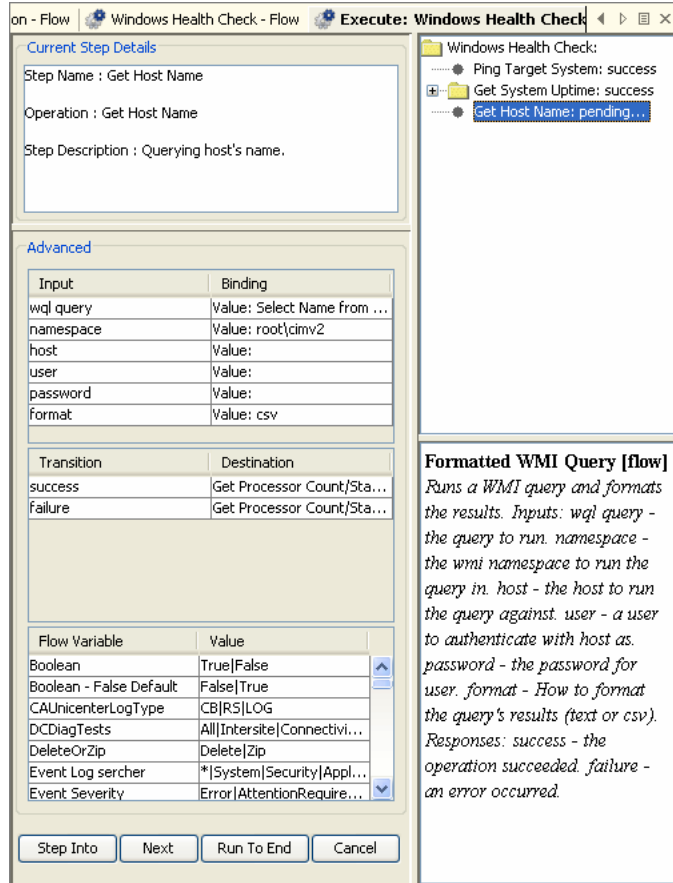


Figure 67 - Execute (debug) sheet

The **Advanced** area shows information on each step when the step is pending.

To view the raw results of a step's operation

- After the run has completed, highlight the step whose raw results are of interest. Raw results appear in the lower-right panel.

Working with regular expressions

Regular expressions (also known as regexes) are a powerful tool that you can use in Studio to:

- Extract key pieces of data from raw results strings for:
 - Saving in variables for use in later operations.
 - Testing to determine a step's response.
- Validate the form of inputs.

For example, suppose the information that an input variable needs from a Ping operation is the packet loss. The output of pinging the IP address 111.111.111.111 looks like the following:

```
Pinging 111.111.111.111 with 32 bytes of data:
```

```
Reply from 111.111.111.111: bytes=32 time=27ms TTL=246
Reply from 111.111.111.111: bytes=32 time=26ms TTL=246
Reply from 111.111.111.111: bytes=32 time=29ms TTL=246
Reply from 111.111.111.111: bytes=32 time=28ms TTL=246
```

```
Ping statistics for 111.111.111.111:
```

```
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 26ms, Maximum = 29ms, Average = 27ms
```

To extract the number of packets lost, you can instruct PAS to search for the string "Lost = " followed by any number. Using a regular expression will help.

A regular expression allows you to search not only for exact text but also for classes of characters as well. For example to match any digit you can use the wildcard `\d`.

Therefore, in the above example, a search using the regular expression `Lost = \d` would return the string **Lost = 0**.

The key wildcards for regular expressions are:

Wildcard	Use
<code>^</code>	Matches the beginning of a string
<code>\$</code>	Matches the end of a string
<code>.</code>	Any character except newline
<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>	Newline
<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

You can modify these wildcards in several ways:

Modifier	Effect
*	Match zero or more
+	Match one or more
?	Match zero or one
{n}	Match exactly n occurrences
{n,}	Match n or more occurrences
{n,m}	Match between n and m occurrences
[abc]	Match either a, b, or c
[^abc]	Match anything except a, b or c
[a-c]	Match anything between a and c
a b	Match a or b
\	Escape a special character (for example \. Means '.' not match anything)

Using these wildcards and modifiers, the following regex extracts the IP address from the ping output:

```
\d{1-3}\.\d{1-3}\.\d{1-3}\.\d{1-3}
```

When using the results filter in Studio, you can combine multiple regexes to isolate the value that you need. For example, consider the output of the Unix ps command:

F	S	UID	PID	PPID	C	PRI	NI
0	S	512	21604	21603	0	75	0
0	R	512	2659	21604	0	76	0

You could create the following filters to extract the time for the ps command:

1. In Studio, open the Unix ps operation.
 - Open the **Output String Field Filters** dialog box.
For information on opening the **Output String Field Filters** dialog box and performing the following tasks, see [Filter details](#).
Extracting the time for ps requires two regexes, one to filter the output down to the line for ps and the second to extract the time.
 - Add a new regular expression filter.

- For the expression value, type `.*ps` (any characters ending with “ps”; be sure not to omit the leading period [.] and check the **Filter line by line** box.
- Click **Test Selected**.
In the **Test Output** box, the only output is the line containing “ps”.
- To extract the time from the line, add a new regular expression filter.
- For the expression value, type `\d*:\d*:\d*` (three sets of digits separated by colons)
- Click **Test Selected**.

