# Contents



# HP Operations Orchestration Studio: Advanced Authoring

Student Guide

Welcome to HP Operations Orchestration Advanced Authoring training. In this course, you will learn many advanced flow development techniques that will prepare you for working on a variety of Operations Orchestration deployments.

The version of HP Operations Orchestration used in training is 9.00.

## Prerequisites

To be successful in this course you should have completed the following training or have equivalent experience with HP Operations Orchestration:

1. Getting Started With HP Operations Orchestration
2. HP Operations Orchestration Administration Basics
3. HP Operations Orchestration Authoring Basics
4. HP Operations Orchestration Intermediate Authoring

## Objectives

By the end of this lesson you will be able to:

- Use Parallel Processing to increase the efficiency of your flows
- Use responses, rules, and transitions to control flow execution
- Use XML Processing content and filters to extract information from XML documents
- Use Remote Action Service to extend HP OO to remote networks
- Work With File Systems
- Work With Email

## Exercises

Exercise 1: Parallel Processing

Exercise 2: Using Responses, Rules, and Transitions Control Flow Execution

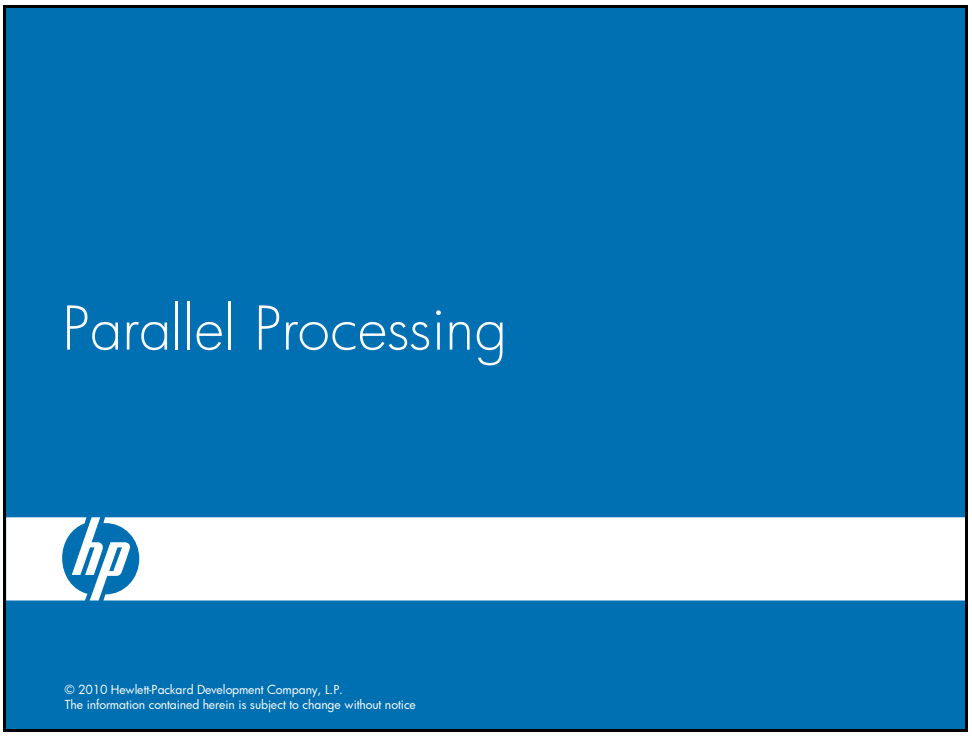
Exercise 3: XML Processing

Exercise 4: Remote Action Service

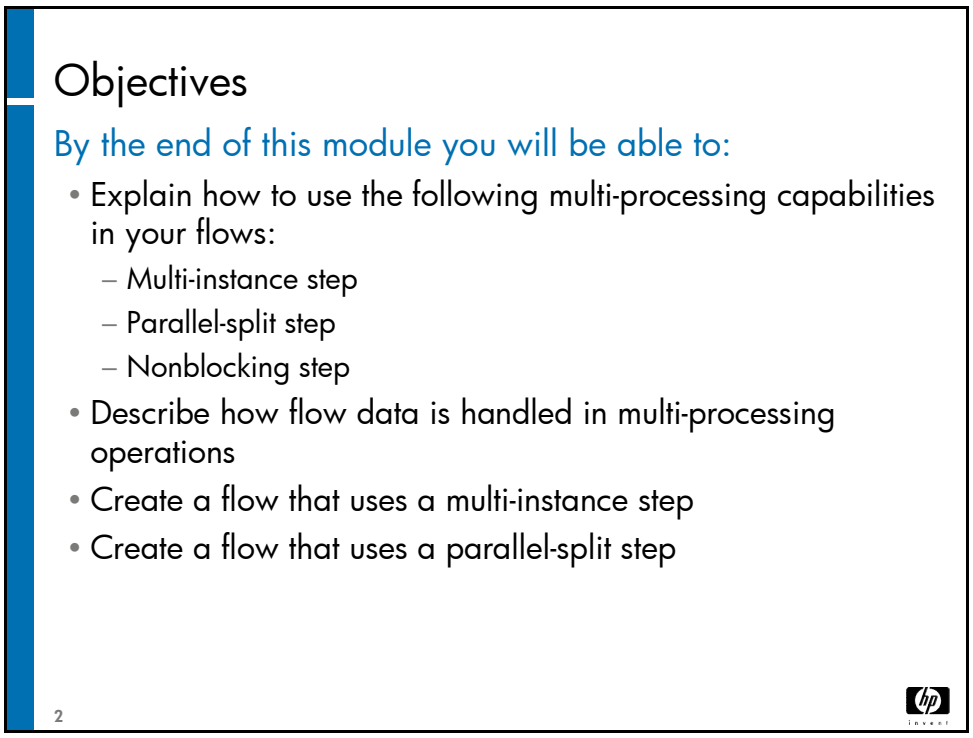
Exercise 5: Working With File Systems

Exercise 6: Working With Email

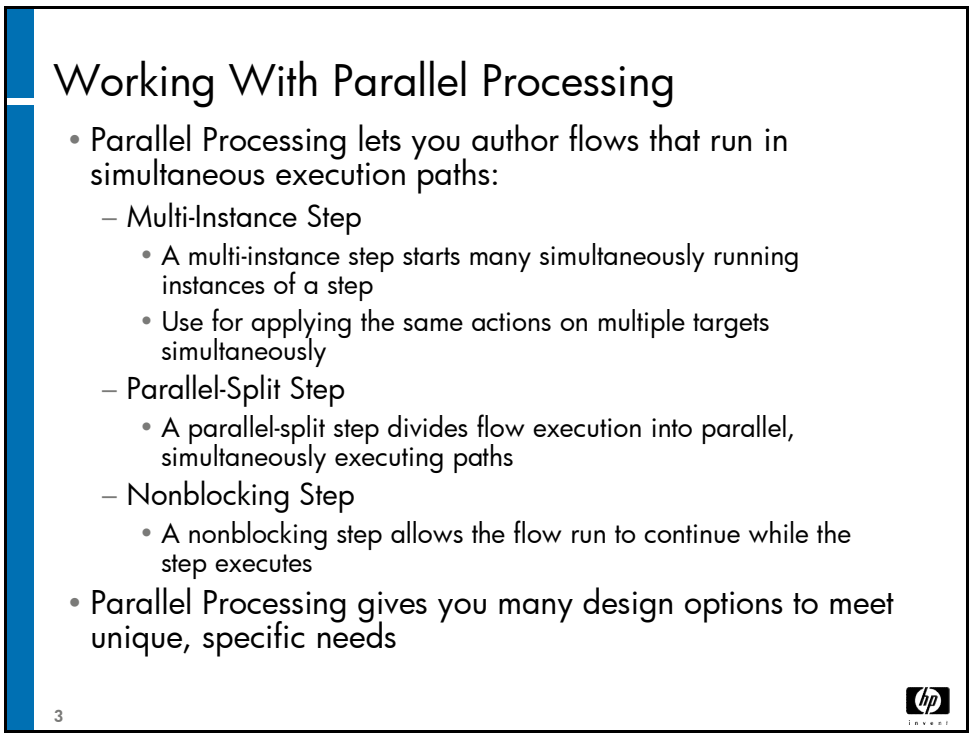
# Parallel Processing




In this section you will learn how to incorporate parallel processing into your flows to increase efficiency and execution speed.

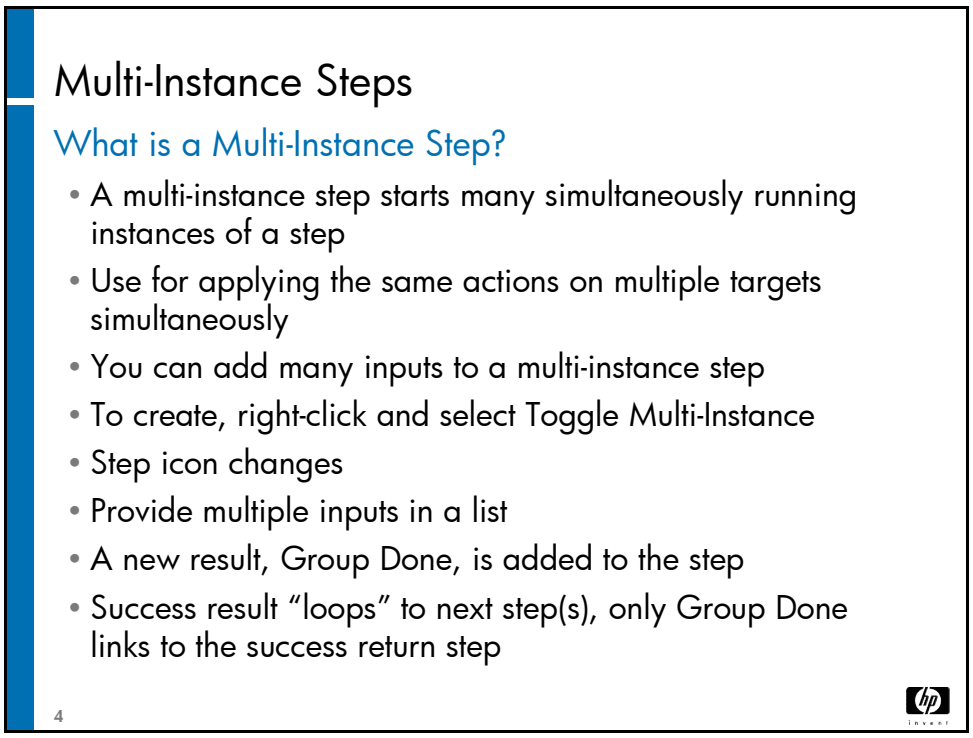



Review the objectives for this section. Here you will learn how to use three methods of parallel processing: Multi-instance steps, Parallel-split steps, and Nonblocking steps.




With parallel processing you can author flows that execute along simultaneous execution paths. There are three types as listed on the slide. Each of these is examined in detail in this module.

Parallel processing opens up many design possibilities and allows you to design more efficient flows.




A Multi-Instance step is one where multiple instances of a step run simultaneously.

This is useful for applying the same actions across multiple targets simultaneously.

Specifying a multi-instance step is basically a toggling exercise – you simply right-click a step in the Design panel and select Toggle Multi-Instance.

When you do this, the icon changes with a new response, Group Done.

You can also provide lists of inputs, like hostnames or IP addresses, for specifying multiple targets.

In other words, a multi-instance step is a very easy and convenient way to basically iterate the same actions to a series of targets. Once all instances are completed, the flow proceeds down the path specified by the Group Done response.

Multi-instance applies only to a step and not to an underlying operation, so turning on multi-instance has no effect on other flows that reference the same operation.

# Using Multi-Instance Steps

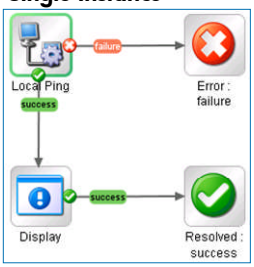
## Single Instance:

- Local Ping success response links to Display, which links to Resolved return step

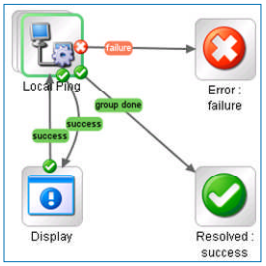
## Multi-Instance

- Local Ping icon changes
- Group Done response is added
- Success response iterates to Display until all instances are complete
- Group Done response links to Resolved return step when all instances are complete

**Single Instance**




**Multi Instance**

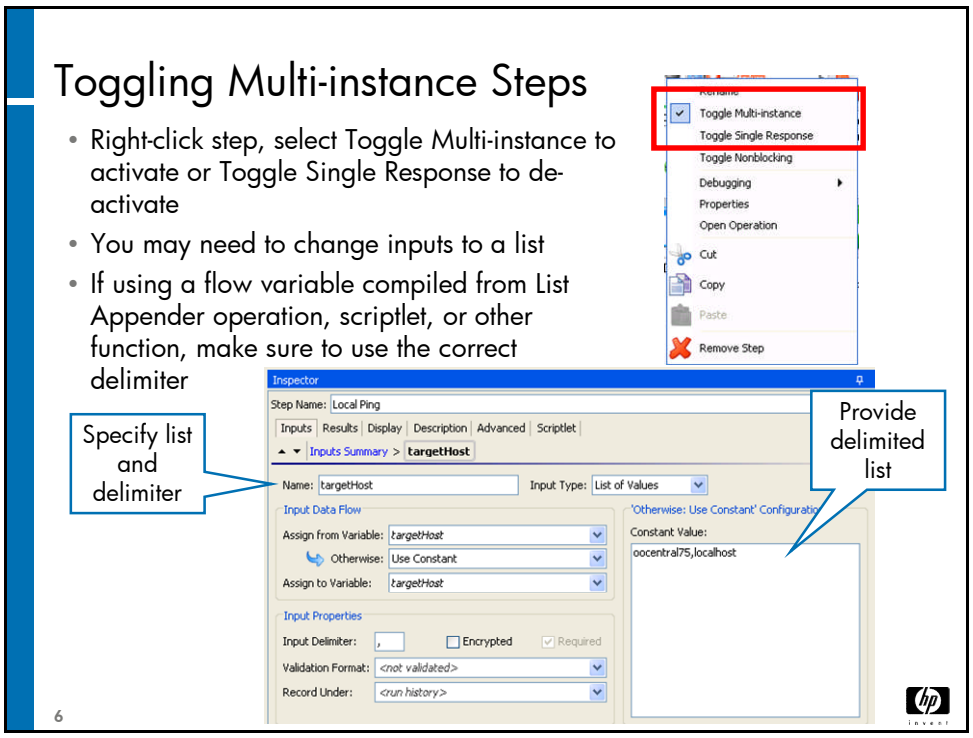



5

The example shows how to use a Multi-Instance step to ping a number of target hosts simultaneously.

