HP Operations Orchestration Software

Software Version: 7.51

Concepts Guide

Document Release Date: August 2009

Software Release Date: August 2009

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 Notices

Trademark Notices

All marks mentioned in this document are the property of their respective owners.

Finding or updating documentation on the Web

Documentation enhancements are a continual project at Hewlett-Packard Software. You can obtain or update the HP OO documentation set and tutorials at any time from the HP Software Product Manuals web site. You will need an HP Passport to log in to the web site.

To obtain HP OO documentation and tutorials

1. Go to the HP Software Product Manuals web site (*http://support.openview.hp.com/selfsolve/manuals*).
2. Log in with your HP Passport user name and password.

   OR

   If you do not have an HP Passport, click **New users – please register** to create an HP Passport, then return to this page and log in.

   If you need help getting an HP Passport, see your HP OO contact.
3. In the **Product** list box, scroll down to and select **Operations Orchestration**.
4. In the **Product Version** list, click the version of the manuals that you're interested in.
5. In the **Operating System** list, click the relevant operating system.
6. Click the **Search** button.
7. In the **Results** list, click the link for the file that you want.

Where to find Help, tutorials, and more

The HP Operations Orchestration software (HP OO) documentation set is made up of the following:

- Help for Central

  Central Help provides information to the following:

  - Finding and running flows
  - For HP OO administrators, configuring the functioning of HP OO
  - Generating and viewing the information available from the outcomes of flow runs

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

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

  The Studio Help system is also available as a PDF document in the HP OO home directory, in the \Studio\docs subdirectory.
- Animated tutorials for Central and Studio

  HP OO tutorials can each be completed in less than half an hour and provide basic instruction on the following:

  - In Central, finding, running, and viewing information from flows
  - In Studio, modifying flows

  The tutorials are available in the Central and Studio subdirectories of the HP OO home directory.
- Self-documentation for operations and flows in the Accelerator Packs and ITIL folders

Self-documentation is available in the descriptions of the operations and steps that are included in the flows.

Support

For support information, including patches, troubleshooting aids, support contract management, product manuals and more, visit the following site: *http://www.hp.com/go/bsaessentialsnetwork*

This is the **BSA Essentials Network** Web page. To sign in:

1. Click **Login Now**.
2. On the **HP Passport sign-in** page, enter your HP Passport user ID and password and then click **Sign-in**.
3. If you do not already have an HP Passport account, do the following:
   a. On the **HP Passport sign-in** page, click **New user registration**.
   b. On the **HP Passport new user registration** page, enter the required information and then click **Continue**.
   c. On the confirmation page that opens, check your information and then click **Register**.
   d. On the **Terms of Service** page, read the Terms of use and legal restrictions, select the **Agree** button, and then click **Submit**.
4. On the **BSA Essentials Network** page, click **Operations Orchestration Community.**

   The Operations Orchestration Community page contains links to announcements, discussions, downloads, documentation, help, and support.

Note: Contact your OO contact if you have any difficulties with this process.

Table of Contents

This guide

This Concepts Guide is intended to introduce you to the basic components and ideas of HP Operations Orchestration.

It is organized into the following sections:

- What HP Operations Orchestration is
- Product concepts
- What you can do with HP OO
- HP OO architecture and administration

What HP Operations Orchestration is

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

- Checking the health of, diagnosing, and repairing networks, servers, services, software applications, or individual workstations.
- Checking client, server, and virtual machines for needed software and updates and, if needed, performing the needed installations, updates, or 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 is a Web-based interface in which users can:

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

HP OO Studio is a standalone authoring program in which flow authors 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 within 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 that you have created with an operation that you create in Studio.
- Specify which levels of users are allowed to run various parts of flows.

HP OO concepts

The following sections describe some important HP OO concepts:

- Ops flows and flow runs
- Repositories
- The OO Library
- Concurrent execution

Ops flows

Ops flows are logically linked sequences of steps that are associated with operations. An Ops flow can perform any task that you can program, on any computer anywhere on an intranet, extranet, or the Internet. For example you might have flows that:

- Check to see whether a service is running, and if not, restart it.
- Triage a Web application to determine which tier is causing performance problems.
- Find errors in configuration files and modify them to correct values.

