HP Operations Orchestration

For the Windows ® and Linux operating systems

Software Version: 9.04

Concepts Guide

Document Release Date: April 2012

Software Release Date: April 2012

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notice

Trademark Notices

REQUIRED: Do not change the name of your copy of the `_HPc_TradeMarksSnip.flsnp` file. Copy this file to your `<FlareProject>/Content/Resources/Snippets/Required/_HPc_TradeMarksSnip.flsnp`.

Adding text to this section is optional. Include only relevant trademark notices that are listed on the HP Legal “Acknowledged List of Trademarks” web page: http://legal.hp.com/legal/pages/tradeack.aspx

If your HP product team receives instructions *to acknowledge trademarks that are required by signed contract or license agreement*, the product team must share that information with you. Include the appropriate symbols: ® ™ See the Information Engineering web site for more information.

If your HP product *does not require any Trademark Notices*, type "None" into a paragraph below and delete the other two paragraphs.

Adobe™ is a trademark of Adobe Systems Incorporated.

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

UNIX® is a registered trademark of The Open Group.

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

REQUIRED: Do not change the name of your copy of the `_HPc_AcknowledgementsSnip.flsnp` file. Copy this file to your `<FlareProject>/Content/Resources/Snippets/Required/_HPc_AcknowledgementsSnip.flsnp`.

Adding text to this section is optional. If the product team receives permission from the Legal Department to acknowledge third-party software used in the product, they must let you know of that permission.

If the product team receives permission to publish the *full* third-party licenses, they must give you the text, and *you must create a separate document for it*. If you are asked to publish the licenses for the open source and third-party software used in your product, use a template from the Information Engineering web site (Word, html, or Flare). See the templates for more instructions about publishing such licenses.

If your product team receives instructions *to provide Acknowledgements*:

If your HP product *does not require any Acknowledgements*, leave the heading and text in this file set to `_HP_Conditions.Internal`.

Documentation Updates

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

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

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

http://h20230.www2.hp.com/selfsolve/manuals

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

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

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

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

REQUIRED: Do not change the name of your copy of the `_HPc_DocUpdatesTableSnip.flsnp` file. Copy this file to your `<FlareProject>/Content/Resources/Snippets/Required/_HPc_DocUpdatesTableSnip.flsnp`.

Adding this table is optional. This `_HPc_DocUpdatesTableSnip.flsnp` file is still required in your Flare project, but the snippet can remain un-modified set to `_HP_Conditions.Internal`.

If your HP product *needs this table*, remove the `_HP_Conditions.Internal` from the [p] and [table] HTML element below (display Flare's XML Editor Blocks, then select the block for each HTML element one-at-a-time: right-click, select Conditions, and then de-select "Internal".

The following table indicates changes made to this document since the last released edition.

Support

Visit the HP Software Support Online web site at:

http://www.hp.com/go/hpsoftwaresupport

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

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

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

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

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

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

Chapter 1

Introduction

This Concepts Guide introduces you to the basic components and ideas of HP Operations Orchestration (HP OO).

This chapter explains:

- "What is HP Operations Orchestration?" below

What is HP Operations Orchestration?

HP Operations Orchestration (**HP OO**) is a system for creating and using actions in structured sequences (called Ops flows, or flows) which maintain, troubleshoot, repair, and provision your Information Technology (IT) resources by:

- Checking the health of, diagnosing, and repairing, networks, servers, services, software applications, and individual workstations.
- Checking client, server, and virtual machines for needed software and updates, and, if needed, performing the necessary installations, updates, and distributions.
- Performing repetitive tasks, such as checking status on internal or external web site pages.

The two main components of HP OO are **Central** and **Studio**.

HP OO Central

This is a web based interface in which you can:

- Run flows
- Administer the system
- Extract and analyze data resulting from the flow runs

HP OO Studio

This is a standalone authoring program in which you can:

- Create, modify, and test flows, including flows that run automatically, as scheduled.
- Create new operations.

  You can create operations within Studio and run them in Central.

  You can also create operations that execute outside Central, in a remote action service (RAS). You do so in a development environment that is appropriate to the task, then associate the code you have created with an operation that you create in Studio.
- Specify which levels of users are allowed to run various parts of flows.