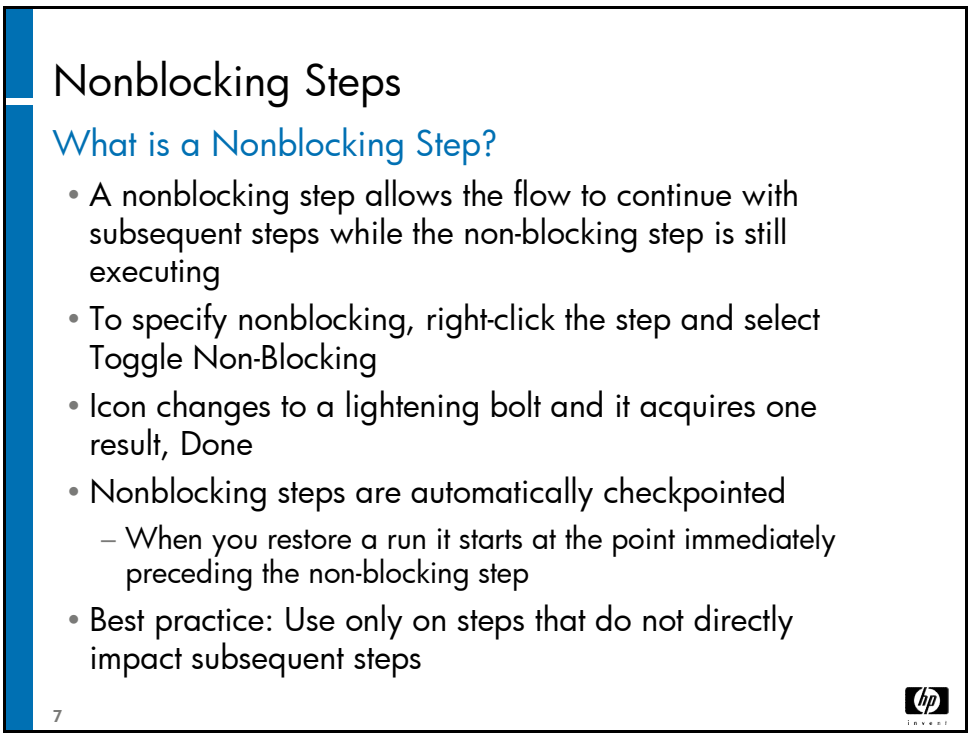
Simply right-click the Local Ping step and select Toggle Multi-Instance. Note the changes to Local Ping. The changes require you to reconnect transitions to accommodate the new Group Done response. You also need to provide a list of inputs to the multi-instance step.

At run time, the multi-instance iterates through all the hosts specified in the input list until it reaches the Group Done condition, when it then proceeds to the Success return step.




The slide shows specifically how to toggle Multi-Instance on and off, and how to specify the list of inputs for the multi-instance step. You basically specify the list and delimiter.

The list can be a constant value or a flow variable with multiple entries separated by the specified delimiter.




A non-blocking step is simply a step that allows the flow to proceed while it is executing.

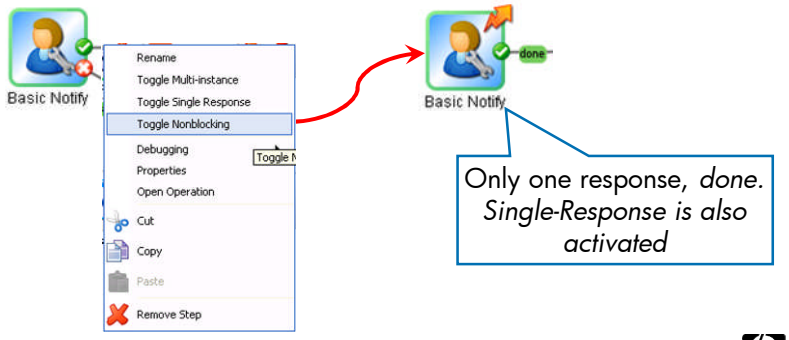
This is very useful, for example, when a step is a subflow that does not depend on the parent flow, and visa versa. You can simply set the subflow step as a nonblocking step and the flow run will proceed while the subflow is executing.

It's important to remember that a nonblocking step should only be used when subsequent steps do not directly require results returned from the nonblocking step.

If the flow run fails because of a service interruption or some other factor, the restored flow run begins immediately preceding the nonblocking step. In other words they are automatically checkpointed.

## Using Nonblocking Steps

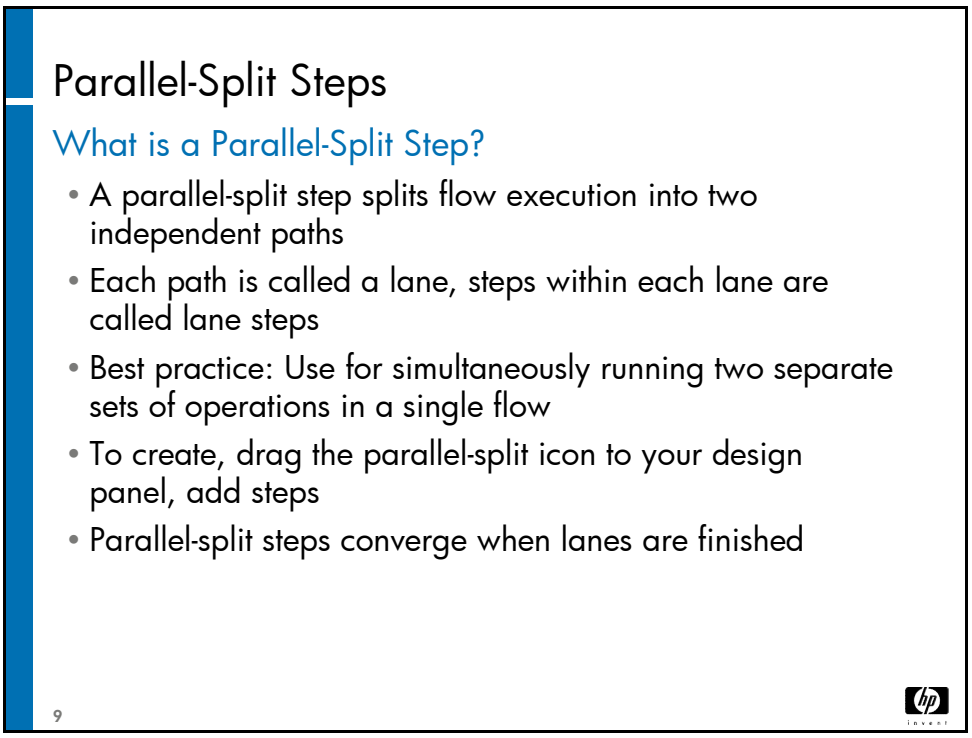
- Right-click the step, select Toggle Nonblocking
- Step changes:
  - Lightening bolt icon indicates non-blocking
  - Only one response, done
  - Type changes to Single Response




8

To use a nonblocking step, simply select the step by right-clicking and select Toggle Nonblocking.

When you toggle nonblocking on, the step gets only one response, done. Therefore you will need to reconnect transitions appropriately after specifying the step as nonblocking.




A parallel split step is the most complex of the three parallel execution steps.

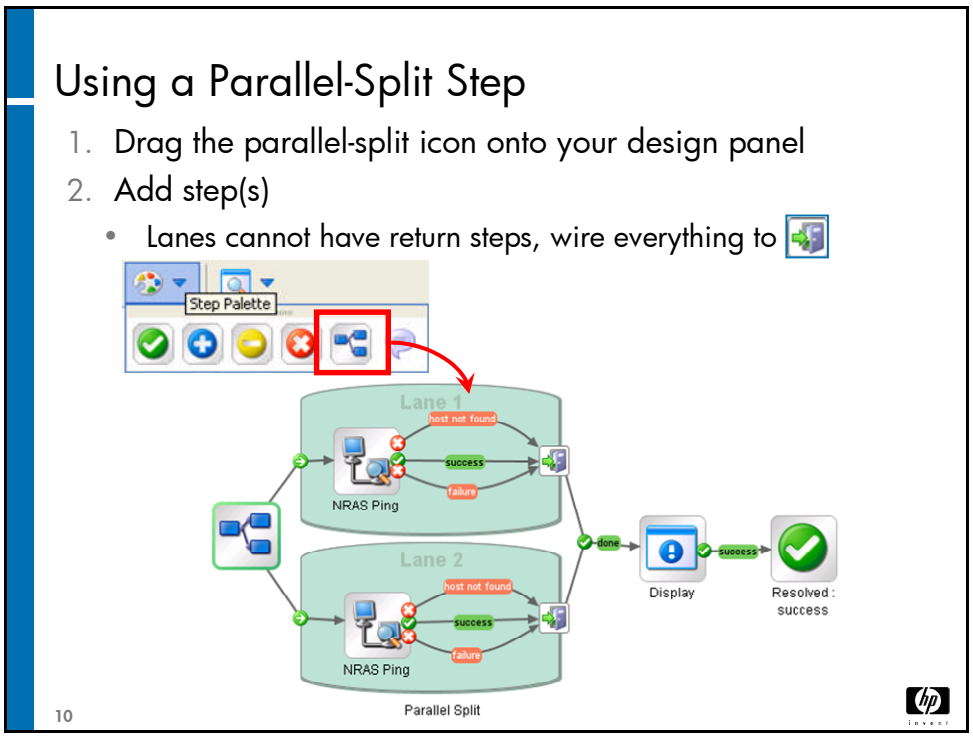
In a parallel-split step, flow execution splits into two or more separate execution paths.

Each path is called a lane, and steps within each lane are called lane steps.

A parallel-split is useful if you want to establish two or more entirely separate sets of operations within a flow and allow them to run simultaneously.

A parallel-split step increases runtime overhead because each lane uses a separate thread in addition to the thread being used by the flow itself. This is an important performance and load balancing consideration. If you have a lot of flows that use parallel-split steps you may need to increase the thread pool.

There are limitations on what you can do inside each lane. For example, you cannot pass a flow variable from one lane to the next at run time, and all lanes converge when completed – there are no return steps within the lanes.

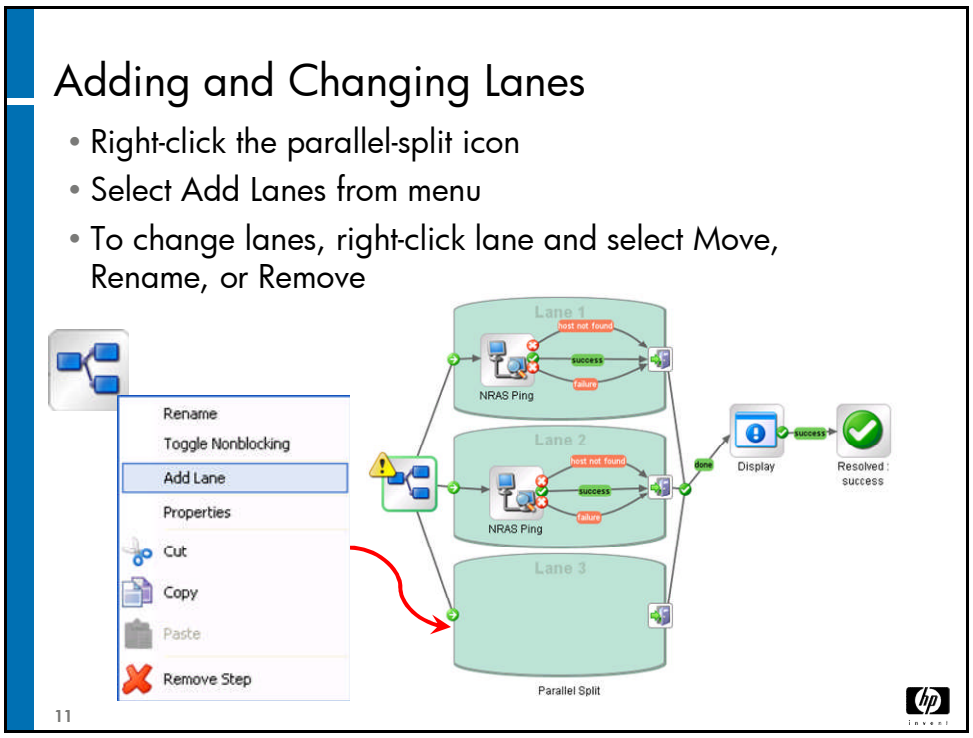



To use a parallel-split step, open the Step Palette and drag the parallel-split onto your design panel.

You can then add steps within each lane. Those steps will execute in parallel at runtime.

Note the way the steps converge, with responses linking to the "exit door" on each lane, which then links to a single Success response.

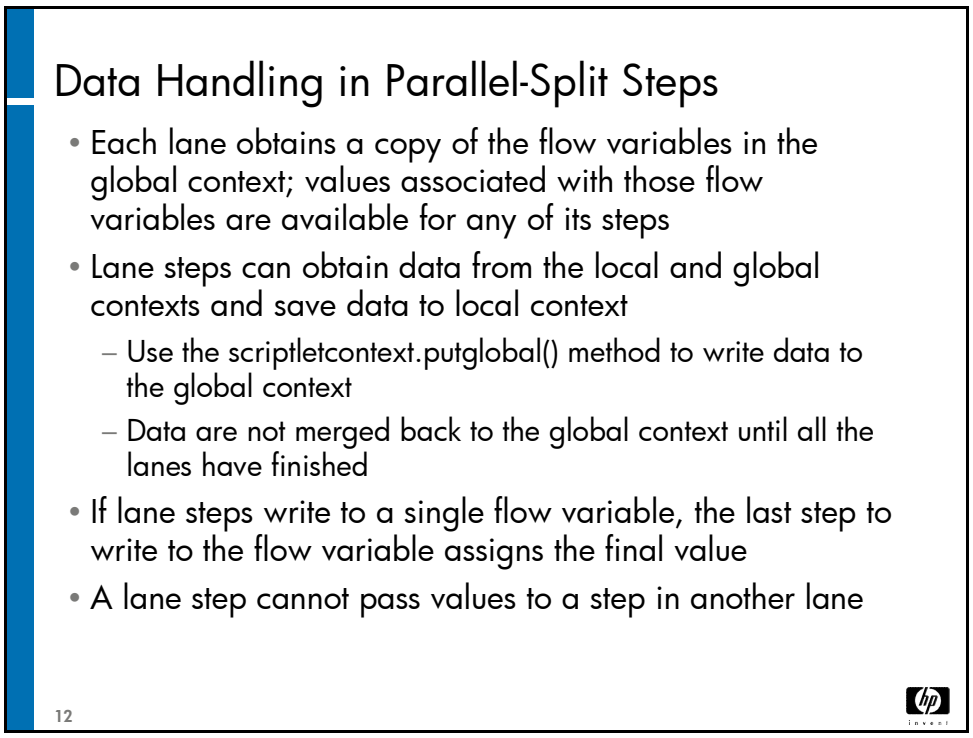
You need to be mindful of the limitations when working with a parallel-split step. In some instances it might make more sense to use subflows rather than parallel-split lanes if the level of complexity required exceeds the limitations presented by using the parallel-split step.




It's easy to add, change, move, or remove lanes.

Just right-click on the parallel-split icon and take the appropriate action.

The example shows how to add a lane to an existing parallel-split step.



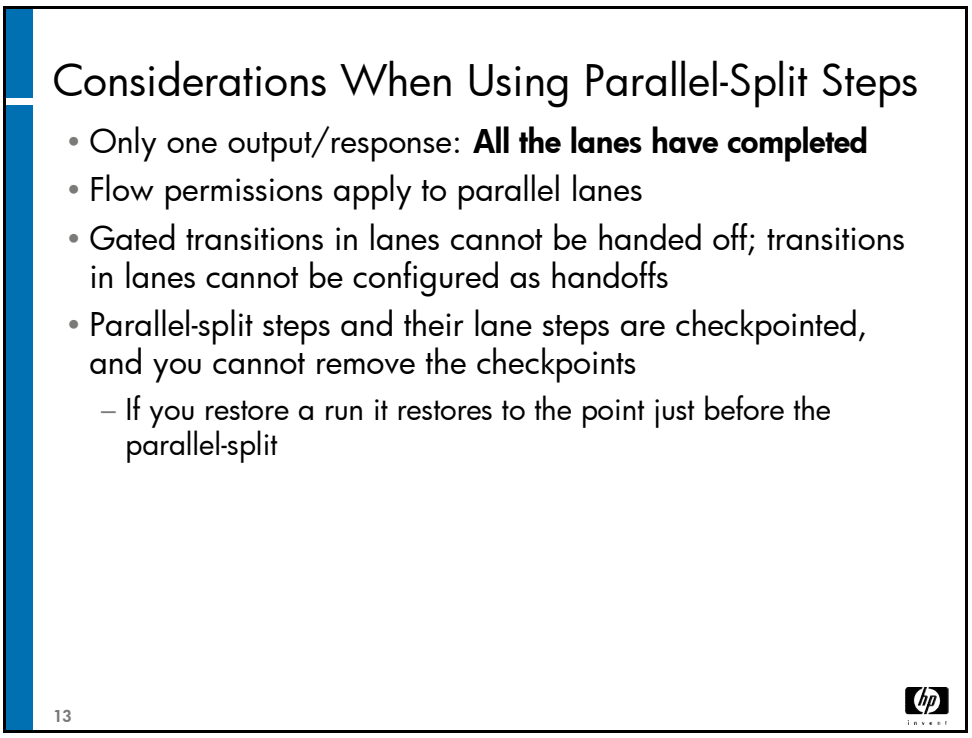

Review the data handling considerations when working with parallel-split steps.