Ops flows are stored in the HP OO Library.

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.**

Flow runs

A run is a single execution of an Ops flow in Central. A run can be interrupted, resumed, or handed off to another Central user. The HP OO administrator manages flow runs. Flow runs collect data that enables the administrator, managers, and executives, to analyze performance of their IT system.

You can view information that is collected from all of a flow's runs, or from all the runs of more than one Ops flow. Collecting such data provides the basis for a higher-level analysis of the state of the IT system and its components.

Repositories

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

There are two kinds of repositories:

- Public repository

  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 environment, the 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 or she 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 *Guide to Authoring Operations Orchestration Flows* (Studio_AuthorsGuide.pdf).

- Private repository

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.

OO Library

The 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: Running several threads at the same time

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, Let's look at what this can mean for you in a few scenarios:

- To diagnose a problem, you need to run health checks against a database server, an application server, a Web server, and a router. You could find your diagnosis much more quickly if you could run the four health checks simultaneously.
- You have a software upgrade to install on 100 servers. If you could install it on 25 servers simultaneously, you could upgrade all the servers in one twenty-fifth of the time it would take to upgrade them all one at a time.
- One of the steps in a diagnostic flow creates a trouble ticket or creates and sends an e-mail. It would speed up resolution if the flow could keep running with subsequent steps while the one step is creating 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 much like a subflow and you might think of them as such, but in several respects, including movement of data into and out of them, they are very different from a subflow or flow. For more information on creating parallel split steps and on how they work, see Help for Studio.

  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 would 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.

  In the example of upgrading many servers at once, let's assume that you have a step that is associated with the subflow that performs the upgrade and that this step takes the target input

that specifies 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 would then run the check against all those servers simultaneously.

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

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

  In the case of a step that opens a trouble ticket, you could make the step a nonblocking step, thus allowing the flow to proceed while the step creates the ticket.

Anatomy of an Ops flow

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, 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 the next step is in a particular run of the flow.

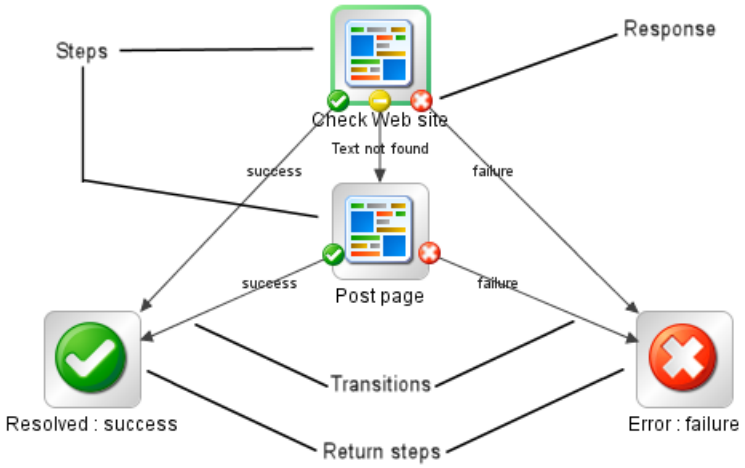


Figure 1 - Parts of a flow

Now let's look at the flow in more detail.

Parts of an Ops flow, in detail

Operations

Operations do the work of the flow. Each operation is the template for the steps that are 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:

- Check the status of a service on a computer.
- Place or retrieve a file with the ftp **put** or **get** commands.
- Launch an installing program on a list of computers simultaneously.
- Query a URL for its availability (using an **HTTPGet** operation).
- Run a SQL query against a particular database table (using a **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 (mysite.com/mypage.htm) and what text to look for ("needed text"). And 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, 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. 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 that you subsequently create reflect the change that you made.