Chapter 2

HP Operations Orchestration Concepts

This chapter explains:

Ops Flows and Flow Runs

Ops flows, or flows, are logically linked sequences of actions (steps), associated with operations. Do not confuse **flows** with **flow runs**. A flow run is a single execution of an Ops flow in Central.

A flow run can perform any task that you can program, on any computer anywhere on an intranet, extranet, or the Internet. For example, you might have flow runs that:

- Monitor the health of a program, a system, hardware, or a complex of systems
- If a problem is found, create an alert or a tracking ticket, and fix the problem
- Provide an audit trail for troubleshooting
- Gather information for IT system analysis

Flows are stored in the HP OO repository.

Flows are a type of operation, because a flow can be the basis of a step in another flow.

- A **subflow** is a flow that is used as a step in another flow.
- The flow that contains the subflow step is the **parent** flow.

A flow run can be interrupted, resumed, or handed off to another Central user. The HP OO administrator manages flow runs. Flow runs collect data, enabling the administrator, managers, and executives, to analyze performance of their IT system.

Repositories

A repository is a hierarchical structure of XML files that constitute a set of flows, operations, remote action service references, IActions (if any are used), and system accounts, and other configurable objects such as domain terms, scriptlets, selection lists, system evaluators, system filters, and system properties.

There are two kinds of repositories:

- **Public repository**

  This is the repository associated with an installation of OO Central. In most systems, there is an installation of Central for development and testing purposes, and another for the production (real world) environment. With this configuration, once a flow has been developed and tested in the development installation, it can be published to the production environment.

  One of the keys to how an author works is whether he is working in direct connection with the public repository. If the public repository is added to and open in the author's Studio as he or she works, then the author is working online, making changes directly in the public repository. With correct system configuration, multiple authors can work online in a single public repository.

  The checkin/checkout feature in the public repository enables multiple authors to work in the public repository at the same time. Checkin/checkout effects version control, enabling authors to work on the same flow without conflicts between contesting changes.

  Checkin/checkout requires developers to check out a flow, operation, or other object, before working on it. When an author checks out an object, he works on the object in his workspace of the public repository. To convey the changes to the repository and be visible to other authors, the author must check the flow back in.

  For more information on the checkin/checkout feature, see the Studio Authoring Guide (Studio_AuthorsGuide.pdf).

- **Private repository**

  This is one of the repositories associated with an installation of OO Studio. A private repository is not directly connected to Central; an author working in a private repository is said to be working offline. To move work from a private repository to a public repository, the author publishes ***from*** the private repository ***to*** the public repository. Conversely, to get Library objects from the public repository, the author updates the private repository from the public repository.

  There can be more than one private repository for your installation of Studio.

Publishing, Updating, and Promoting a Repository

Publishing from one repository to another merges the changes in the source repository into the target repository. A preview panel shows which flows, operations, and library objects in the source repository (such as system accounts and domain terms) are new or have been changed, and you can select which of the changes or additions you want to apply.

Updating is the reverse of publishing: new and changed flows, operations, and library objects from the target repository are merged into the source repository.

Note: The source and target repositories do not change according to whether you are publishing or updating. In OO Studio, once you have identified a target repository, it remains the target repository until you identify a different one, regardless of whether you publish or update in the meantime.

Promotion is publishing specifically from a one public repository to another. Remember that each installation of OO has a single public repository. To promote a repository, you must:

- Have READ access privileges on the public repository being promoted; **and**
- Be a member of either the ADMINISTRATOR or the PROMOTER group on the target OO installation

In a configuration in which there is an installation of OO in a development environment, and another in a production environment (possibly with a testing or a staging environment between), the restriction of the ability to publish to the public repository in the production environment protects the repository whose flows are actually used.

Workspaces

A workspace is a view.

Snapshot

A repository snapshot is a view of the state of a public repository's checked-in work.

Library

The HP OO Library (**Library**) is a directory structure in Studio and Central.

- In **Central**, the Library shows a repository's collection of flows.
- In **Studio**, the Library displays the entire repository, including flows, operations, remote action services, system accounts, and other configurable objects.