The test output now shows only the time from the ps line. Now you can assign this value to a variable.

For information on storing values in variables, see [Inputs and flow variables: Providing the flow with data](#) or [Flow variables: Making data available wherever you need it](#).

Changing and testing Ops flows

It is critical that authors never make and save changes to an Ops flow that is in use (that is, that an PAS Central user is currently running or has paused, or that is in the process of being handed off). If changes to an Ops flow that is in use are saved to the database, any current runs of the Ops flow break and data related to the Ops flow and the Ops flow runs can be corrupted or lost.

To avoid corrupting or losing data when making changes to Ops flows that are in use, make sure that there are no current runs of the Ops flow before saving or copying changes to the production database. The safest method is to:

1. Use a staging server to test changes to a flow.
2. Deploy the changed version of the flow during non-production hours.

To deploy changes to an existing flow from a test server

1. On the development server, make changes to and test the flow.
2. After saving changes, export the flow as a repository.
 - **Note:** Unsaved changes in a flow or operation are not included in the export.
3. Publish the flow to the staging server and test it there.
4. To see whether there are any current runs of the flow:
 - Open the PAS Central Web site, logging in with an account that has PAS administrator rights.
 - Click the **Administration** tab.
Any current runs of flows are listed under **Run Administration**.
5. Cancel any current runs of the flow you’re going to change, and instruct Central users not to start any new runs of the flow.
6. Publish from the staging server to the production server .
7. To test the flow on the production server, set its properties so that only you have access permissions to run the flow, then test it.
8. After the flow tests successfully, grant access permissions to appropriate end-users.

Restricting access to Ops flows

You can restrict who can author or use an Ops flow in three ways:

- Restricting access permissions, specifying which PAS role can see, work on, or execute the flow in PAS Studio.

Groups (ADMINISTRATOR, AUDITOR, LEVEL_ONE, LEVEL_TWO, and LEVEL_THREE) are the units to which you grant permissions. So if you publish your repository but don't want another author who updates the repository to be able to work on the flow, you could specify that that author's group does not have execute permission for the flow.

Note: The ADMINISTRATOR group has full permissions for all objects.

For information on defining PAS roles or mapping external roles or groups to the PAS groups described here, see the *PAS Administration Guide* and Help for PAS Central.

- Hiding the flow from Ops Central users
If your organization does not use a staging server for testing Ops flows and you publish a flow that is not ready for production use in Central, you can hide it from Central users.

For more information, see [Controlling who can see a flow in Central](#).

- Limiting access to a flow beyond a given transition: Specifying in a transition's properties which PAS role members can continue running the flow beyond the transition.

Suppose one part of a flow can be run by a member of the LEVEL_1 PAS role but a subsequent part of the flow needs to be escalated because it requires user credentials that a LEVEL_1 user doesn't have. When you create the flow, you could make the transition that leads to the step where the credentials are required a gated transition: one that requires membership in a certain role for the user to continue running the flow.

For more information on creating gated transitions, see [Transitions: connecting responses to steps](#).

- Adding an Acquire Lock step to the flow that prevents anyone but a user who possesses the key to the lock from continuing the flow run beyond that point.
For more information on creating Acquire Lock and Release Lock operations, see [Types of operations: setting properties](#).

Permissions

Permissions are access rights to individual objects, such as individual folders, flows, operations, or system accounts. To find and run an Ops flow in PAS Central, users must have read and execute permissions for the flow. In Studio, authors must have read, write, link, and/or execute permissions for objects that they use to author flows. For instance:

- To debug a flow, an author must have the execute permission for that flow.
- A flow author must have the Link permission for any flow or operation from which he or she creates a step in a flow.
- To change a system account, an author must have the Read and Write permissions for the system account.

The following two tables describe the permissions and which of them are needed for objects in PAS Studio.

Permissions for PAS objects

Permission	Description
Read (R)	Can view the object in Studio or Central
Write (W)	Can change the object
Execute (X)	Can start a run of the flow. This is not a recursive requirement. That is, for a Central user to run a flow or for an author to debug a flow, he or she does not have to have execute permission for every object, such as operations and configurable items such as system accounts, associated with the flow.
Link to (L)	Can use the flow or operation to create a flow step.

PAS objects and the permissions needed to work with them

Object	Action	Necessary permission(s)
Folder	View contents	Read
	Add to contents	Read, Write
		Read, Write (also needed for all children of the folder)
	Move	Read, Write
	Rename	Read, Write
Flow or operation	View/open	Read
	Modify	Read, Write
	Rename	Read, Write
	Execute/Run	Read, Execute
	Use as a step or sub-flow	Link To
System account	View account name	Read
	Change account password	Read + Write
	Rename account	Read + Write
	Use in flow or operation	Link To
	Use at runtime	Execute

For procedures for managing group membership and managing capabilities, see Help for PAS Central.

You can assign these access permissions for individual flows, properties, and selection lists, or you can set default access permissions for groups. The access permissions that you specify are granted to those groups for each new flow or operation that is created.

Note: You cannot specify permissions for the ADMINISTRATOR role, because it has all permission for all the objects in PAS, and these permissions cannot be revoked.