Outputs, responses, and step results

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

Operation and flow outputs

Outputs (which used to be also 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. Thus 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, 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”
}
```

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 **get Web page** operation has three responses: **page not found**, **text not found**, and **success** (if we found the page and it has the desired text). A **copy file** operation might have just **success** and **failure** for its responses.

Rules that evaluate the operation’s output fields determine which of the several available responses will be 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). Thus the outcome of the evaluation of one step’s operation outcome determines which will be the next step in the flow run—that is, 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, 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 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, thus making the properties available to operations, steps, and transitions in the parent flow.

  For more information on filtering outputs, see Help for Studio.

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

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

Transitions

Our **get 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 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.

Checkpoints

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.

A little deeper dive on steps and operations

Now that you're familiar with the basic parts of an Ops flow, let's look at Ops flow steps and operations in more detail.

What happens when a step is executed

When a step is carried out, the following sequence of actions is executed:

5. Input values are obtained from the data sources that have been defined for the inputs and are made available to the step's operation.
6. 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.

7. 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.

      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.
8. If the operation's scriptlet has not already set the operation's primary result, the operation's primary result is set now.
9. 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.
10. The step results are extracted.
11. 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.
12. The transition that is associated with the selected response is selected.
13. The next step is executed, using this same sequence.

Steps and operations, compared

Steps are the building blocks of Ops flows. Each step in a flow is created from an 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. Thus you can specify different inputs in different steps created from a single operation.