If your data requirements exceed the limitations imposed by the parallel-split step then you should consider using subflows rather than the parallel-split step to accomplish the task.

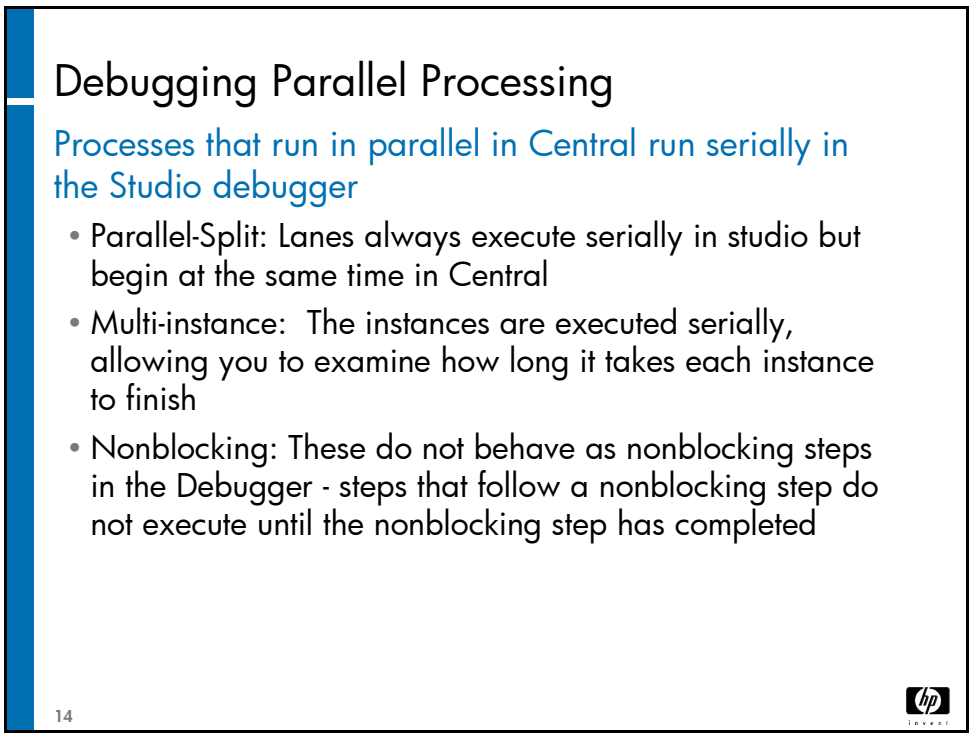
You need to use the scriptletcontet.putglobal() method to write data to the global context from within a lane.

If a lane writes to a single flow variable at runtime, the last step to write to the flow variable assigns the final value.

Finally, you cannot pass values to a step in another lane.




Review the considerations listed on the slide. If you have an application that exceeds the limitations of working with parallel-split steps then consider using subflows instead.

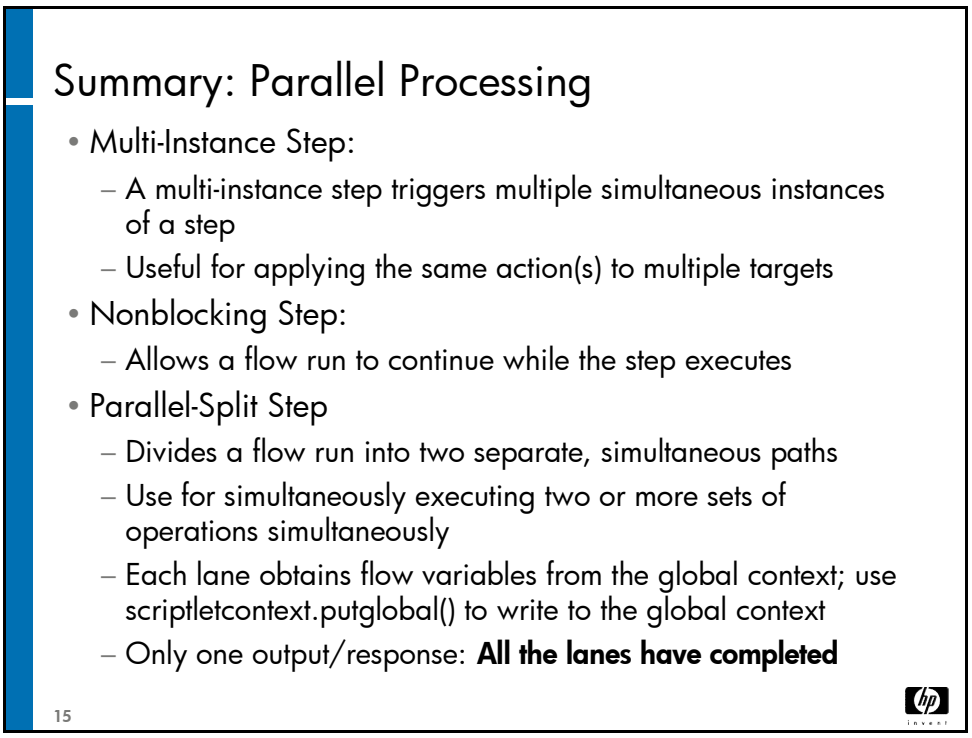



It's important to note that there are differences in how flows run in Studio as compared to Central.

In the Studio debugger, parallel lanes execute serially, not simultaneously as they do in Central. Therefore the flow will behave differently and take longer to execute in Studio.

The same is true for multi-instance steps – each instance executes serially in Studio. So if you need to time a multi-instance step you should do so in Central.

Nonblocking steps do not behave as nonblocking in Studio, the flow run waits for the nonblocking step to complete before proceeding.

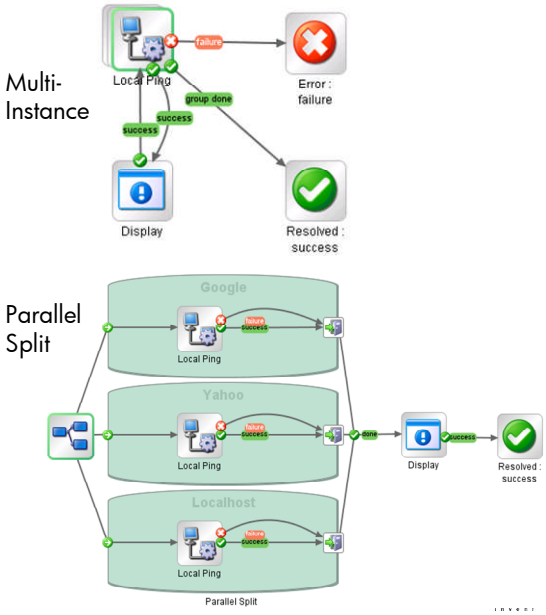



Review the summary for this section.

# Exercise 1: Parallel Processing

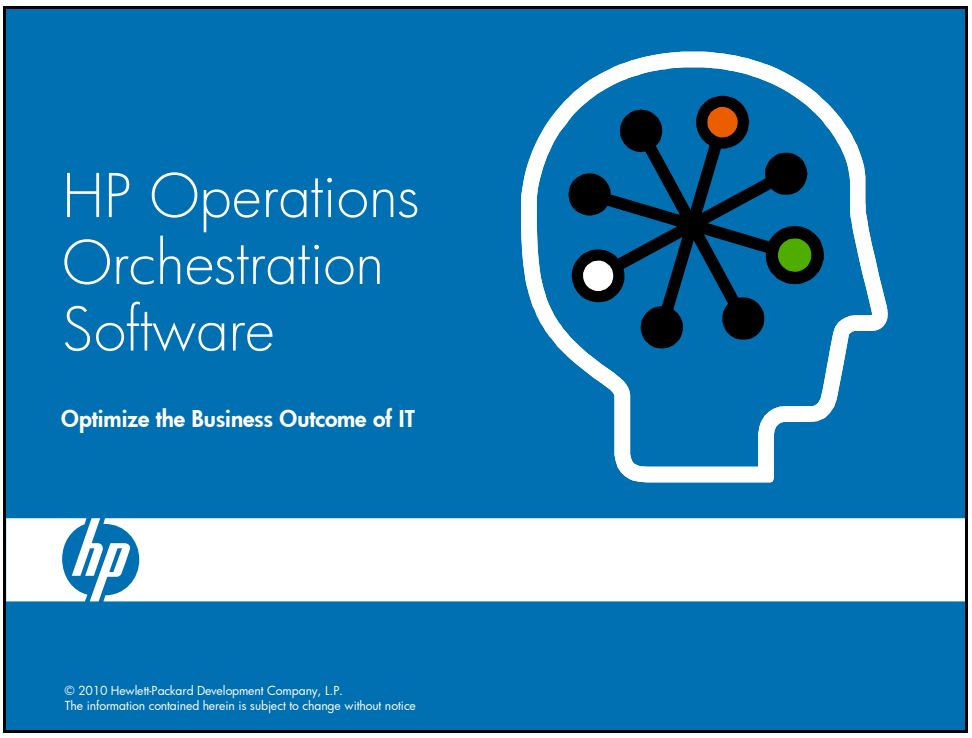
## Author flows that use:

- Multi-Instance Step
- Parallel Split Step

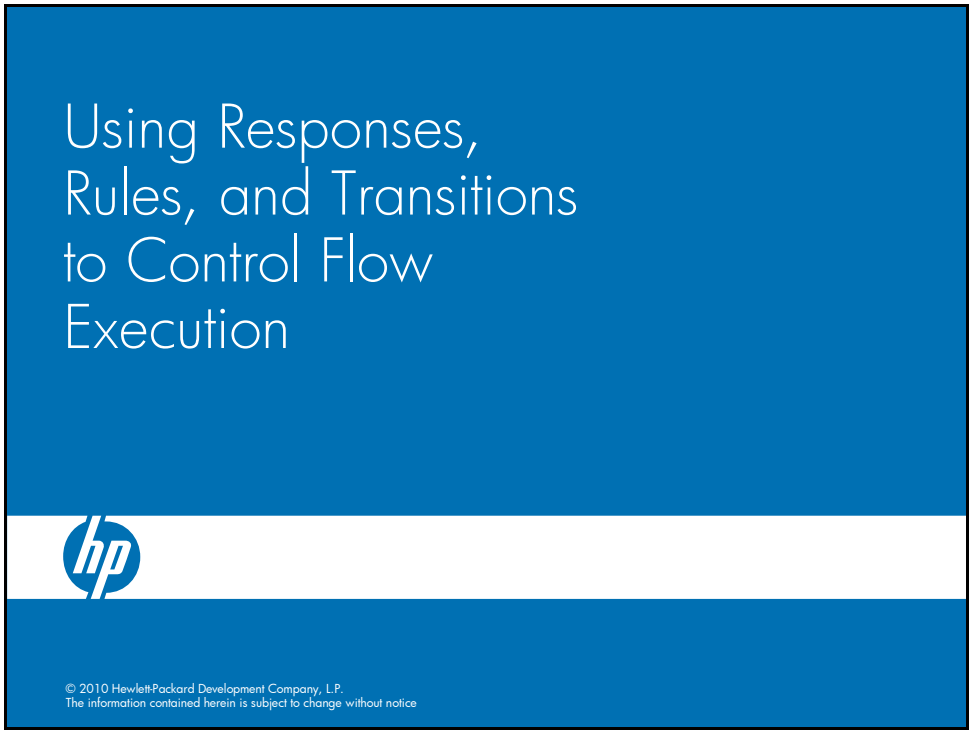



16

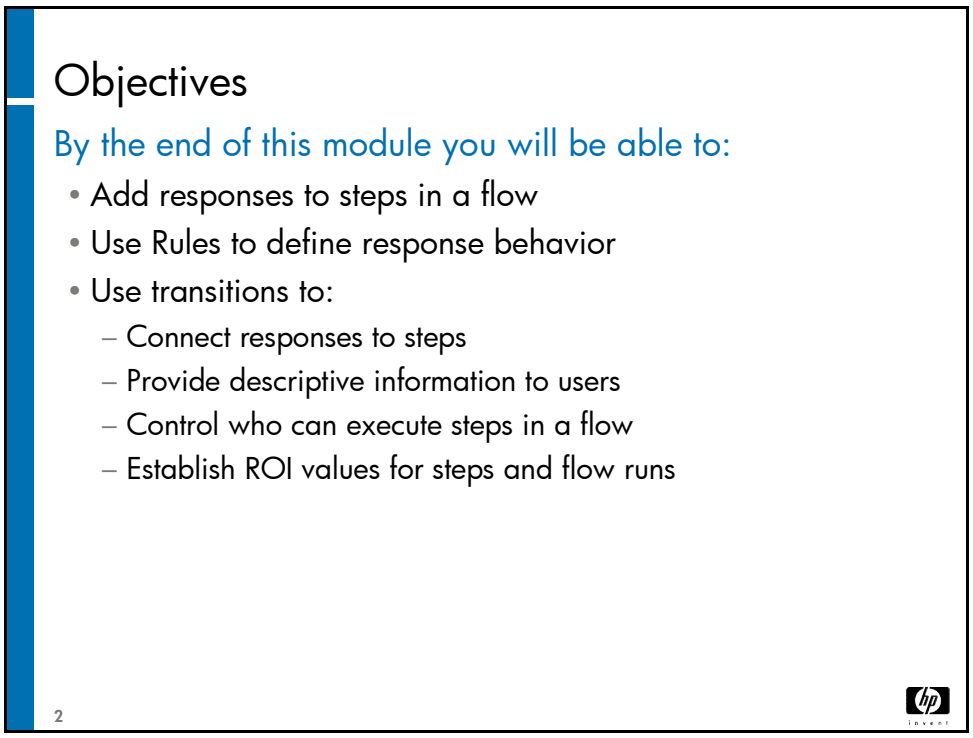
In the exercise you will work with all three types of parallel processing – nonblocking, multi-instance, and parallel-split steps.