To assign access permissions to an PAS object

1. In the Library, right-click the object and then click **Permissions**.
2. In the following dialog, select any appropriate permissions for each group of users, and then click **OK**.

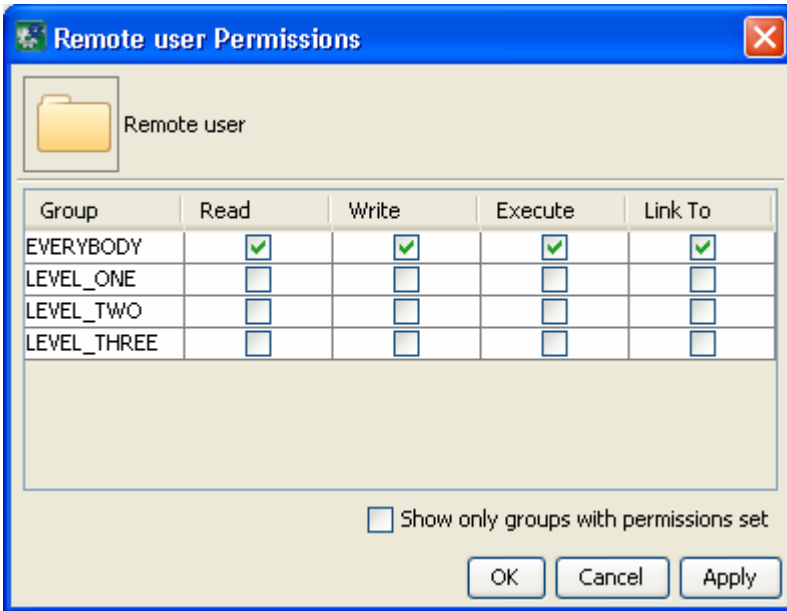


Figure 68 – Setting permissions

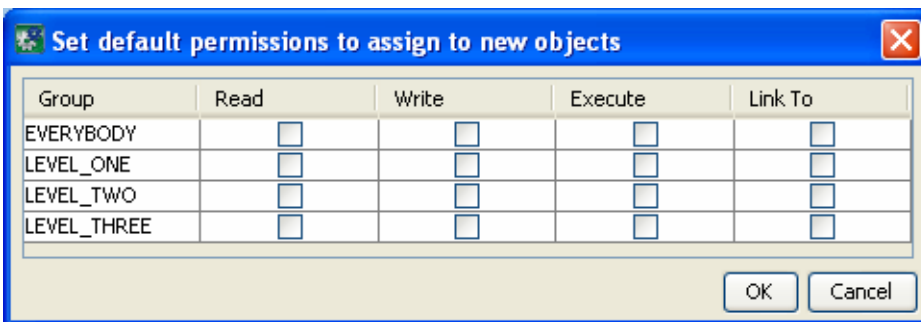
When you select or unselect a permission for a group, the box for that permission is outlined in black.

3. For each parent folder of the Ops flow, specify the same permissions that you did for the Ops flow.

You can also grant default permissions for groups, so that they have the permissions you specify for every new folder, flow, operation, and system account that you create. For instance, to allow a group to debug but not author flows, you could grant its members Read, Execute, and Link permissions, but not

To set default access permissions for groups

1. On the **Tools** menu, click **Set Group Mask**.
2. In the following dialog, select the permissions for each group that you want that group to have in Studio and Central by default on all new PAS objects, and then click **OK**.



Controlling who can see a flow in Central

There are two ways to hide flows from PAS Central users:

- Marking the flow as hidden hides it from all Central users, regardless of their access permissions or group capabilities.

- To specify who can see the flow and who can't, specify which groups have read permissions for the flow.

To mark a flow as hidden

1. Open the Ops flow and then click the **Properties** tab.
2. Select **Mark as Hidden**, and then save your work.

To select which groups have the read permission

1. In the Library, right-click the flow, and then click **Permissions**.
2. In the **Permissions** dialog that appears, select the **Read** permission for the groups you want to be able to see the flow in Central and Studio, and then click **OK**.

Backing up and restoring repositories

The public repository (the repository that PAS Central uses) and the Studio default local repository are each automatically backed up to a .jar file when you start Studio and when you save an object. By default, 10 backup .jar files are kept. On the 11th save, the oldest backup file is deleted. However, you can also manually back up the private repository.

To back up a repository

1. Open the repository that you want to back up.
2. From the **Repository** menu, choose **Create Backup**.

To restore a repository

1. Unpack the .jar file that was created by the backup.
2. From the **Repository** menu, choose **Import Repository**.
3. In the Select **Repository Directory** dialog box, navigate to the folder that contains the .jar file that defines the repository.
The folder that contains the .jar file contains two folders, **backups** and **data**.
4. Click **Open**.
After checking the PAS Central public repository for version conflicts with the objects that you want to import, PAS imports the repository.
5. If there are conflicts, resolve them as you do when you import repositories otherwise.
For more information on importing repositories, see [Importing a repository](#).