Concurrent Execution

Concurrent execution, also called **parallel execution**, means running entire runs of a flow simultaneously, or designing steps within a flow to run simultaneously with each other. For example:

- To diagnose a problem, you need to run health checks against a database server, an application server, a web server, and a router. Your diagnosis is quicker if you are able to run all the checks simultaneously.
- You have a software upgrade to install on 100 servers. The process is much quicker if you are able to install the upgrade on all (or several) servers simultaneously, instead of one at a time.
- One of the steps in a diagnostic flow creates a trouble ticket, or creates and sends an e-mail. Resolution is quicker if the flow continues running with subsequent steps while that one step creates the ticket or e-mail.

HP OO provides multiple ways to achieve such parallel execution.

- To run **entire flows** in parallel, you can do either of the following:
  - On the **Schedule** tab in Central, schedule simultaneous runs of a flow
  - Specify parallel automatic runs of a flow by a URL

Within a flow, you can create three kinds of parallel execution:

- **Parallel split step**, in which one step contains several series of steps that execute simultaneously.

  Parallel split steps are intended for performing dissimilar and separate series of steps at once. Each series of steps is called and represented visually in the flow diagram as a **lane**. The steps contained in each lane are called **lane steps**. The lane's series of lane steps are like a subflow, but in several respects, including movement of data into and out of them, they are very different from a subflow or flow.

  In the scenario in which you want to run several different kinds of health checks simultaneously, you could create a parallel split step with four lanes. In one lane, you create the lane steps necessary to run a health check on the database server; in the next the steps necessary to run a health check on the application server, and so forth.

- **Multi-instance step**, which executes simultaneously with multiple members of a list provided for a value in an input of the step's operation. You can **throttle** a multi-instance step, which means limiting how many values it can process at once.

  Using the example of upgrading many servers at once, assume you have a step associated with the subflow that performs the upgrade, and this step takes the target input specifying which server the subflow runs against. You could turn this step into a multi-instance step, and supply it with a list of the servers you want the subflow to upgrade. The step then runs the check against all those servers simultaneously.

  If running the subflow against too many target servers simultaneously bogs down your infrastructure, you could throttle the multi-instance step by specifying that it would only process, for example, 25 target inputs at once.

- **Nonblocking step**, which allows the flow to continue with subsequent steps while the nonblocking step is still executing.

  Using the example of a step in a diagnostic flow that creates a trouble ticket, you could make that step a nonblocking step, allowing the flow to proceed while the step creates the ticket.

Chapter 3

Ops Flow

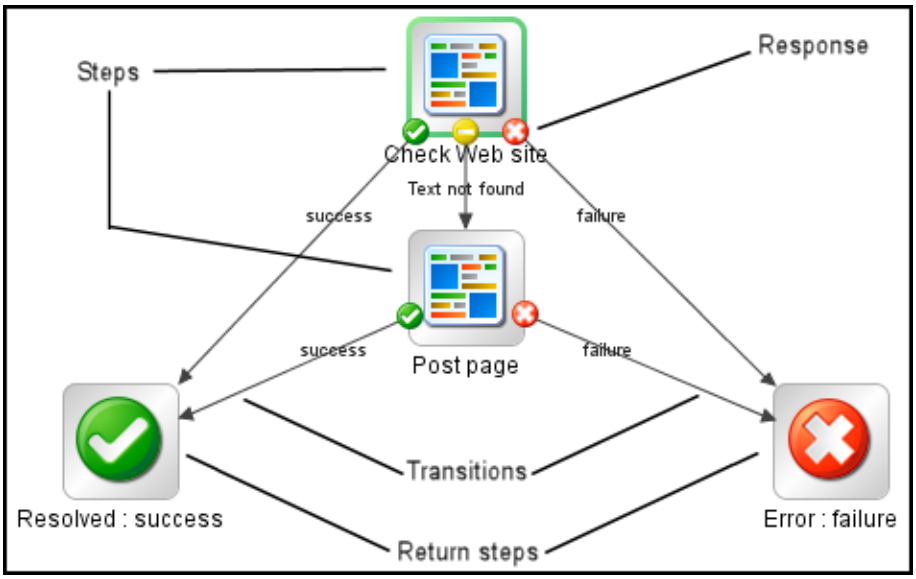
This chapter explains:

Overview

Steps are the basic units of an Ops flow. Each step is created from an operation.