# Using Responses, Rules, and Transitions to Control Flow Execution

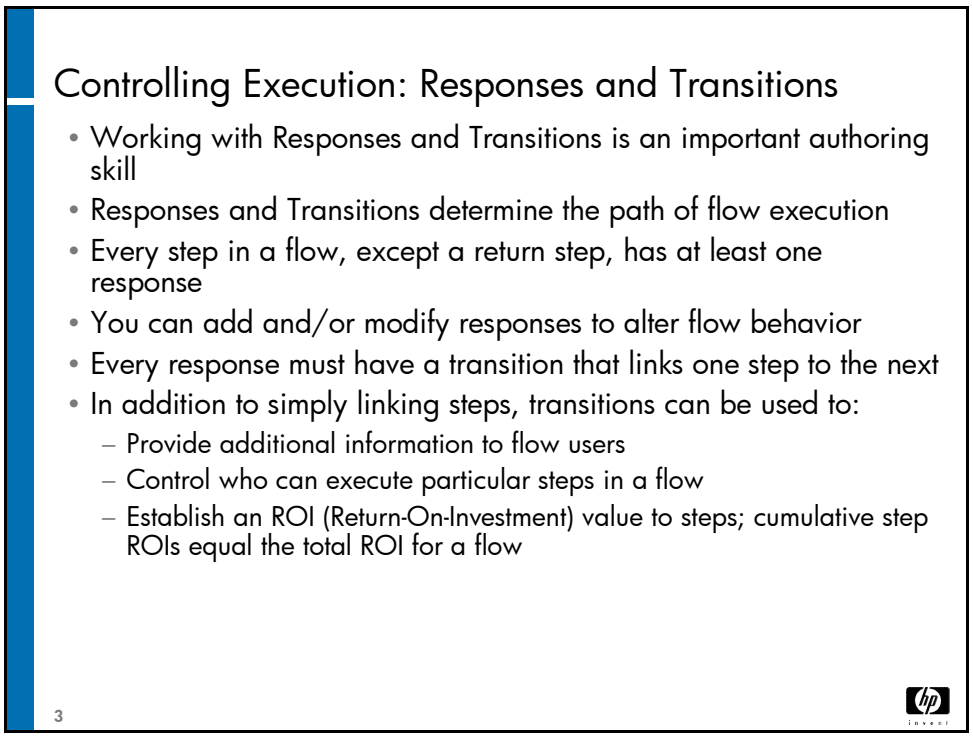



In this section you will learn how to use responses to direct the execution path of your flow depending on conditions that you specify in your flow.




In this section, you will see how to modify the properties of an operation to add responses – success, failure, no action, and diagnosed – and then how to determine rules to be followed that will cause a particular response to be taken at runtime.

Additionally you will learn more about how to use transitions to control flow execution, establish a return-on-investment value, and provide descriptions to users.

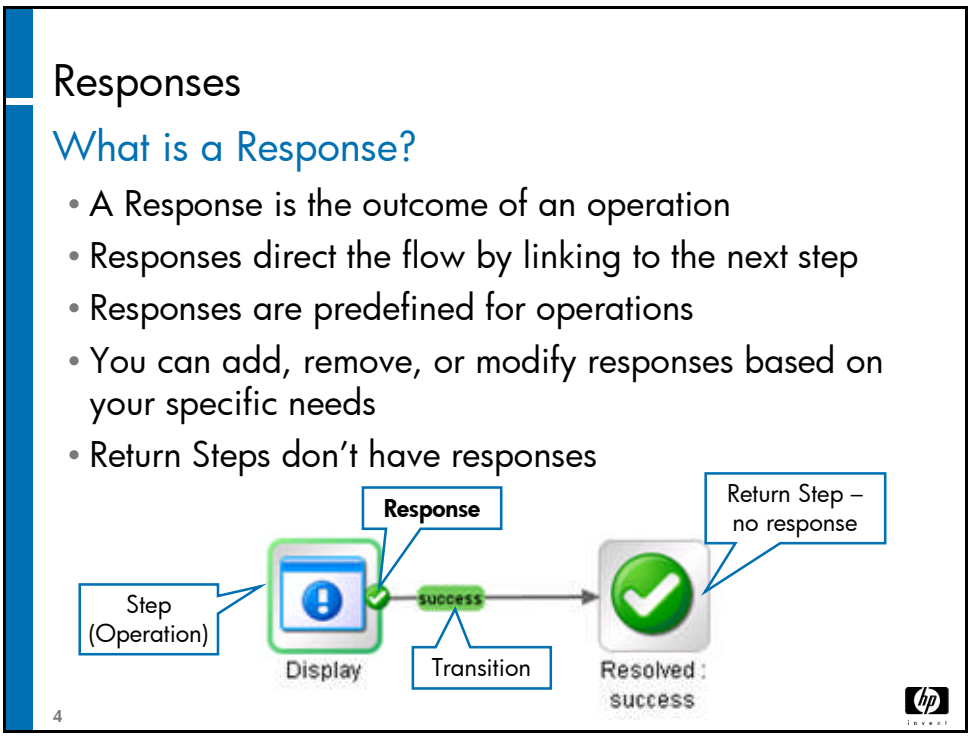



Working with responses and transitions is a very important skill and allows you to gain full control over how your flow executes based on conditions that are detected or become available at runtime.

These two topics are presented together because every response on a flow step requires a transition to connect it to a subsequent step.

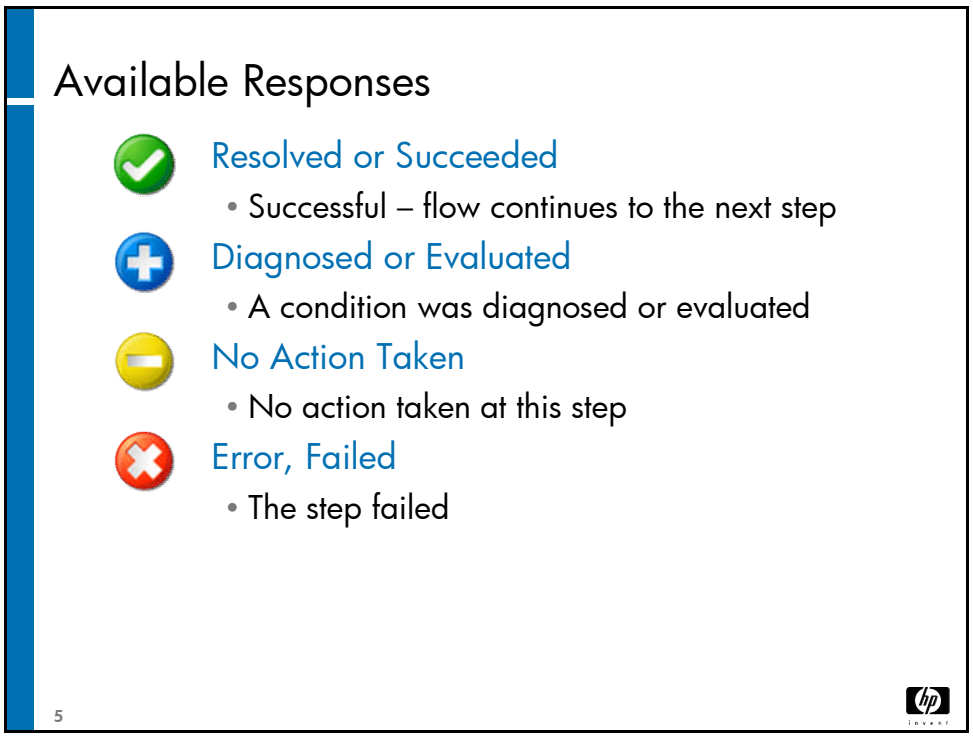
By adding responses and/or applying rules to new or existing responses you can gain full control over the behavior of your flow.

Additionally, modifying transitions allows you to further control flow execution while providing valuable information to users at runtime.

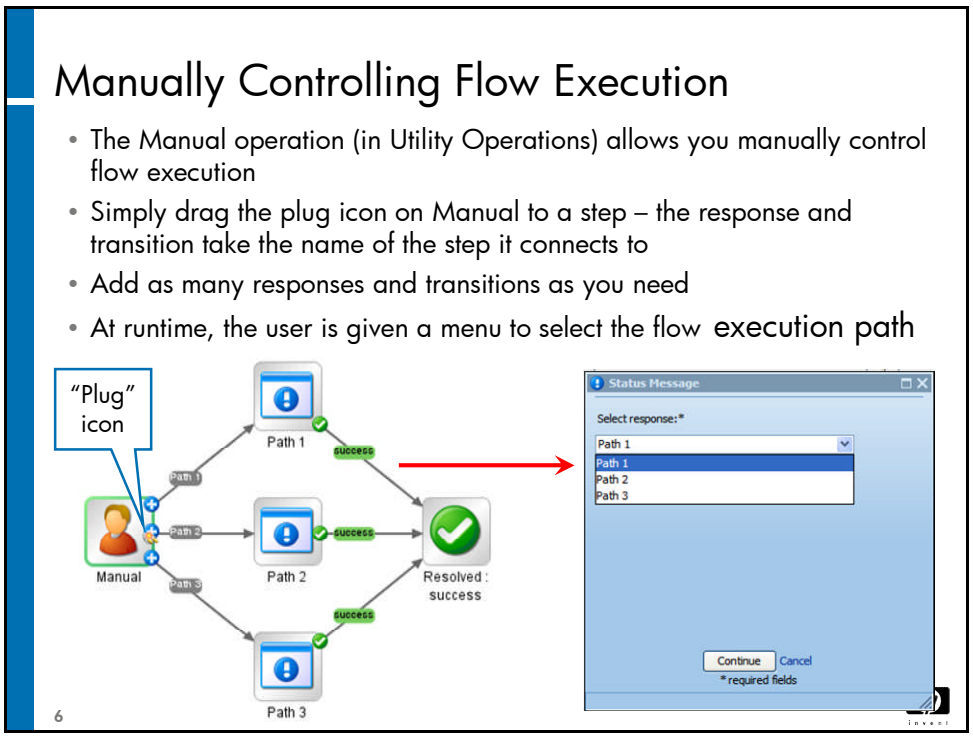



As you have already seen, a response is the outcome of an operation. A response directs flow execution by linking to the next step. Responses are predefined for operations in the OO library but you can add, remove, and change responses by modifying the Responses tab in the operation's properties editor. Additionally you can control what rules to follow so that a particular response is selected at runtime based on conditions present in your flow at that moment.

As you can see on the slide, every response must be connected by a transition to a subsequent step. Note that return steps do not have responses because they are the endpoints of flow execution.




The available responses are shown on the slide. Even though the behaviors listed on the slide – success, failure, etc. – are predefined for operations in the library, as a flow author you have complete control over determining what constitutes success, failure, etc. By assigning rules to a response – which you will learn how to do shortly – you have complete control over defining the behavior associated with a response.



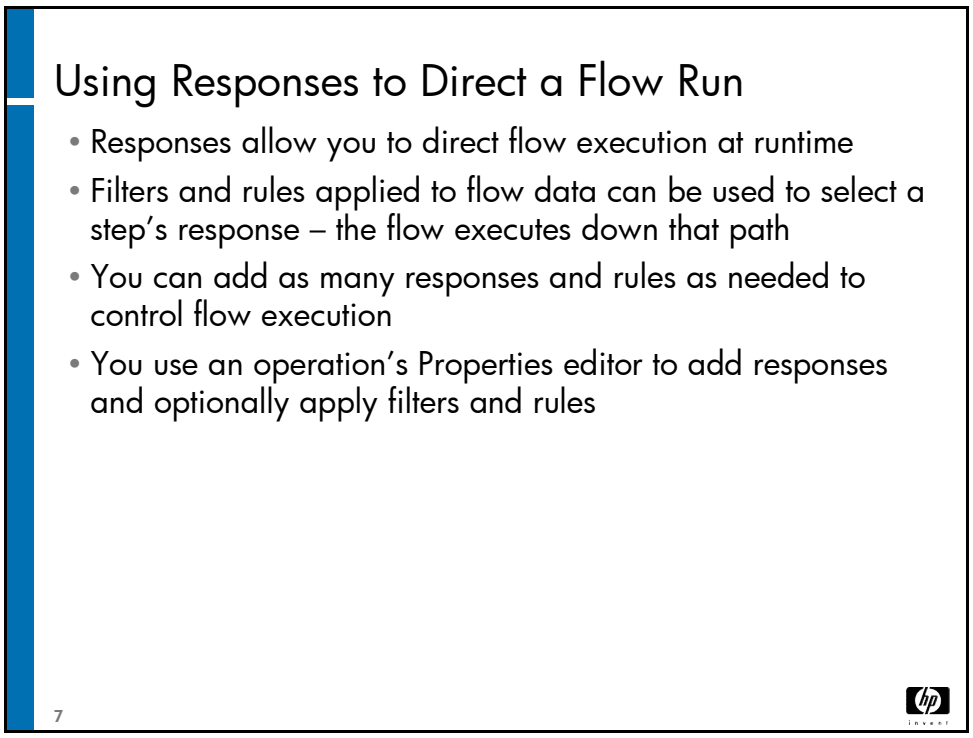

To begin learning how this all works, you can use the Manual operation in the OO library and build a small flow that demonstrates how to direct the path of a flow depending on a condition present at runtime, in this case, a user selection.

Simply drag Manual onto your design panel and add a few more operations, as shown on the slide – these are simply "do nothing" Display operations used for illustration purposes.

To wire this together, simply drag the plug icon on Manual to a subsequent step. This automatically adds a named response to the step you connect it to.

When you run the flow, you will be presented with a menu. Selecting an item from the menu selects the execution path, depending on which response is "activated" when the selection is made.

The lab guide has an exercise where you build this flow.



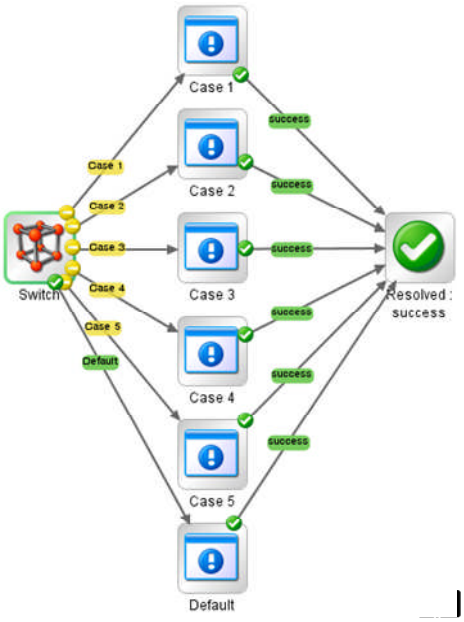

Responses allow you to direct flow execution at runtime.

The Manual operations is a very simple way to illustrate this because it automatically adds responses when you connect the plug icon to a subsequent step.

In practice you will be editing the properties for an operation and specifying responses there. Additionally you will use the Rule Editor to define what conditions will cause that response to be selected at runtime.

## Example: OO Switch

- Switch step has six responses
- At runtime, user is presented with a menu – Case 1 through Case 5
- When the user makes a selection, filters in Switch select the correct execution path
- Path selection can be accomplished without user interaction simply by applying a rule to a flow variable




8

In this example, the Switch operation has six responses, five "no action" and one "success".

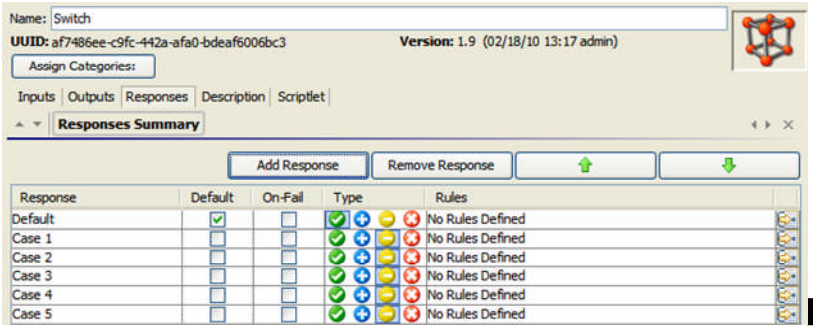
At runtime, the user makes a selection from a menu – the choices are Case 1 through Case 5. Then depending on which selection the user makes, that particular path is selected at runtime.

Note that the condition that directs flow execution can be anything that can be detected or measured in the system at runtime, not just a user selection – the value of a flow variable, the result of a previous step, etc. All of these can be used to define the rules that will cause a particular response to be selected, depending on the rules you have defined for the response.

In this example, the Switch step is simply a modified Display operation whose properties were modified to include the new responses and rules.