Operating outside PAS Central with remote action services

A RAS enables PAS Central to run operations that are not defined in the Central repository and to run operations on machines that are remote from the Central server – or, indeed, anywhere on the internet, even on the other side of firewalls. By using RASs, you can integrate PAS with the programming interface of any other application or platform. For instance, suppose you need to restart a server in a different domain from the one in which Central is installed. If you have installed a RAS on a machine in the domain of the server you're going to restart, you can run a flow that restarts the server by pointing the flow at the RAS on that domain.

You can also use a RAS to:

- Carry out an operation on a Windows server when PAS Central was installed on a Linux server.
- Distribute the load of your automation across multiple servers.

When you integrate PAS with another API, you use the Opsware Software Development Kit (SDK) and C# or Java to implement the IAction interface, then deploy the operation to the Java or .NET remote action service (RAS). This level of authoring requires the ability to program with .NET or Java as well as Windows Scripting Host or Perl scripting.

How Central runs RAS-dependent operations

The RAS contains a dynamic-link library (DLL, or .dll) or Java archive (.jar) that contains definitions of operation classes. When the RAS is called, it runs an instance of the class of the required operation.

A RAS-dependent operation contains the information that Central needs to identify the RAS on which the operation must run:

- **Action Class** is the name of the operation that is inside the archive or DLL and is the op that you want to run.
- **Archive** is a DLL (.dll) or Java archive (.jar), such as Dotwebactions.dll, that contains operations.
- **Servicename** is [RAS]_operator_path, which is either the default URL or the URL that you specified when installing the RAS on a remote machine.

These pieces of information are specified in the **RAS Operation fields** on the operation's **Properties** sheet. Operations in the **Operations** folder in the Library or an Accelerator Pack already have this information provided. If you change the URL of the RAS operator path, you must change the URL of the RAS service in PAS Studio. If you create an operation from scratch, you must:

- Create a DLL or .jar to contain the operation's definition.
- Specify that DLL or .jar in the installation of a RAS service for running the operation.
- Provide the name of the operation class, the DLL or .jar, and the name of the RAS service in the operation's **RAS Operation fields**.

Creating operations that access Web services

A Web service is a Web-based piece of programming logic that performs a particular set of functions. If you want to integrate an Ops flow with a Web service, monitor it, or validate that it is running, you can access the Web service through the methods that are defined in the Web service's WSDL (Web Service Description Language) file, which defines the interface of the Web service. To access a Web service, you create a SOAP (Simple Object Access Protocol) message (written in XML) and send it to the Web service. The Web service responds to the SOAP message and returns XML.

For the purposes of an Ops flow, the WSDL defines the methods that an Ops flow operation can use to interact with the Web service. The Web service Wizard is a tool that displays a list of the methods in the interface of the Web service that you specify in the Wizard. Within the Wizard, you pick the methods that you want to use,

and with one click, for each method you have selected, the Wizard creates an PAS operation that can execute the method.

Or suppose that you want an operation to perform a Web search using the Google search engine. You would locate the GoogleSearch Web service WSDL and navigate to the WSDL in the Web Service Wizard. From the list of methods that appear in the Wizard, you would select doGoogleSearch and, by clicking Generate XML, create the XML that defines an operation that performs the Google search.

Note: To use RAS-dependent operations, you must also configure PAS for extended functionality. For information on doing so, see *Administering Opaware PAS* (PAS_AdminGuide.pdf).

For information on using the Webservice Wizard, see [Creating operations from Web services](#).

Troubleshooting the running of RAS-dependent operations

There are three reasons that Central could fail when attempting to run a RAS-dependent operation:

- The Servicename does not point at the correct RAS service (the RAS that contains the DLL or .jar that contains the definition of the op you want to run).
- The RAS service was never installed on the target machine.
- The library is not present in the RAS service.

PAS Studio example: An Ops flow that runs in any of multiple domains

By using a flow variable, you can create a flow that chooses the correct RAS on which to run an operation remotely. Basically, you store the name of the desired RAS service in the flow variable, and Ops flow logic determines which RAS service that is.

Suppose you want to restart either ServerA in DomainA, ServerB in DomainB, or ServerC in DomainC. With the following steps in the flow, the Central user can decide, when he or she runs the flow, which server to run the flow against.

1. Create a scriptlet step that prompts the user for the server name, including the domain name.
- Write the step's scriptlet to:
 - Extract the domain name.
 - Depending on the domain name, store the name of the RAS that serves that domain in the flow variable "rasname".
 - Provide the flow variable rasname as an input to the RAS operation that restarts the server.

You might want to rename the RAS operation in the RAS operation's rasname property.

Creating operations from RASs

To create operations that use a RAS, you can

- Create your own RAS and operations.

After you import your Web service and operations, you move the operations from the location where they are originally imported in the Library. If you leave the operations in their original location and subsequently export any flows that use these operations, the operations are not exported with the flows. The exported flows will not work on any other machine. Moving the operations to another folder within the Library avoids this situation.

- Import and use Opware-provided content that uses the `NRAS_Operator_Path` and `JRAS_Operator_Path` remote action services.

For information on configuring a RAS in Studio, see [Creating or changing an Ops flow](#).

To import custom Web service content, you create, name, and assign a URL to a Remote Action Service. Identification of the Remote Action Service by primarily by name rather than by URL means that changing the URL in the Studio enables you to use the Remote Action Service in multiple locations.

The **Configuration** folder contains a **Remote Action Services** folder that holds the existing RASs.