- An Ops flow step's operation uses input data to perform a task. From the performance of the task and optional subsequent processing, the operation obtains results.
- The operation has several possible responses, with the chosen response being determined by the operation's results.
- Each response is connected by a transition to one of the possible next steps in the flow.

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



Ops Flow Details

This section gives details about the parts of an Ops flow.

Operations

Operations do the work of the flow. Each operation is the template for the steps created from it. For example, a step might be created from an operation that checks a web page to see whether it contains specific text. Another step in the same flow might be an instance of an operation that copies a file. You could use these two steps in combination to update a web page.

An operation is an action and, optionally, subsequent manipulation of the data that the action produces. An Ops flow is, technically, an operation, so a step can be created from an Ops flow as well. This means that an Ops flow can contain another flow as a subflow.

The following are examples of tasks that an operation can accomplish:

- Checking the status of a service on a computer
- Placing or retrieving a file with the ftp **put** or **get** commands
- Simultaneously launching an installation program on a list of computers
- Querying a URL for its availability (using an **HTTPGet** operation)
- Running an SQL query against a particular database table (using an **SQL Query** operation)

Inputs

Inputs give the operation the data that it needs to act upon. For example, an operation to check a web page needs to know which page to check and what text to look for. For another example, a copy file operation needs a source location and a destination.

Inputs define the means by which operations obtain the data they need to do their work.

Each input is mapped to a variable, whose value can be set in any of several ways:

- In a user prompt, the value is entered by the person running the Ops flow.
- The flow author can set the input's value to a specific, unchanging value.
- The value can be obtained from another step.
- One of a collection of variables (called **flow variables**) and data values that are available to the entire flow, can be assigned to the input.

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.

  When possible, 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. This makes 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. The operation processes a step input's value only during the execution of that step. In another step that is based on the same operation, the operation processes the second step's value for the input of the same name.
- When you change an input for the operation in the Library, all instances of the operation you subsequently create reflect the change you made.

Outputs, Responses, Step Results and Transitions

Outputs, responses, step results and transitions are related to each other in flow authoring.

Operation and Flow Outputs

Outputs (previously called results) are all or part of the output of an operation.

Remember that a flow is a kind of operation. This is particularly apparent when a step in a flow is associated with another flow. Therefore, flows also have outputs.

The raw output is all of the operation's return code, data output, and error string.