- If you change the operation, you are modifying the template that is the basis for all the steps associated with that operation. The steps that have been created from that operation are either changed also or broken, depending on what you have changed. In turn, all the flows that contain those steps may be broken due to the operation's having changed.

For more information on what the relationship between steps and operations means when you are creating an Ops flow, see Help for Studio in the section on Ops flow architecture and concepts.

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 commands, 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). Our **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 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.

- **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, data strings, return codes, or error codes.

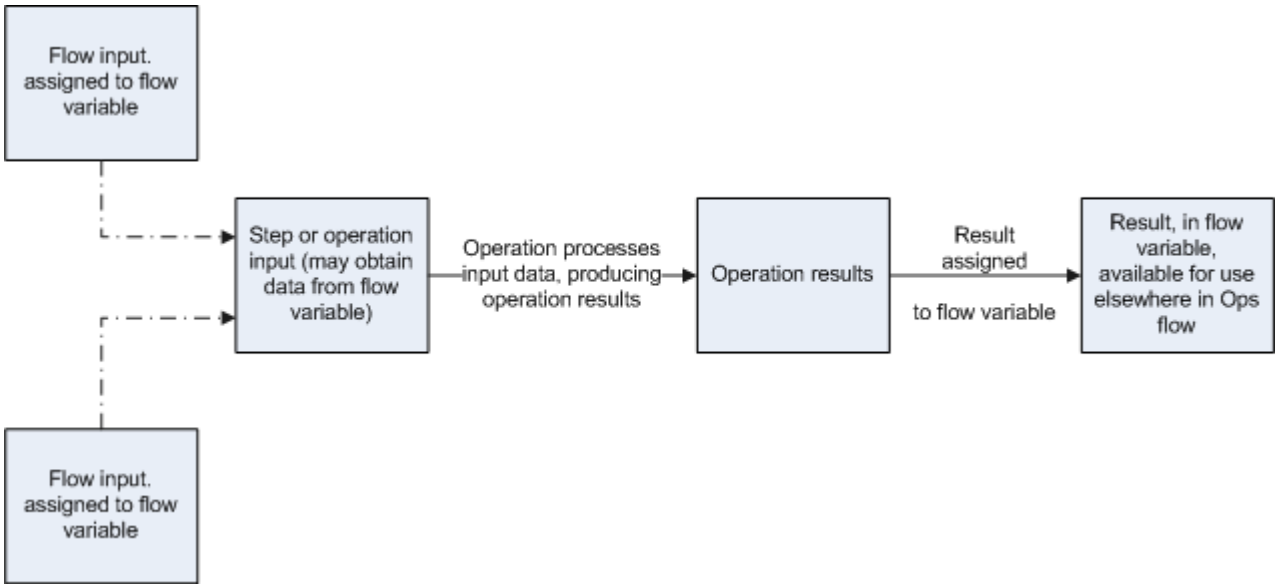
  A given response for an operation is selected when the conditions described in the response's rule match the data that you have specified in or extracted from the one of the results fields. The rules that you create are comparisons or matches between a string of text that you describe in the rule and 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.
- How data in operation results can be passed to flow variables, thus becoming available to other parts of the Ops flow.

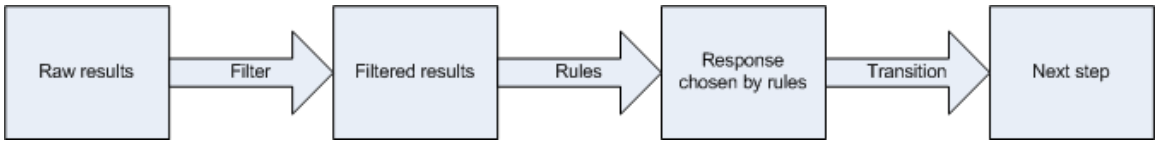


Apply this diagram to a flow that runs the Windows command-line **dir** command on a directory:

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 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 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.

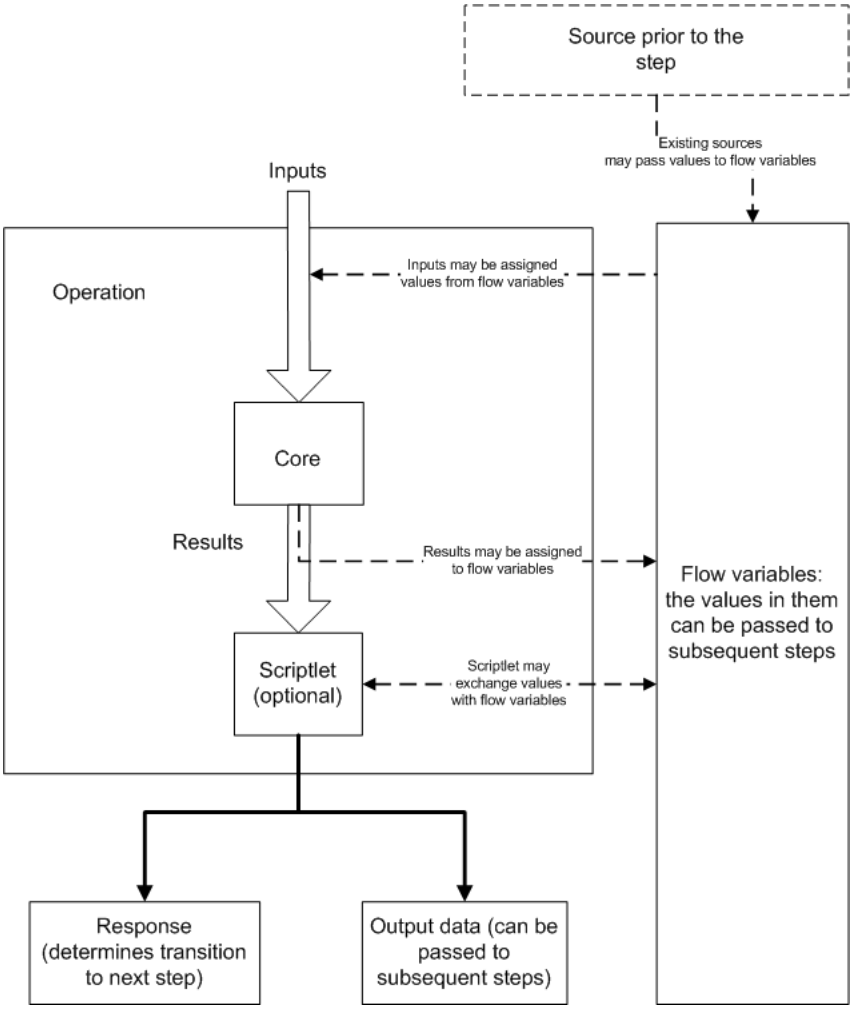


Figure 2 - Information flow through an operation

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.

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 (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, such as 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 would 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 (RASes), which are services that are installed with HP OO, are the means by which HP OO operations that are carried out outside HP OO. Running outside of 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 a 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 that you create in Studio.

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:

- By reading the flow or operation's **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 a RAS to run, because the RAS operation can hide this information from the author or user of the flow.
  - Other environmental or usage requirements.
- By using the **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 *Guide to Authoring Operations Orchestration Flows* (Studio_AuthorsGuide.pdf).

Index