To configure a remote action service

1. In the **Library** pane of Studio, open the **Configuration** folder and then the **Remote Action Services** folder.
2. Double-click either **JRAS_Operator_Path** or **NRAS_Operator_Path**.
3. On the **Remote Action Services** tab that appears in the authoring pane, click **Add**.

A new row appears in the list.

4. In the **Name** column, name the new service.
5. In the **URL** column, type a URL with one of the following syntaxes:
 - When configuring a JRAS:
`https://<yourHostname>:<yourPort>/JRAS/services/RCAgentService`
If you accepted the default port number in the JRAS installation program, the port number is 4085.
 - When configuring an NRAS, the URL have the following format:
`https://<yourHostname>:<yourPort>/NRAS/services/RCAgentService.asmx`
If you installed NRAS to run on IIS, the default port number is the IIS port number, 80.

If you installed NRAS to run standalone and accepted the default port number in the NRAS installation program, the port number is 4080.
6. Click **Check Availability**.
Doing so checks for the availability of the services' URLs and displays their availability (or lack of availability) in the **Availability** column.

If you installed NRAS to run on IIS and the service that you just configured is not available, make sure that the IIS default Web site is running on the machine.
7. When the service is available, save your changes and close the tab.

Now you're ready to import Web service content (operations and sample Ops flows) from a RAS.

To import RAS content

1. Install the RAS that contains the content you're interested in.
For information on installing a RAS, see *Installing Opsware Process Automation System* (PAS_InstallGuide.pdf).
2. Open Studio.
3. To import the RAS content, from the **File** menu, choose **Import** and complete the importing process as described in "Importing an Ops flow," as described earlier in this chapter.
4. To make sure that the RAS you're using supports your operation, you might want to review them.

Note: The JRAS (JRAS_Operator_Path) and NRAS (NRAS_Operator_Path) that are installed with PAS contain all the operations used in the iConclude Accelerator Packs and standard Library\iConclude contents.

To create operations from a RAS

1. In PAS Studio, create or highlight a folder in which you want to create operations from the Web service.
2. From the **File** menu, point to **Create Operations from RAS**.
The **Import RAS** dialog box appears.
3. In the drop-down list box, select the RAS that contains the operations you're interested in, and then click **OK**.

A folder containing the RAS operations appears in the folder that you selected in step 2. The folder is named for named for the RAS's DLL or Java archive.

If you have already created these operations in this folder, you are prompted to confirm whether you want to overwrite the operations that are already in the folder.

4. After you have created the operations, you can move it within the library.

Note: By default, the operation that you create from the IAction bases any responses only on IAction return codes. However, after you have created the operation, you can create response rules that test any other output of the IAction.

The PAS Web application is now ready for PAS Central clients to run against it. However, to have Ops flows for the Central users to run, you must install and open PAS Studio, then either import or create Ops flows and save them to the database. For information on opening the Studio, importing and creating Ops flows, and saving Ops flows to the database, see [Creating or changing an Ops flow](#).

To add an operation that is not included in the default JRAS or NRAS

1. Create an IAction that performs the task you want an operation to accomplish and store the IAction in a dynamic-link library (DLL) or JAR file.
OR
Obtain a third-party DLL or JAR that contains the desired operation.
2. To add the DLL or JAR to the PAS JRAS or NRAS, copy it to one of the following locations:
 - A DLL (added to NRAS), whether it is your own or comes from a third party:
`%PAS_HOME%\RAS\ .NET\Default\RCAgentService\services\bin\Actions\`
 - A JAR (added to JRAS):

`%PAS_HOME%\RAS\Java\Default\repository\`

- Third-party JAR (added to JRAS):

`%PAS_HOME%\RAS\Java\Default\webapp\WEB_INF\lib\`

Note: Because all RAS actions have to implement the IAction interface, in order to make the third-party library work, you probably must build an IAction class that contains the library.

3. To make sure that your remote or extended RAS operations function correctly:
 - After the RAS that you need is installed and configured in Studio, be sure to check its availability in Studio.
For information on checking the availability of an RAS, see [Checking the availability of an RAS](#).
 - You may also want to make sure that the RAS you're using supports your operation. To do so, you can import the operations in the RAS and review them. See [Creating operations from RASs](#).

Creating IActions for Ops Flow Operations

In advanced operations authoring, you use the Java Remote Action Service (JRAS), or .NET Remote Action Service (NRAS) to execute the core of an operation outside the PAS.

The Remote Action Service client is a service that manages remote actions. The JRAS or NRAS instance that you install contains the IAction interface and is an implementation of the Remote Agent Service. The Remote Agent Service communicates with the PAS Web application and PAS Studio using the SOAP/HTTP(S) protocol. The Remote Actions Service manages a database repository of IAction implementations, which are contained in Java archives (.jar files) for use in a Java environment and in dynamic-link libraries (.dll files) for use in a .NET environment.

The IAction interface uses the execute() method and methods that are specific to the Web application, standalone application, platform, or extension service on which you want actions performed. These methods map actions in the PAS Central SDK to actions of the target system's or application's SDK. The IAction interface thus mediates between the PAS and systems external to the PAS.