## Adding Responses to an Operation

- Open the Operation Properties
  - Double-click in Library, or right-click operation in Design Panel and select Open Operation
- Select Responses tab, click Add Response
- Name the response
- Select the response Type
- Specify Rules and Filters

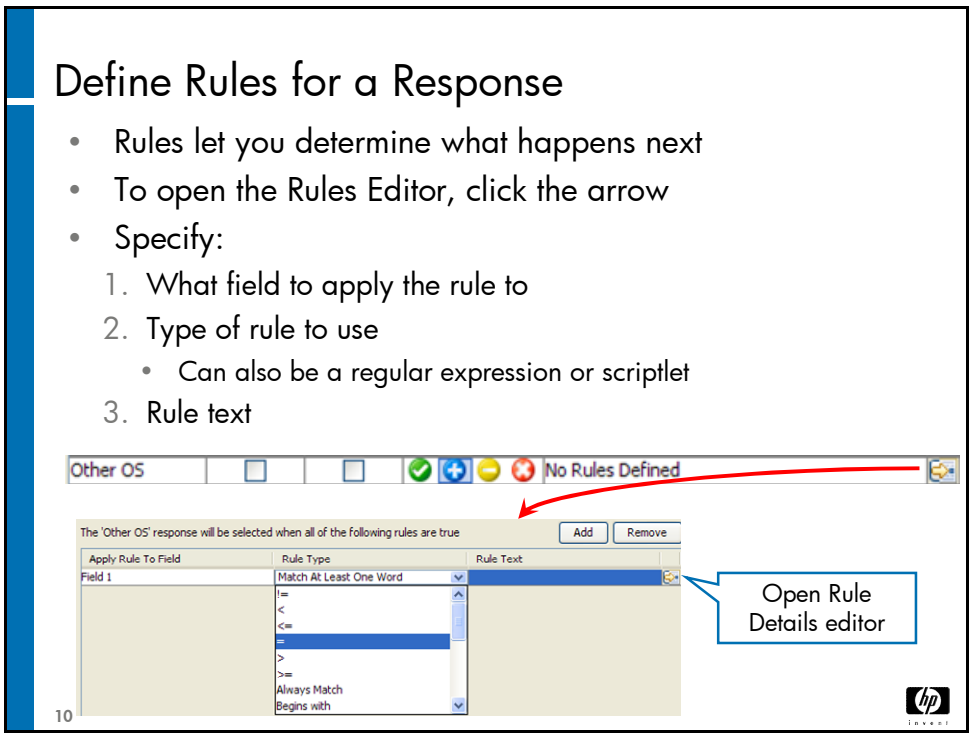



9

To add responses to an operation, double-click the operation in the Library or select Open Operation in the Design panel to open the properties editor.

A word of caution – make a copy of an operation before adding responses. If you add a response to an operation that is shared among different flows, every flow that calls that operation will have an Unhandled Response error and will not run. Working on a copied version of the operation avoids that problem.

To add the response, select the Responses tab and add the responses you need. Initially no rules are defined, as shown on the slide.

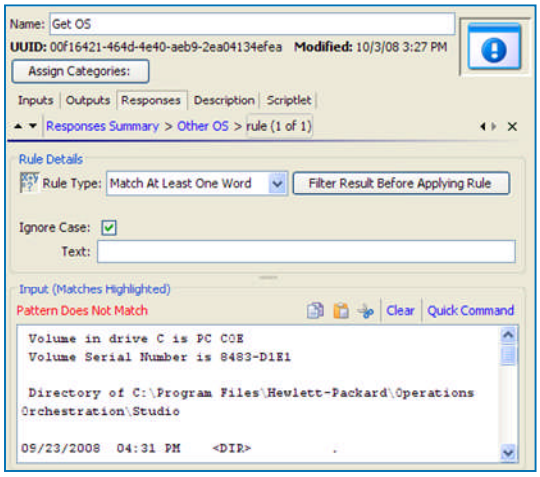



To define rules, just open the Rules editor and add the rules you want – for example, in this case the field Field 1 is being tested to see whether it exactly matches a text string. Field 1 would be visible in the Inputs of the Step Inspector.

You can apply more than one rule and you can also open the Rule Details editor to filter complex output and test for a particular value. The rule can also be a regular expression or scriptlet result.

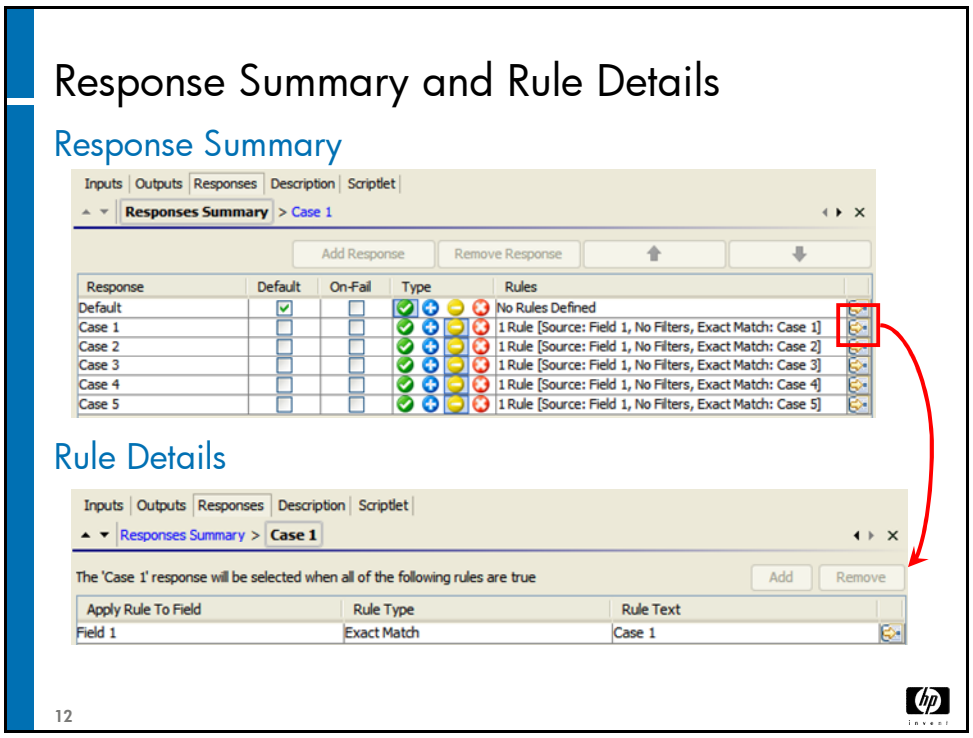
## Rule Details Editor

- Click the arrow icon ( ) in the Rules editor to open the Rule Details editor
- Allows you to specify advanced options for the rule
- Click Filter Results to apply advanced filters or regular expressions
- Click Scriptlet tab to use a scriptlet for the rule
- Use Quick Command to generate sample data for defining filters

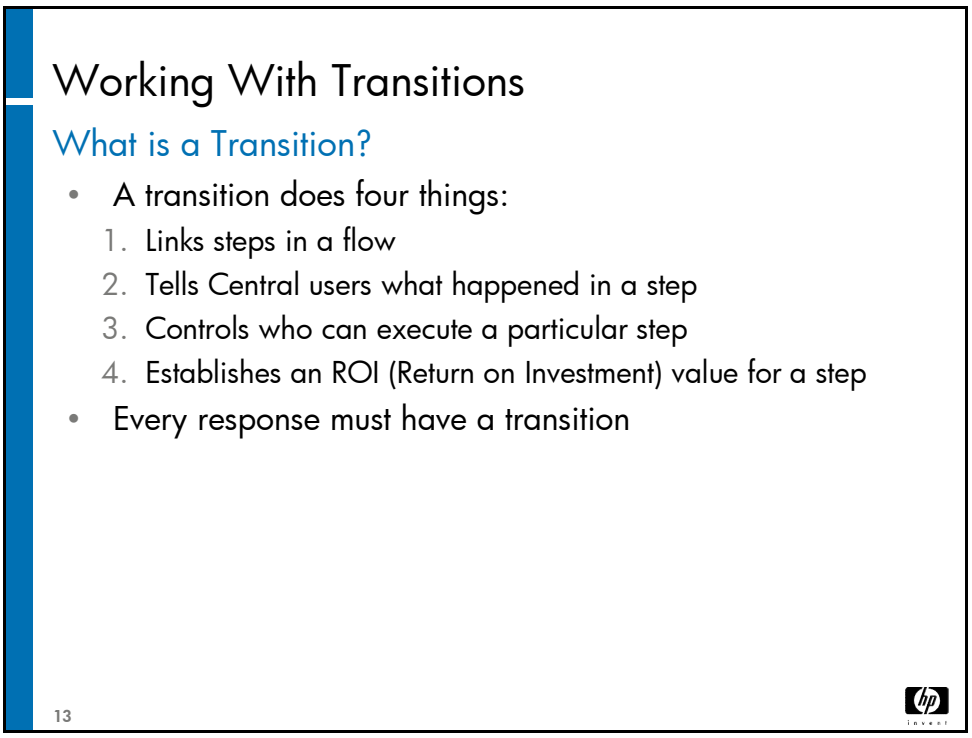



11

The Rules Detail Editor is very similar to the Filter Editor – it allows you to enter test output, either by copying and pasting or with Quick Command – and develop a filter to extract the value you want to test for in the rule.



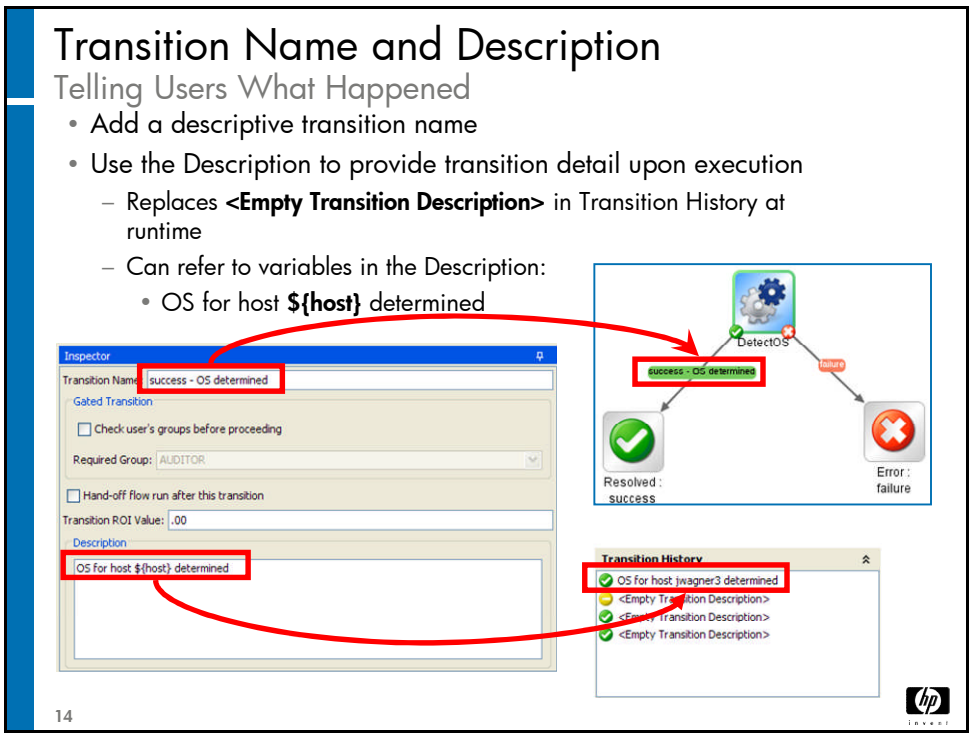

Once defined, the rules are summarized on the Responses tab. You can always click the Rule Editor to change a rule. In the example shown, the Case 1 response will be selected if the value for Field 1 is an exact match of the text “Case 1” which is provided by user selected at runtime. If the user provides a different value – Case 3 for example – then the Case 3 response is selected.




You have already worked a lot with transitions. Every time you connect one step to the next, you are simply connecting a response to a subsequent step with a transition. In other words, the transition is the arrow that connects steps in a flow.

But there's much more to a transition than that. With a transition you can tell Central users what happened in a step, control which group can execute a particular step, and establish an ROI value for a step.

Every response must have a transition that connects it to a subsequent step. The exception is return steps which do not have responses and thus serve as endpoints for a particular execution path.




One important role the transition plays is to inform the user what happened at a particular step. You use the Transition Name and Description for this purpose.

The Name is the label that appears on the transition in the design.

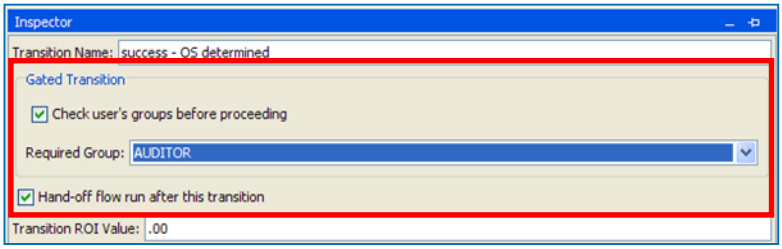
The Description appears in the Transition History when debugging the flow and also shows up appropriately in the reporting for the flow run.

By using the Transition Name and Description you can add a lot of value to your flows by making them more descriptive, which is a big benefit when the flow moves into production.

# Gated Transition

## Controlling Who Performs the Next Step

- Use a Gated Transition to check the group
- Select Hand-off flow run to require a new group to proceed with the flow execution
  - Select "Check user's groups before proceeding", specify the Required Group
  - Only Central user(s) in the specified group can continue the flow run




15

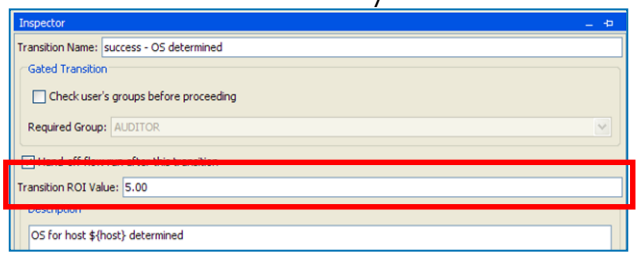
A Gated Transition allows you to control who performs the next step in the flow. A gated transition checks the group at runtime and if a different Required Group is specified, the flow run is handed off to that group.

At runtime there are various ways of alerting users that a flow run is being handed off. Emails with execution links can be sent, for example, when the flow is handed off.

# Specifying ROI

## Establishing a Value for a Step

- ROI determines the value for a particular step
- Cumulative ROIs determine value for a flow run
- You can use different approaches for calculating ROI, for example:
  - You know that a particular step in a flow replaces a manual process that takes 15 minutes
  - You were paying someone $20/hr to perform that step manually
    - Step value: $20 * 15/60 minutes = $5
  - ROI values appear in Central run reports
  - You don't have to set an ROI value – you can leave the field blank

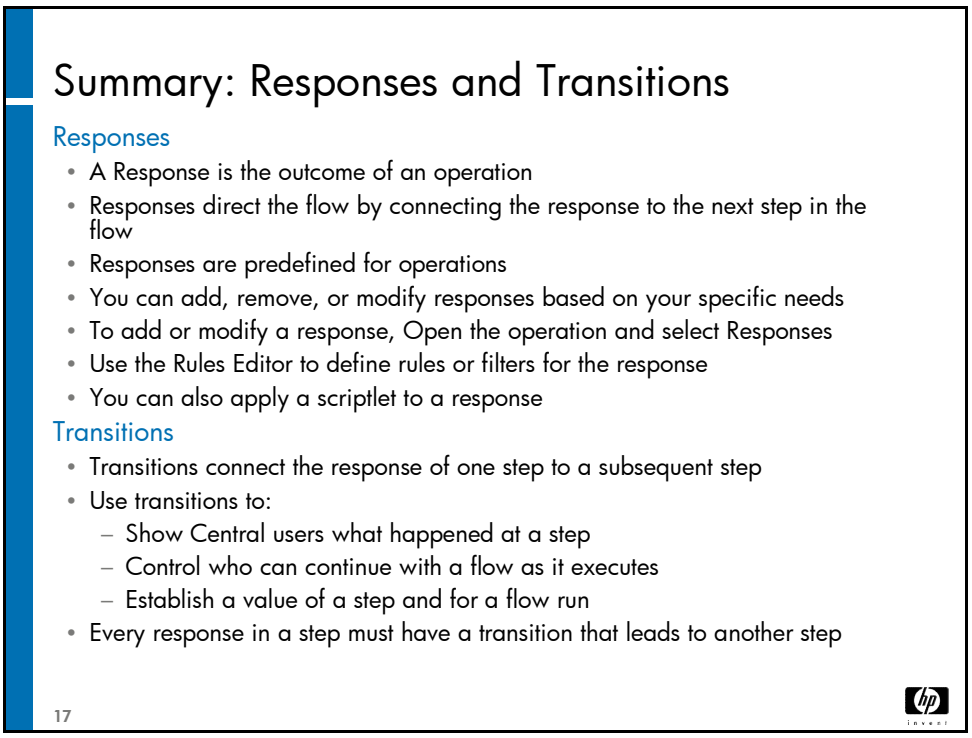



16

Finally you can specify an ROI for a particular step. The transition is where you specify the ROI.