For example, if you execute the ping operation on a windows XP machine, 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"
}
```

The primary and other outputs are portions of the raw output (for example, success code, output string, error string, or failure message) that you define as the contents of one of the operation's fields. (You can narrow a primary or other output to a more highly focused selection, by creating one or more filters for the output.)

Responses

Responses are the possible outcomes of operations.

For example, a **copy file** operation might have **success** and **failure** for its responses.

For another example, a **get web page** operation has these responses:

- page not found
- text not found
- success (the page is found, and it has the desired text).

Rules that evaluate the operation's output fields determine which of the several available responses are the operation's actual response for the current run. In a step created from an operation, each response is the starting point from which a transition may be connected to another step (or back to the step from which the transition started). Therefore, the outcome of the evaluation of one step's operation outcome determines which is the next step in the flow run: what the flow does next.

Return steps are an exception: return-step responses return an outcome for the entire flow, and are the last step in any given run.

Step Results

Step results are the step's analog to the outputs of the operation from which the step was created. You can pass step results as data to other steps in the flow, or to other flows. You can create filters to extract and modify parts of the operation's output.

For example, assume 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 by filtering them 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. This makes the properties available to operations, steps, and transitions in the parent flow.

  You can pass the step result's value to either:

  - A local flow variable
  - A flow output field for the flow.

Transitions

A **get web page** operation may need to respond differently if the web page cannot be found, if the page is there but the text is not present, or if the page is there and the 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 is.

Flow Variables

Flow variables are variables that store data obtained by operations or their scriptlets, for use in other flows, or other steps within a flow.

- **Global flow variables** are available for all flows (both parent flows and subflows) within the current run. When a step in a subflow stores a value in a global flow variable, the variable's value becomes available to the parent flow and any other subflows.
- **Local flow variables** are available only to the current flow, whether it is a parent flow or subflow.

Context

Context is a key concept for controlling how data moves in and out of operations. The context is a container that holds various values that can be exchanged with a step at various points.

The types of context are: **local** and **global**.

- Local context exists for the duration of the step.
- Global context exists for the duration of the flow.

You can pass values to and from the local or global context.

Chapter 4

Steps and Operations

This chapter explains:

Step Execution

When a step executes, the following sequence of actions occurs:

1. Input values are obtained from the data sources that have been defined for the inputs. These values are made available to the step's operation.
2. 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 preceding section.
3. If the operation has a scriptlet, the operation's scriptlet executes.

   The operation's scriptlet can do the following:

   a. Select the operation response.

   b. Set the operation's primary result.

      The primary result is the result that supplies a value to an input whose assignment is **Previous Step's Result**.

   c. Make changes to local and global flow variables.
4. If the operation's scriptlet has not already set the operation's primary result, the operation's primary result is set.
5. If the operation's scriptlet has not already selected the operation's response, the response is selected, using the operation's evaluation rules for selecting a response.
6. The step results are extracted.