Standard operations interact with the PAS infrastructure (the database and Internet Information Services) and are limited to the actions that are possible within the infrastructure. When you program the IAction interface in the operation, you can interact (via the extension services) with entities outside the PAS. Thus you can create operations that:

- Interact with systems throughout the network or Internet.
- Integrate the PAS with other entities that JRAS or NRAS can interact with through IActions.

Programming operations with Web extensions requires advanced .NET or Java programming skills.

Overview of creating operations that implement IActions

Overall, the authoring process involves creating one or more IAction implementation classes and moving them into PAS Studio:

1. Create custom IAction implementation classes in your development environment and compile them into a dynamic-link library (DLL) or Java Archive (JAR) file. The following section describes programming the IAction implementation class execute() method.
2. Copy the DLL or JAR into your Web service (your Web server's \bin\Actions directory).
3. Import the Web service into PAS Studio.

Programming the IAction class execute() method for Web extensions

This section assumes that you have a background in installing J2EE Web applications.

Programming IAction implementation classes (IActions) for Web extensions requires that either JRAS or NRAS be installed. For information on installing JRAS and NRAS, see *Installing Opware Process Automation System (PAS_InstallGuide.pdf)*.

To create a new Web operation using Java IActions

1. Create a Java project in a development environment.
For instance, you might use an Ant build file.
2. Add "JRAS-sdk-1.05.jar" to classpath.
3. To create IActions, implement the IAction interface, particularly the getActionTemplate() and execute() methods.
 - To implement the IAction interface, see the PAS SDK.
 - To implement Java IActions, interfaces, and classes, see the Java Extension Service SDK.
 - To implement .NET IActions, interfaces, and classes, see the C# SDK.
4. Compile the project.
A sample ant project (with source code and reusable build file) is available at: JavaExtensionService/samples/MailAction.
5. Create a .jar file that contains all class files.
Do not include third-party libraries in the JAR file that you create.
For an example, see the sample ant project at: JavaExtensionService/samples/MailAction.
6. Copy the .jar file into the repository folder.
7. Copy all third-party .jar files into the **lib** subdirectory of your repository folder.
8. Restart the PAS Web application.

Best practices for creating custom IActions

Use Java SDK classes only in the IAction implementation class. The IAction implementation class should only translate from the JRAS SDK data types to the host application's data types (or custom data types).

Debugging IActions

The most efficient means of debugging IActions is line-by-line debugging in your preferred integrated development environment (IDE). This primarily involves connecting the IDE to the JRAS or NRAS remote debugger.

JRAS

The following procedure describes how to connect Eclipse or IntelliJ to the JRAS remote debugger, also known as, remote Java Virtual Machine (JVM) debugging.

To debug JRAS

1. Within the PAS home directory, navigate to \RAS\Java\Default\webapp\conf\.
2. Add one of the following lines to the end of the wrapper file (wrapper.conf), replacing **[open port]** with an open port number:
 - If the JRAS was developed in the Eclipse IDE:

```
wrapper.java.additional.3=-Xdebug -Xrunjdpw:transport=dt_socket,address=[open port],server=y,suspend=n
```
 - If the JRAS was developed within the IntelliJ IDE:

```
wrapper.java.additional.3=-Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdpw:transport=dt_socket,server=y,suspend=n,address=[open port]
```
3. Restart the RSJRAS service, using the following sequence of command-line commands:

```
net stop rsjras
net start rsjras
```
4. With JRAS running, set a break-point in the IAction code.
5. Connect the IDE to the JRAS remote debugger.

You are ready to start debugging the IAction.

NRAS

As with debugging JRAS IActions, a prerequisite for debugging NRAS IActions is to connect the IDE Visual Studio debugger to the NRAS service process that is already running.

To debug an NRAS IAction

1. Determine whether NRAS is hosted by IIS or is running as a standalone service, RSNRAS service.
2. If NRAS is hosted by IIS, make sure that NRAS has been initialized inside IIS. To make sure NRAS has been initialized, run an Ops flow that uses NRAS.
3. In Visual Studio (with your IAction project open), from the **Debug** menu, choose **Processes**.
4. In the **Processes** dialog box, select the Asp_net.exe process.
5. Click **Attach**.
6. In the **Attach to Process** dialog box that appears, make sure that **Common Language Runtime** is selected, click **OK**, and then (back in the **Processes** dialog) click **Close**.

7. Set a break-point in the IAction code.
8. Run a flow that references the IAction.
The IAction should stop at the break point.

Ops flow architecture and concepts

The Quick View introduced you to basic Ops flow concepts and information on how to make simple modifications of existing flows. This chapter explores the following concepts in greater depth:

- The order of actions performed in the execution of a step
- Operation architecture and information flow
- Step and operation requirements such as defining data sources for inputs and defining responses
- Ops flows, steps, and operations as PAS objects
- Creating scriptlets that make possible the additional manipulation of an operation's output data.

Scriptlets are JavaScript or Perl scripts contained within an operation. Scriptlets are good for programming quickly, with relatively little customization of the operation.