The ROI is simply a monetary value. The flow author can use any method that makes sense to determine the value of a step. The cumulative step ROIs equals the value of the entire flow run.

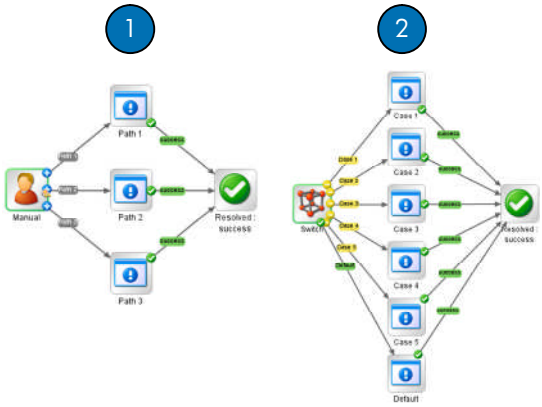
Specifying an ROI is optional – you don't need to enter an ROI value for any step in a flow.




This slide summarizes what we covered in this section. Make sure to review the summary and remember to make a local copy of an operation before adding or changing its responses.

## Exercise: Responses and Transitions

- Author flows that:
  - Use manual selection
  - Use responses and rules to direct flow execution



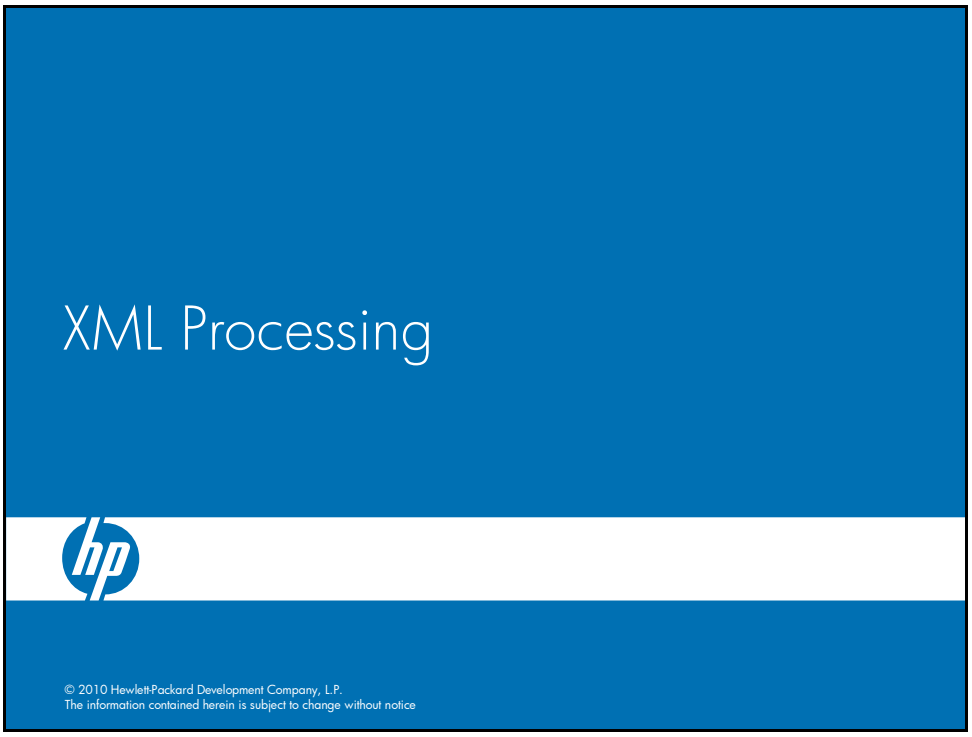

18

The exercise has two parts. In the basic exercise, students add responses to an operation that branch depending on values determined at runtime.

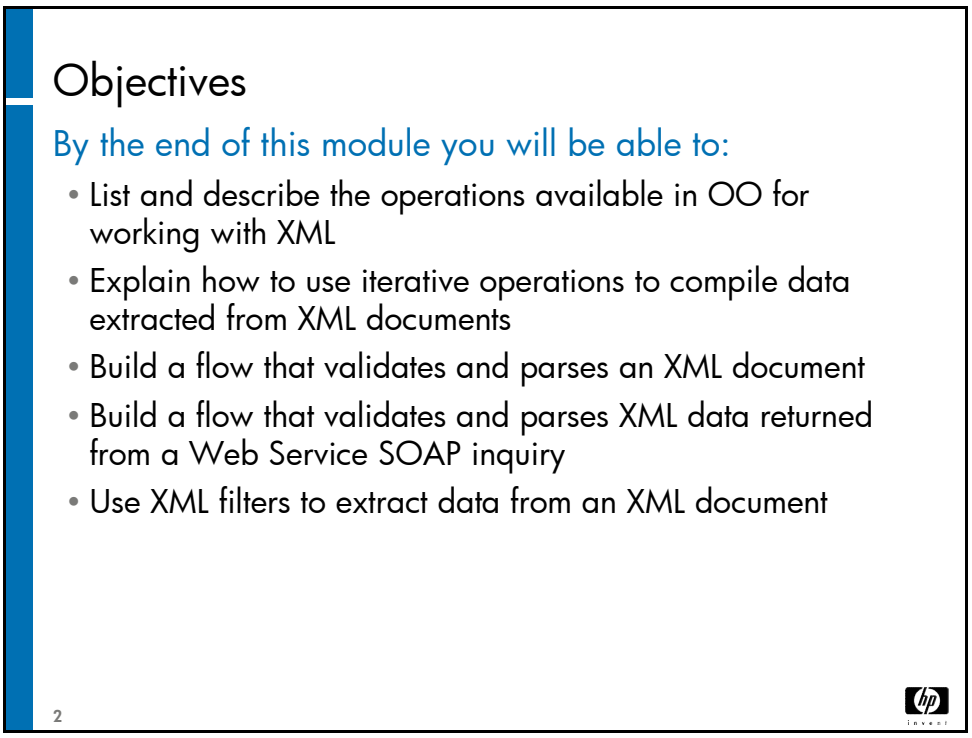
In the Advanced exercise Windows or Linux commands are executed depending on the type of target platform.

The Advanced exercise is optional.

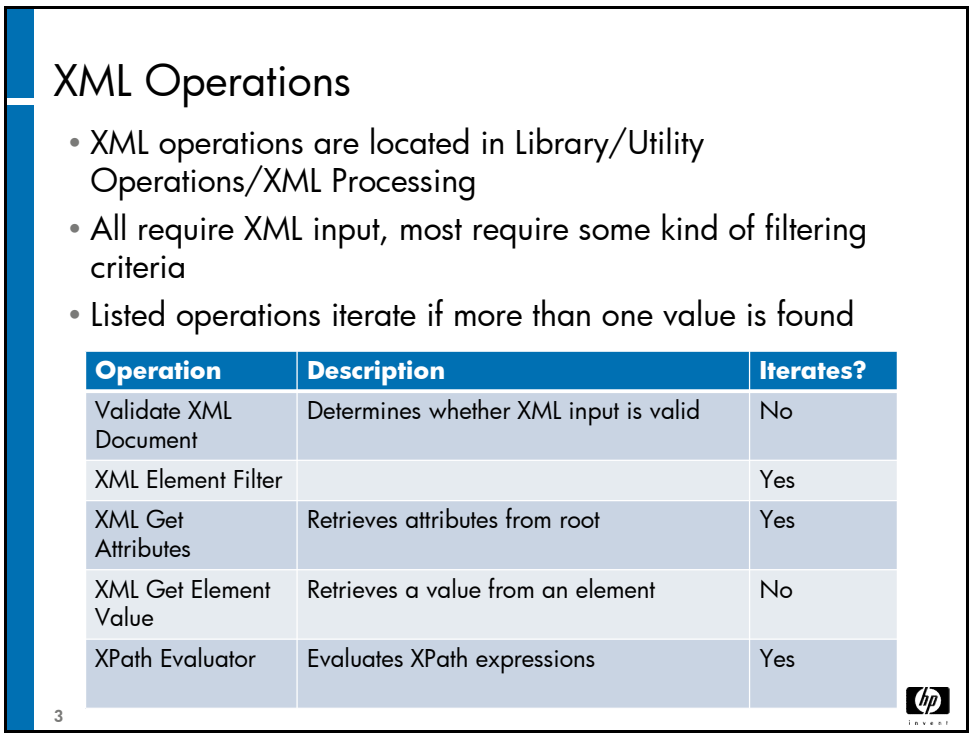
# XML Processing




In this section you will learn how to use OO's XML Processing capabilities to extract data from XML documents.




The goal of this module is to use the new XML operations available in the Library for processing XML content as well as the XML filters available in the Filter Editor.



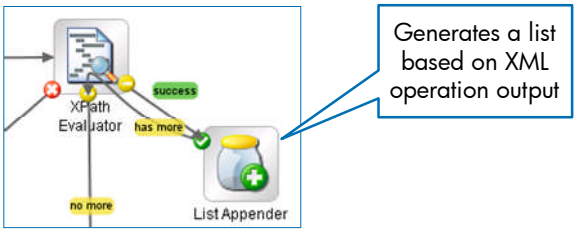

Operations Orchestration contains content in the library specifically designed for dealing with XML content.

The slide lists the operations, provides a brief description, and shows whether they are iterative operations or not.

Most of the XML operations iterate through the document so the process of working with iterative operations in OO apply to the XML Element Filter, XML Get Attributes, and XPath Evaluator.

## Iterative Operations

- XML Element Filter, XML Get Attributes, and XPath Evaluator iterate if more than one attribute/element is found
- Use with an iterative operation
- Example: XPath Evaluator and List Appender




| Assign To Input | Required | Type | From |
|---|---|---|---|
| keyName | ☑ | Single Value | Value: listItems |
| resultText | ☑ | Single Value | Result of previous step |
| delimiter | ☐ | Single Value | Value: , |

4

The iterative operations are typically paired with another operation that is intended to operate iteratively, like the List Appender.

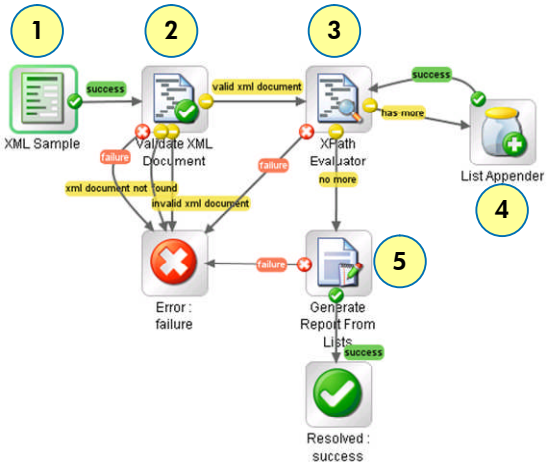
In the example on the slide, the XPath Evaluator operation iterates through the XML document and finds data that matches the XPath expression.

Whenever it finds data that matches the XPath expression it sends the result to the List Appender where it is added to a list. Control then goes back to XPath Evaluator and the process repeats until the No More condition is met. At that point the flow continues to the next step.

## Example

This flow:

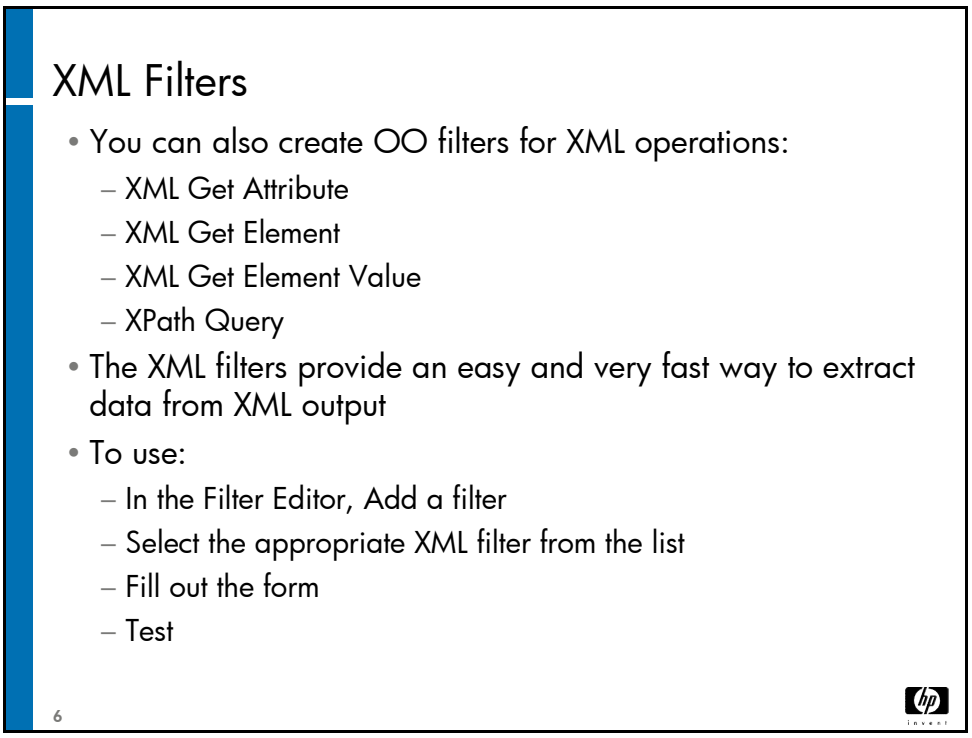
1. Generates XML data
2. Validates the document
3. Filters the specified element
4. Generates a comma-separated list
5. Writes a report to the Display



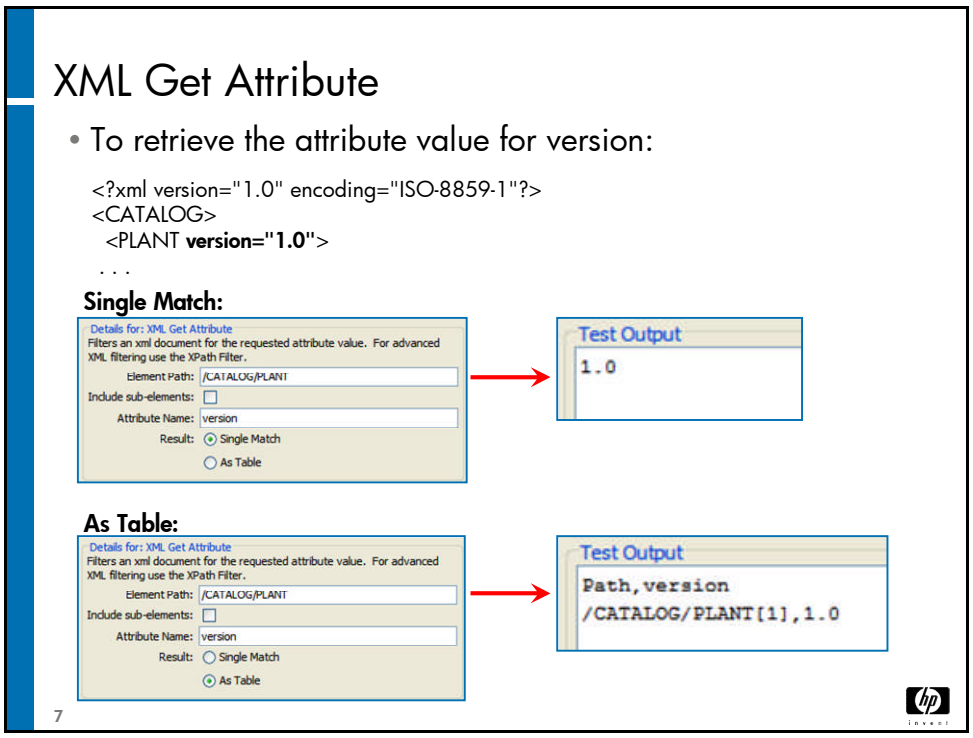

5

The example flow:

- Generates sample XML data and sends the document to the Validate step
- The Validate step checks whether it is a valid XML document. If it is control goes to the XPath Evaluator. If not, it goes to the error step.
- The XPath Evaluator prompts the user to enter an XPath expression and iteratively compiles a list using the List Appender operation.
- When there is no more data to evaluate, the compiled list goes to the Generate Report operation, which presents the user with a formatted report of the results of the XPath expression.
- Finally the flow ends at the Resolved step.