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 cannot set the step's primary result.

8. The transition associated with the selected response is selected.
9. The next step is executed, using this sequence.

Comparison of Steps and Operations

Steps are the building blocks of Ops flows. Each step in a flow is created from an operation; the operation which the step is an instance of.

Because the step is an instance of the operation:

- The step inherits the inputs, flow variables, and properties of the operation.
- Changing a step's input or local variable changes the input for the operation or subflow when that step runs. The operation itself remains unchanged. Therefore, you can specify different inputs in different steps, created from a single operation.
- Changing the operation, modifies the template that is the basis for all the steps associated with that operation. The steps created from that operation are either changed also, or broken, depending on what is changed. In turn, all the flows that contain those steps may be broken due to the change in the operation.

Operations: the Models 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 command, including command-line, 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's results is 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). A **get web 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 may be wanted for use later on in the flow. For example, using a single command, an operation to get memory statistics on Linux tells how much memory is free, how much total memory is available, how much swap memory is being used, and much more.
- **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:** These evaluate the operation's results to determine which operation response to take when the step runs. Rules can evaluate any of the results fields, data strings, return codes, or error codes.

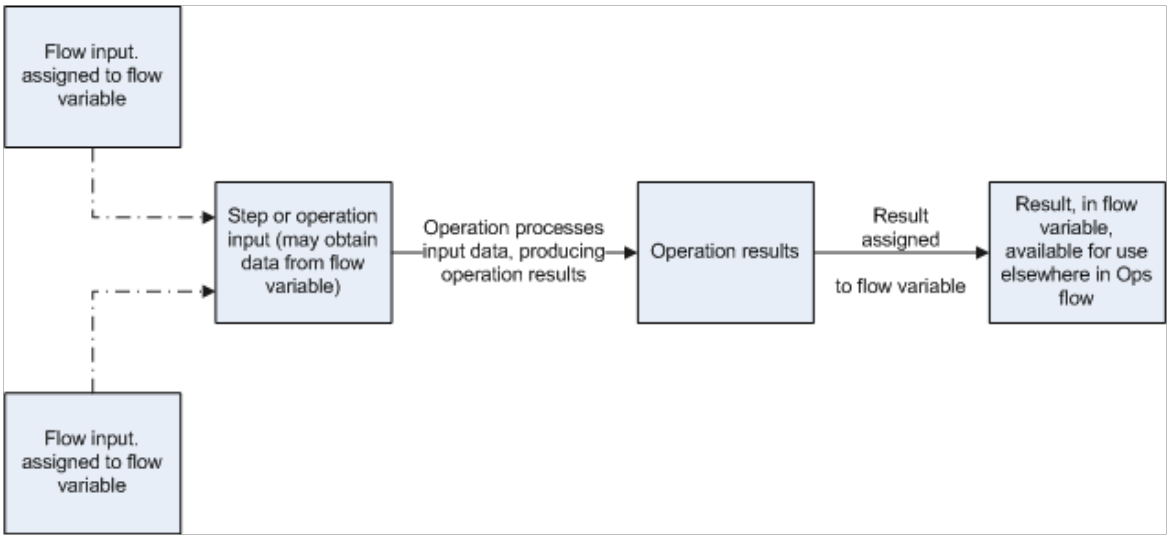
  A given response for an operation is selected when the conditions described in the response's rule match the data specified in or extracted from the one of the results fields. The rules you create are comparisons or matches between a string of text described in the rule and the results field selected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

  Consider the **get web page** operation example:

  - The **page not found** response is selected if the http return code is 404.
  - The **text not found** response is selected if the text to check is not in the data on the page.
- **Scriptlets:** These optional parts of an operation (written in JavaScript or Perl) manipulate data from either the operation's inputs or results, for use in other parts of the operation or flow.

The following diagram shows:

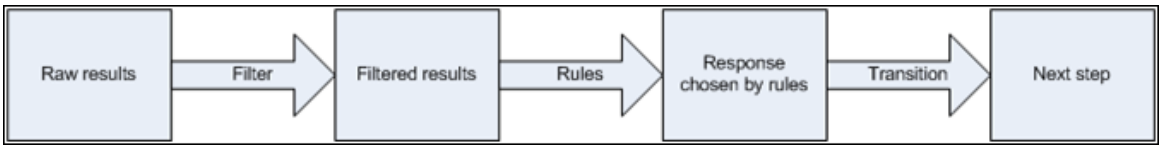
- How operations can get values from flow, step, and their own operation inputs.
- How data in operation results can be passed to flow variables, becoming available to other parts of the Ops flow.



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

1. Flow inputs could provide two needed pieces of data: the host computer and the name of the directory to run the dir command against
2. These two input data items could be stored in flow variables named host and directory, respectively.
3. The operation inputs could obtain the values from the flow variables.
4. 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.

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 contexts: 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 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**

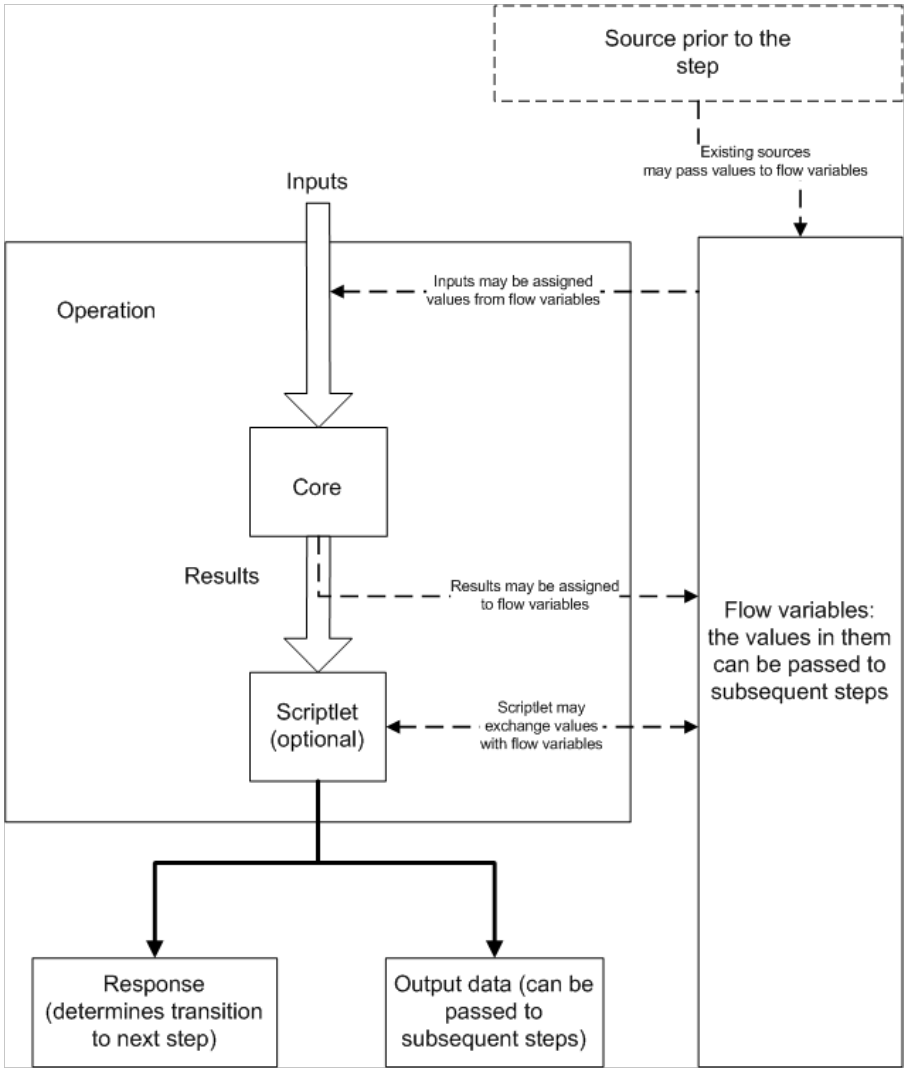
  An optional interpretive program that may be executed at the end of a step. The scriptlet often evaluates the raw results of the operation and produces the output data of the operation.
- **Output data**

  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.

Note: Operations provided in the **Operations** folder are sealed; other than supplying fixed, specific values for inputs, they may not be modified.

The following diagram shows information flow through an operation.



Chapter 5

OO Architecture and Administration

HP OO includes the following components:

- Central

  Central is the Web-based application in which you execute flows and review the information that the Ops flows have extracted on the health of the IT resources against which the flows have run.