In addition to writing scripts in JavaScript or Sleep, this level of authoring requires that you understand:

- Input and flow variables
- Distinctions between steps and operations
- Distinctions between results, raw results, and responses
- Data filters

Step execution order

When a step is carried out, the following actions are carried out, in this sequence:

1. Input values are obtained from the collection of flow variables and global data values and applied to the inputs, making them available to the step's operation.
2. Any changes to the input values are applied to the collection of flow variables and global data values.
3. The step's operation is carried out.
For details on how information is obtained by, flows through, and can be modified within an operation, see the following section, [Operations: architecture and information flow](#).
4. If the operation has a scriptlet, the operation's scriptlet is executed.
The operation's scriptlet can do the following:
 - a. Select the operation response.
 - b. Set the operation's primary result.
 - c. Make changes to local and global flow variables and data values.
 - d. Read the operation response value if there's a value present to be read, such as if the operation contains an IAction that uses a RAS. That IAction has a response, which the operation scriptlet can get and read.
5. If the operation's scriptlet has not already set the operation's primary result, the operation's primary result is set now.

6. If the operation's scriptlet has not already selected the operation's response, the response is selected now, using the operation's evaluation rules for selecting a response.
7. If the step has a scriptlet, that step scriptlet is executed.
The step scriptlet can do the following:
 - a. Select the operation response.
 - b. Make changes to local and global flow variables.**Note:** The step scriptlet can not set the primary result.
8. The transition that is associated with selected response is selected.
9. The next step is execute, using this same sequence.

Operations: architecture and information flow

An operation is a defined sequence of actions associated with a step in an Ops flow. When we consider a step that has a flow associated with it as a subflow, we say that the subflow is a type of operation. However, the following description is limited to a single operation.

The parts of an operation are:

- Core functionality (called the *core*), which encapsulates the business logic of the operation
For Web operations, the core functionality is the IAction interface execute method and the method's parameters, which provide data to and influence the behavior of the execute method.
All input values are copied to the local or global context before execution of the operation. Input values can also be bound to the local or global context.
The core might map input onto raw results.
- Further processing of raw results by a scriptlet (optional)
- Determination of a response

The *context* is a container that is independent of the step and holds various values that can be exchanged with the step at various points (see the following diagram). There are two kinds of context: global and local. Local context exists for the duration of the step; global context exists for the duration of the Ops flow. You can pass values to and from the local or global context.

In information Ops flow through an operation, as shown in the following diagram, these parts of an operation play roles:

- Raw results
If the IAction interface is part of the core functionality, the raw results are the data and state.
- Scriptlet
An optional interpretive program that may be executed at the end of a step. The scriptlet often evaluates the raw results of the operation and produces the output data of the operation.
- Output data

The data produced by the operation, if any.

- Response

The evaluation of the operation's output and the resulting determination of the transition from among the possible transitions for the operation's step.

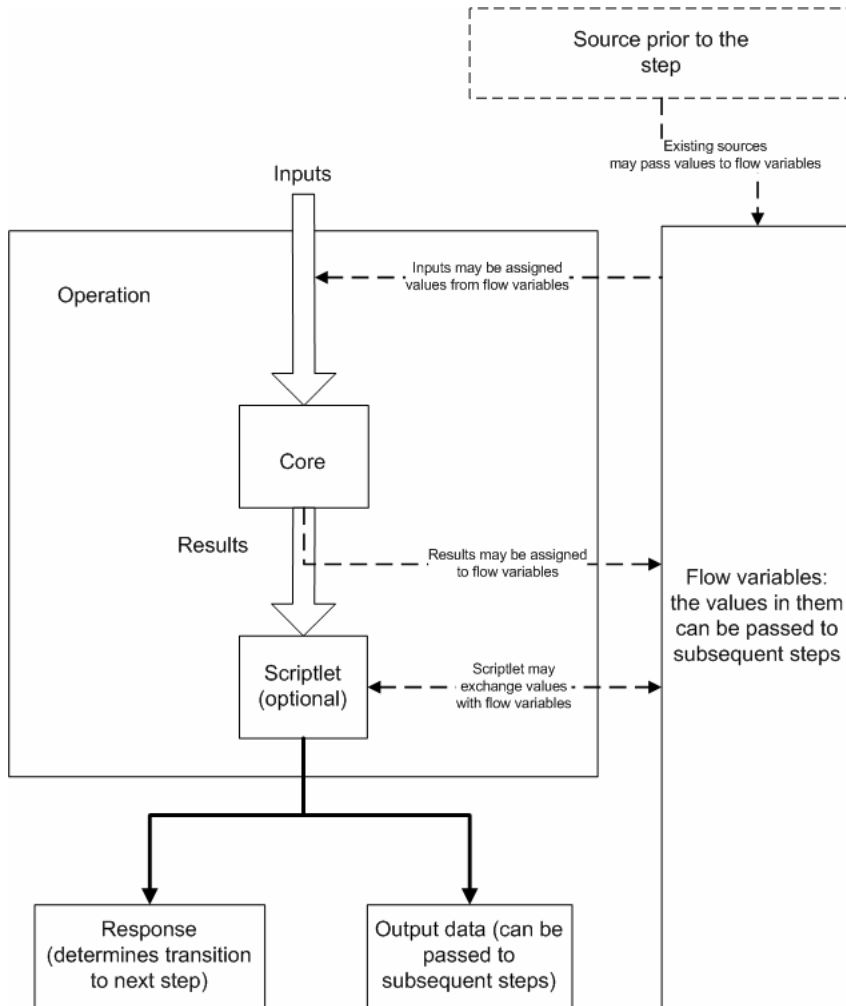


Figure 69 - Information flow through an operation

Note: Operations that are provided in the **Operations** folder are sealed—that is, you cannot modify them other than to supply fixed, specific values for inputs.