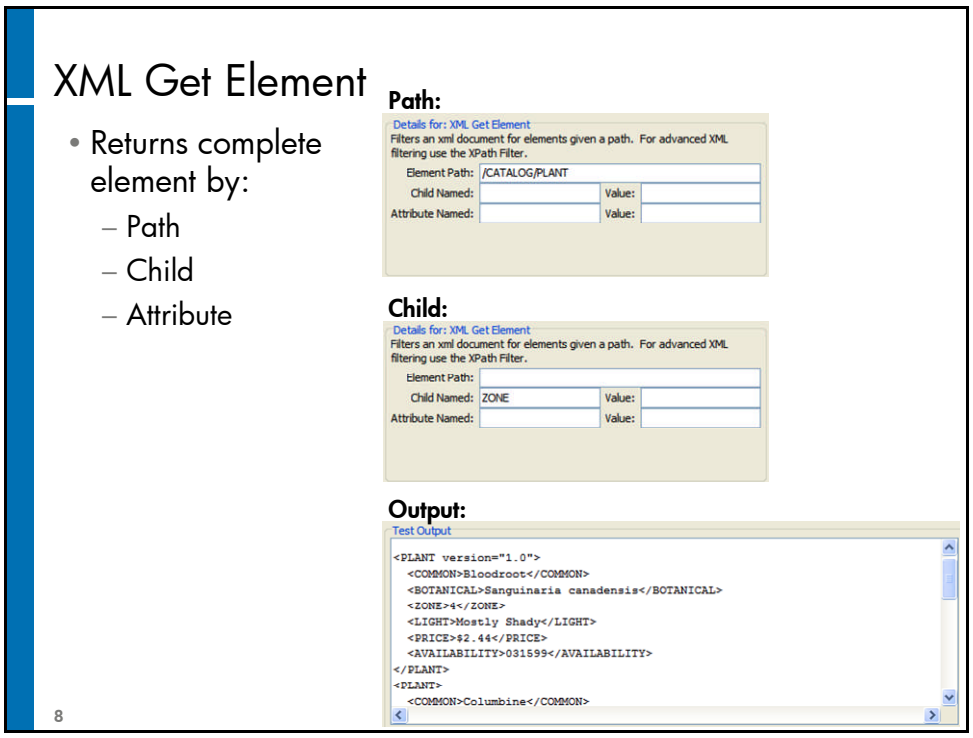



The content in the OO library for dealing with XML was added in version 7.50. In version 9, XML filters were added. Using XML filters is much easier and faster than using the XML content in the OO library. While the XML content in the library can still be used if needed, using the XML filters will probably prove to be a better alternative for many or most applications involving XML.




The first filter, Get Attribute, does just that- it gets an attribute from an XML document named version in the specified path.

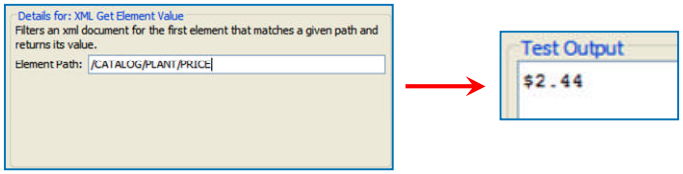
In this example, a sample plant catalog is being queried for an attribute value. The Single Match shows simply the value of that attribute. The As Table match shows the path and the value of the version attribute.




The XML Get Element filter takes either an element path, a named child, or a combination to produce filtered XML output, shown in the Output field on the slide.

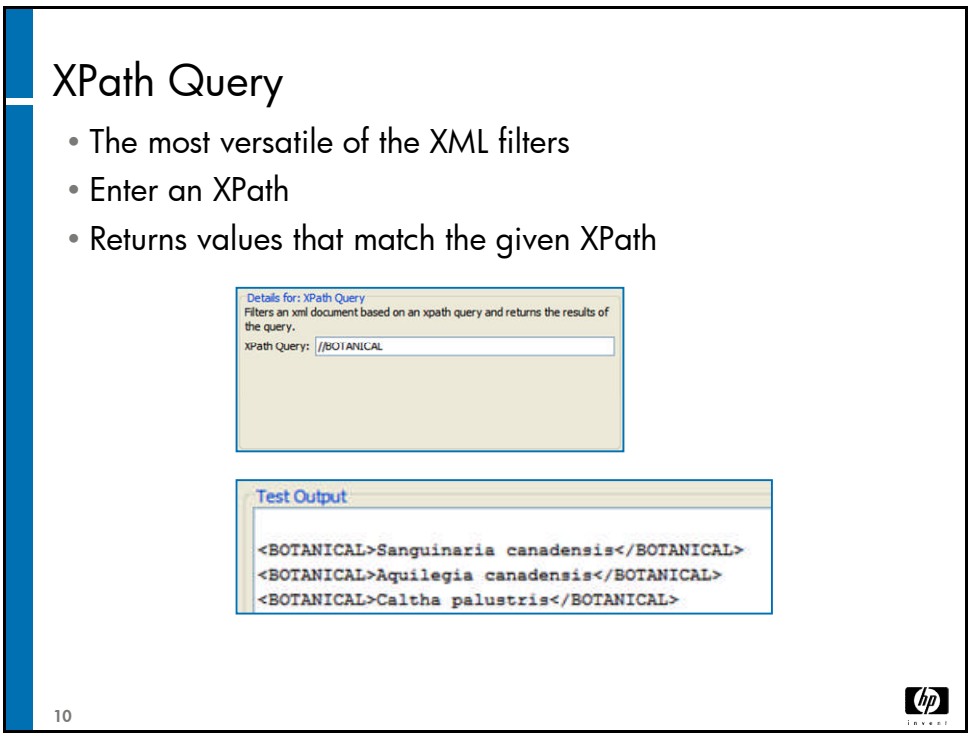
## XML Get Element Value

- Filters for the matching element
- Enter the full or relative path to the element
- Examples:
  - **/CATALOG/PLANT/PRICE** – First matching element value
  - **/CATALOG/PLANT[3]/PRICE** – Third matching element value

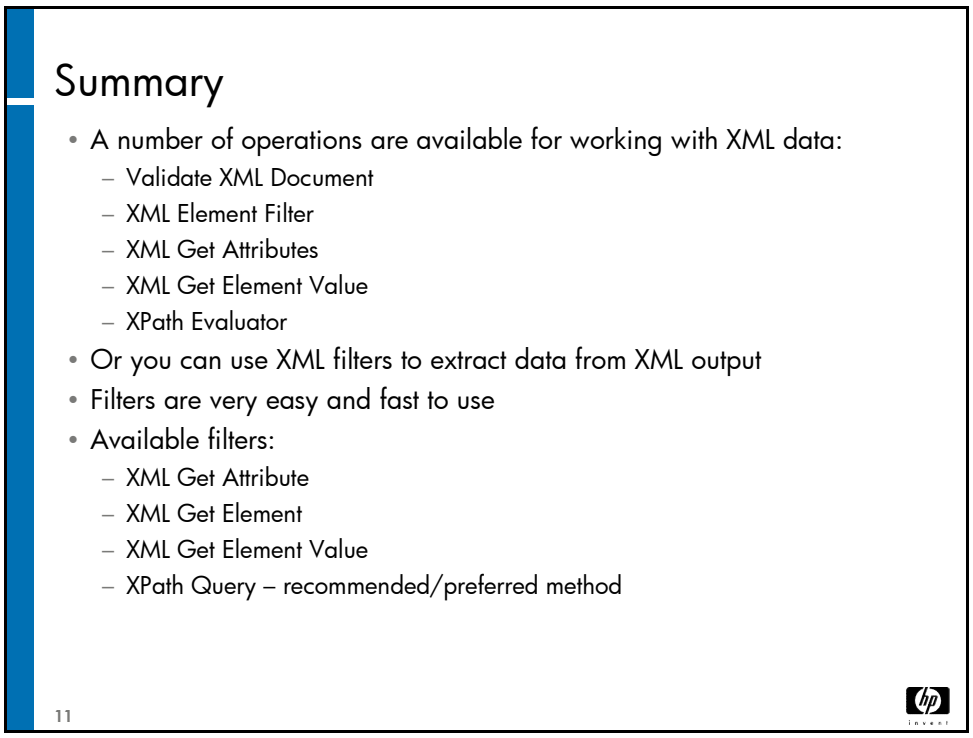



9

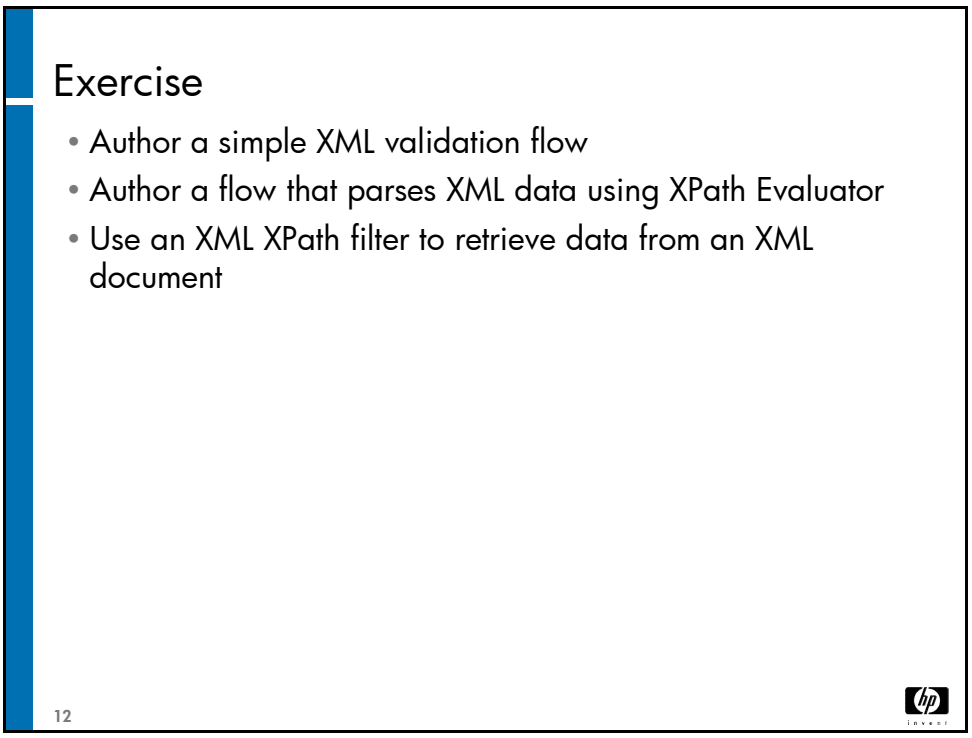
The Get Element Value takes the specified path and provides the value. Review the examples on the slide.




XPath Query is the most flexible of the filters. It allows you to specify an XPath query of any complexity, and returns the values that match the query. As a reminder, you can do a Google search on XPath Query to see many different tutorials on how to construct an XPath. XPath expressions can be very general to very detailed and offer virtually unlimited searching capabilities.

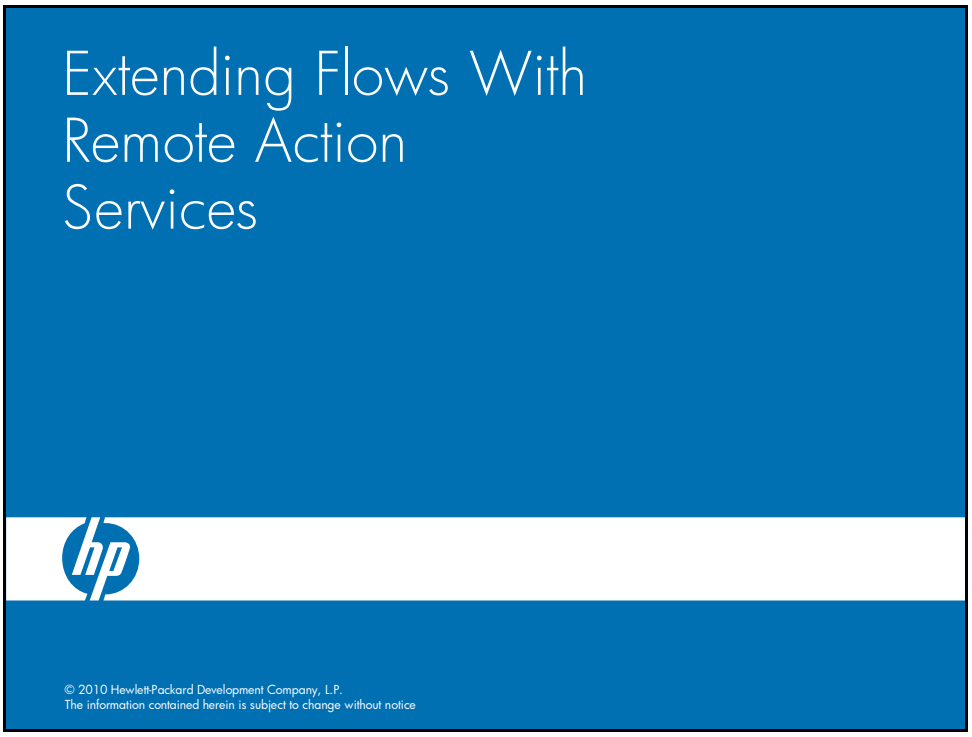



Review the summary. As a reminder, the content in the library for working with XML is very useful but in many or most cases can be replaced by easy-to-use filters applied to flow variables that contain XML output.