- Studio

  Studio is the standalone application in which you create Ops flows or import them (in **Accelerator Packs**).

- A database, which stores data related to results from running flows.

- HP OO content, which comprises the flows and operations that are available with initial installation of HP OO. You configure these flows and operations and use them by themselves, or to build more flows. They are organized into the following folders:

  - **Accelerator Packs**

    These flows are designed to provide the following on most networks:

    - Complex health checks, triage, diagnosis, or remediation flows.
    - Simple flows that gather one or more pieces of data and display it to the user, or simply acknowledge alerts, gather some data, and place it into a ticket.

    The flows at the top level of an Accelerator Pack tend to be full health checks, triage, diagnosis, and remediation.

  - **Integrations**

    Operations and flows that can be used from OO to interact with third-party systems-management products.

  - **ITIL**

    This folder contains flows that automate integrations to other Enterprise level software in accordance with ITIL specifications.

  - **Operations**

    This folder contains general-purpose operations that work with common technologies. These operations are sealed and cannot be changed once you have installed HP OO Central.

    Because you cannot change operations in the **Operations** folder, they should not have any static values set for inputs. All inputs should either prompt the user, or be **No Assignment**. There are exceptions to this rule, for example when a very general purpose operation such as a WMI command is used.

The flows in the **Operations** folder and subfolders are meant to work as subflows or operations. Flows that you want to run as the parent flow are in the **Accelerator Packs** folder.

- **Utility Operations**

  This folder contains operations and subflows that gather and display data, replace simple command-line operations, manipulate and analyze data, provide structure to flows, and perform other non-technology-specific functions.

- Remote action services

  Remote action services (RAS), which are services installed with HP OO, are the means by which HP OO operations are carried out outside HP OO. Running outside HP OO means that a flow is either:

  - Running remotely or automatically, or
  - Interacting with platforms, environments, or applications that are not directly accessible from HP OO.

  Once you have installed an RAS, then in Studio you can configure a reference to the RAS. Operations that use the RAS to run remotely, access the RAS through the reference you create in Studio.

Chapter 6

How to Find Information about Flows and Operations

There are two ways in which you can find detailed information about each flow and operation in the OO Library:

- Reading the flow or operation **Description** tab
- Using the **Generate Documentation** feature

Reading the Flow or Operation Description Tab

- For a flow, click the **Properties** tab and then click the **Description** tab. For an operation, just click the **Description** tab.

  The Description tab contains the following information:

  - A description of what the flow or operation does.
  - **Inputs** that the flow or operation requires, including where users and authors can find the data that the inputs require and the required format for the data.
  - **Responses**, including the meaning of each response.
  - **Result fields**, including a description of the data supplied in each result field.
  - Any additional implementation **notes**, such as:
    - Supported platforms or applications, including version information.
    - Application or Web service APIs that the flow or operation interacts with. (This can be particularly important for operations that require an RAS to run, because the RAS operation can hide this information from the author or user of the flow.)
  - Other environmental or usage requirements.

Generate Documentation Feature

This feature pulls the information from the **Description** tab of the flows and operations in the folder you select, and presents it in a coherent, linked list of HTML files.

For more information on the **Generate Documentation** feature, see the Studio Authoring Guide (Studio_AuthorsGuide.pdf).


Java™
COMPATIBLE