Advanced flow, step, and operation concepts

The following concepts are important to successfully authoring Ops flows.

Local vs. global variables, in operation, step, and flow inputs

You can use local and global variables to move data among steps within a flow.

- Global variables are available to all the components of a flow within its run, including any subflows and parent flows. When you supply a value to a global variable in a subflow, the value is available to any parent flows in that run.
- Local variables are available only for the flow within which they are defined.

Steps vs. operations

An operation is a template from which you create a step. When you drag an operation onto a flow's authoring canvas, you create a step that is associated with the operation. However, there are significant differences between the step and the operation. For instance, you define inputs for the step on the step's **Properties** sheet, not by accessing the operation from the step. If you modify the operation, you are modifying the template that is the basis for all the steps associated with that operation. This means that you are changing all the steps associated with that operation, regardless of which flows the steps are part of. Thus, unless you are careful, making changes to the operation could have unintended consequences of breaking Ops flows that use that operation.

There are also distinctions between scriptlets that are created on the step and on the operation. For instance:

- Operation scriptlets cannot read the value of the response of the operation.
- The operation scriptlet does not appear in the Scriptlet tab of the step created from the operation.
- A step's scriptlet result cannot be passed to the step's result. (You can get around this limitation by performing the task you want in a scriptlet filter for the step's result rather than in the step's scriptlet.)

Flow, step, and operation inputs

Each input is mapped to a variable, whose value can come from a variety of sources.

Which element you add an input to can determine when the input's value is obtained:

- An input for a flow obtains a value before the first step runs.
It is best practice to set any inputs that the Ops flow needs and that are not produced by processing within the flow in the Ops flow's properties, thus making these input values available to the Ops flow before it begins to run.

- An input for a step obtains a value before the step's operation runs.

Inputs also reflect the distinctions between elements:

- An input that you create for a step (on the step's **Properties** sheet) is not an input for the operation associated with the step. Its value is obtained before the operation runs.
- An input that you create for the instance of an operation that is associated with the step is specific to that instance. Changing the input for the instance does not change the input for the operation in the Library. That is, if you right-click a step and choose **Open Operation**, then modify the input, only this instance of the operation is affected, not the operation in the Library.
- When you change an input for the operation in the library (on the Properties sheet that you open by right-clicking the operation in the Library), all instances of the operation that you subsequently create reflect the change that you made.

For information on defining the data source for an input, see [Inputs: Providing data to operations](#).

Outputs

There are several kinds of outputs for flows, steps, and operations: raw results, results, and responses. Steps have results, and flows have results and responses.

- Raw results are a collection of key value pairs representing the raw data that was returned from an operation executed in the context of a flow.

For example, if you execute the ping operation on a windows XP machine, you will get the following results:

```
{
  Code = "0"
  Error String = ""
  Output String =
    "Pinging apple.com [17.254.3.183] with 32 bytes of data:

    Reply from 17.254.3.183: bytes=32 time=24ms TTL=244
    Reply from 17.254.3.183: bytes=32 time=24ms TTL=244
    Reply from 17.254.3.183: bytes=32 time=25ms TTL=244
    Reply from 17.254.3.183: bytes=32 time=26ms TTL=244

    Ping statistics for 17.254.3.183:
      Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
      Minimum = 24ms, Maximum = 26ms, Average = 24ms"
}
```

You can filter the values of the raw results to create a more refined result. For example, if in the above data you are interested only in the average latency of the ping operation, you can select the Output String field of your operation and specify a series of filters which extract the '24ms' token at the end of the string. For more information, see [Filter details](#) or [Filtering an operation's results](#).

- A result is an output string that a step's operation produces. A step or Ops flow can have only one result. The result can be produced by an operation scriptlet or obtained from one of the fields in the raw results. You can use results to annotate strings or send them to the local or global context.
- Responses are the outcomes that are connected, each response by one transition, to a following step. Because you link a response to a subsequent step, the response is the determination of what the flow does next. Return steps are an exception: Return-step responses return an outcome for the entire flow.

The following table shows which output types pertain to operations, steps, and flows, respectively. Note, however, that you cannot change the handling of outputs for sealed operations.

	Raw results	Results	Responses
Operations	✓	✓	✓

Steps		✓	
Flows	✓	✓	✓

Troubleshooting

Following are problems with special solutions.

The script in my operation does not run correctly.

By default, PAS Central expects to find scripts in the PAS home directory, in the \Central\scripts subdirectory. This default location is specified in the Central.properties file.

If the script that the operation uses does not reside in the \Central\scripts subdirectory, do one of the following:

1. Move the script to the \scripts subdirectory.
2. In Central.properties, find the following line:
`dharmapp.script.repository=${iconclude.home}/Central/scripts`
3. Append to the line a semicolon separator and the location of any scripts that you want to use.

For example, to execute scripts residing in the c:\MyScripts directory, you would modify the line to read as follows:

```
dharmapp.script.repository=C:\Program
Files\iconclude\Central\scripts;c:\MyScripts
```