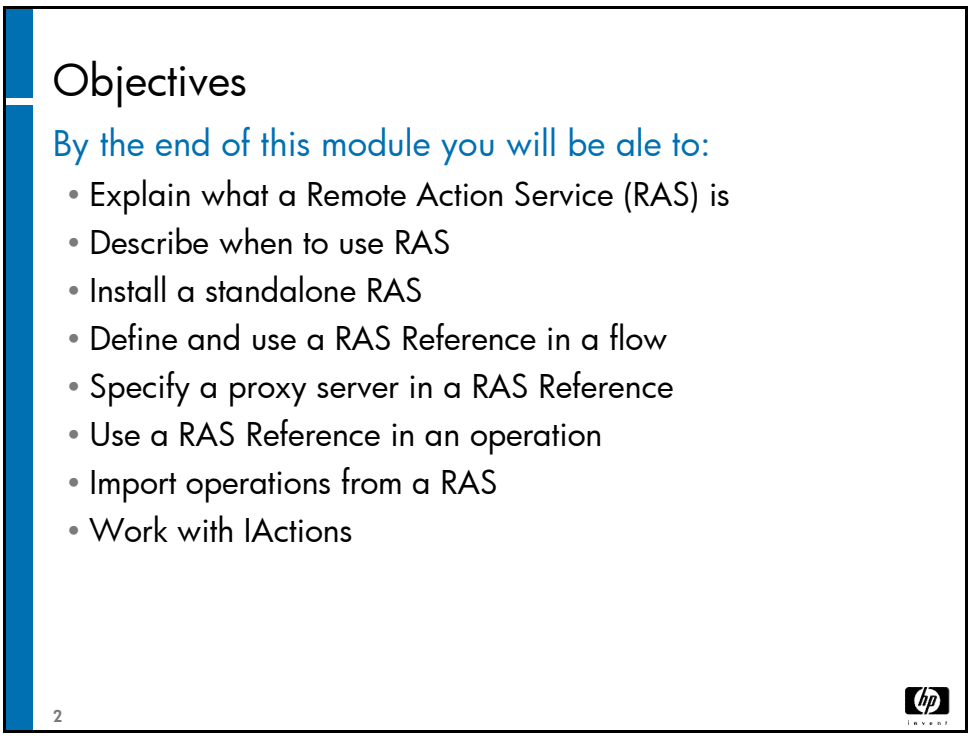

In the exercise you will author a flow that evaluates an XML document supplied by the XML Sample operation. The flow is similar to the one shown on the previous pages.

# Extending Flows With Remote Action Services

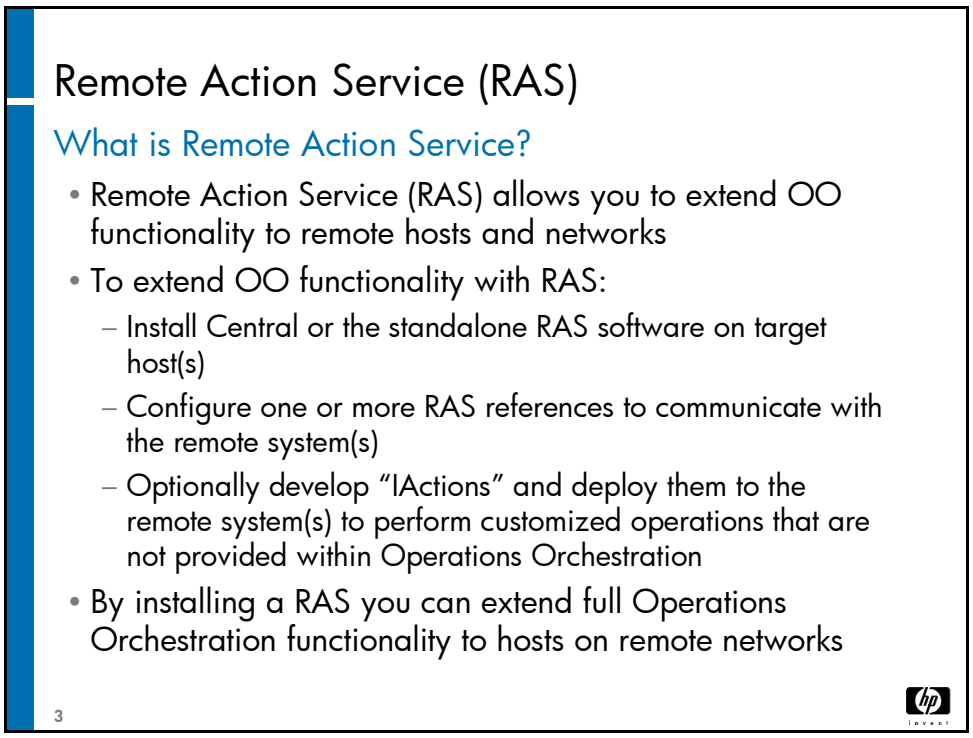



In this module you will examine Remote Action Services.

This is a very large topic and is extremely important to Operations Orchestration. It is important for students to understand RAS, what it is, what it does, and when to use it.




Review the objectives and see if students have any questions before proceeding.




The Remote Action Service (RAS) allows you to extend OO functionality to remote hosts and networks.

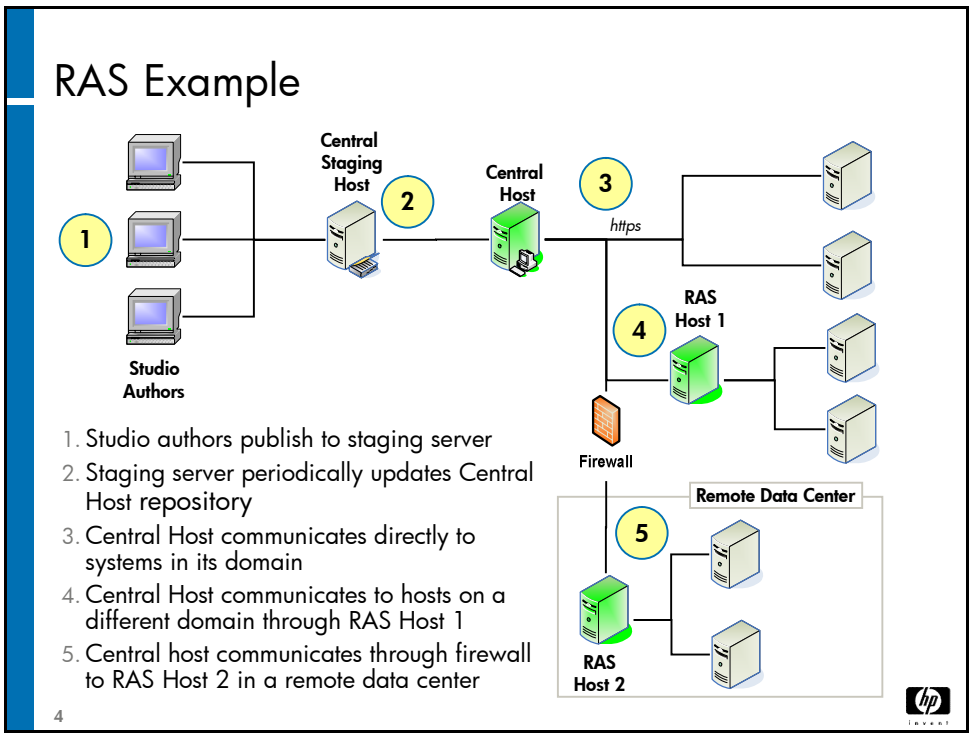
All Central installations have a default RAS. This RAS contains the iActions (Java classes or DLLs) that perform most of the work in Operations Orchestration. In other words, most content in OO is RAS-based.

You can install a standalone RAS to extend OO functionality by making iActions available to remote networks.

You need to install a RAS when your flows need to work with systems in a different Windows domain, are across a firewall, or you want to simply distribute the workload.

You do not need to install a RAS server on every host that will be accessed by your OO flows. Your flows can access all hosts within your Windows domain or are inside your firewall using the single default RAS that installs with Central.

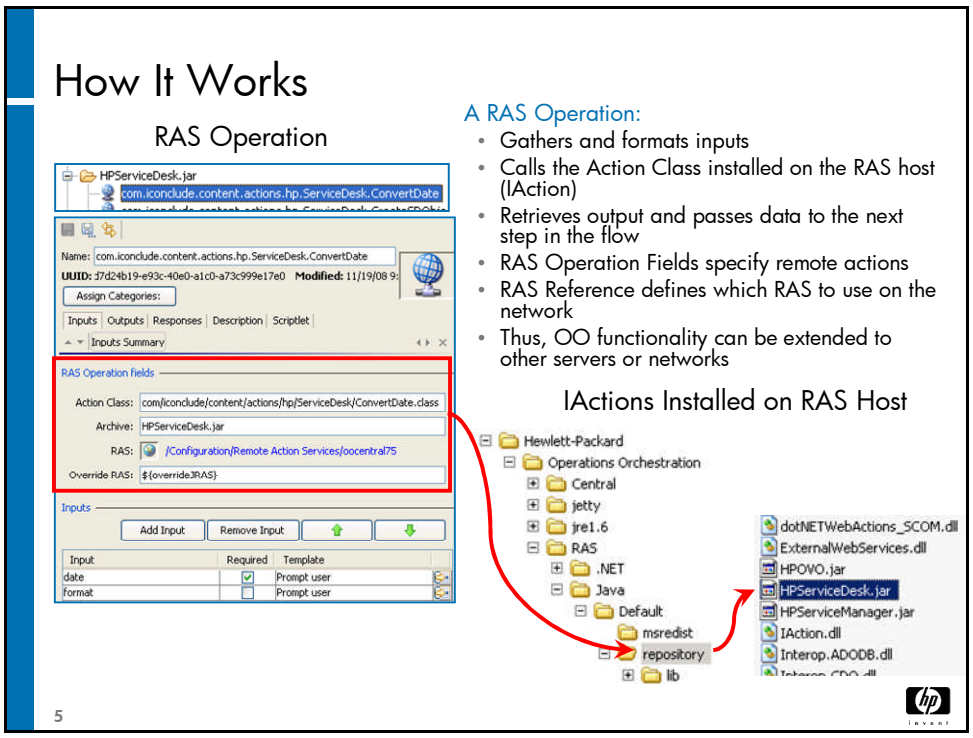
The ability to install standalone remote RAS servers is a very powerful tool that allows you to extend functionality to remote networks.




The slide shows a graphical representation of RAS hosts installed in a typical network topology.

1. In this example, studio authors communicate to…
2. a Central staging server which communicates to…
3. a Central production server, which communicates directly to hosts in its domain or…
4. to remotes using RAS Host 1.
5. A second RAS Host 2 communicates to the hosts that are across the firewall.

The RAS hosts contain the iAction Java Classes or DLLs that your flow content uses to accomplish tasks on the remote hosts.




This slide ties a RAS reference in a RAS operation to the Java class that performs a task.

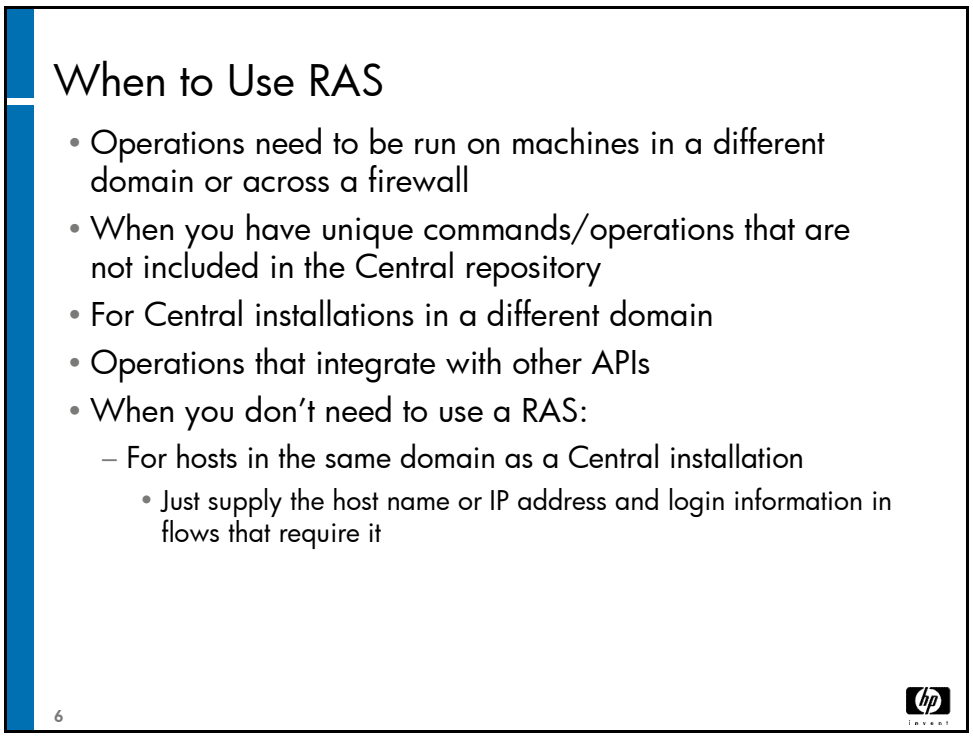
In this example, the RAS operation gathers and formats inputs.

It then calls the Action Class (iAction) installed on the RAS host.

It retrieves output and passes data to the next step in the flow.

The RAS Operation Fields specify the remote actions by referring to the Archive (JAR file) and Action Class (Java Class).

The Override RAS allows authors to programmatically refer to a different RAS at runtime.




The slide explains when to use a RAS and when one is not needed. Just review this slide.

## Installing RAS

- RAS installs with Central
- You can also install a standalone RAS on remote hosts to extend OO functionality to that domain without installing Central

### To install a RAS

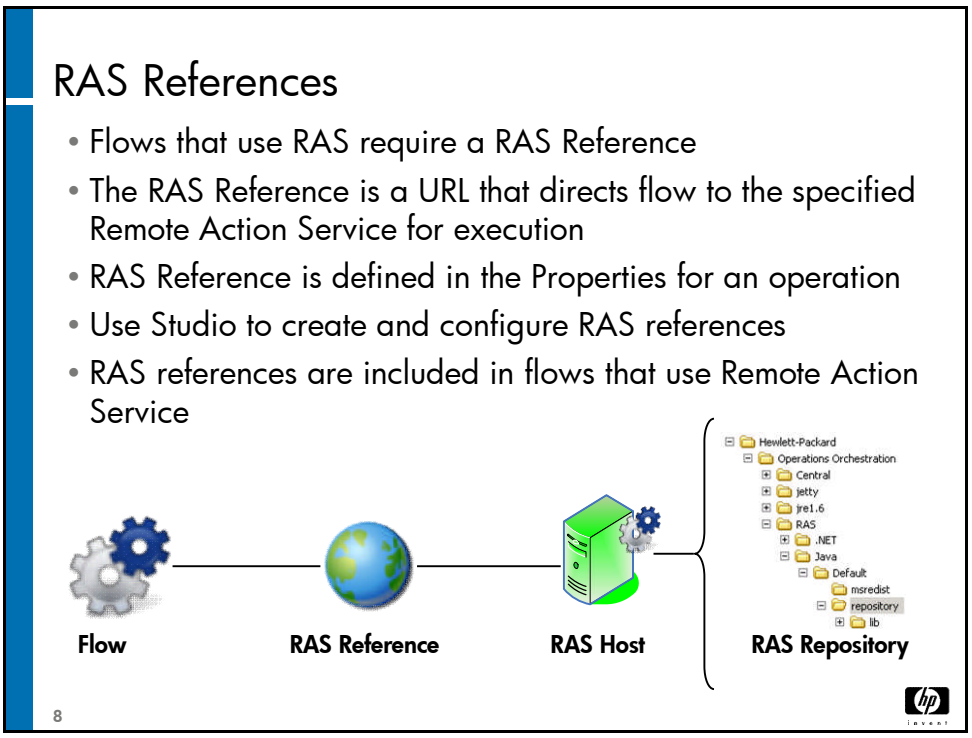
- Locate the RASInstaller on the OO installation media (Linux/Windows)
- Run the installer, provide inputs


7

OO has a standalone RAS installer that allows you to install the RAS server on a remote host.

Just locate the installer on the OO installation media for Windows or Linux and run the installer.

It is important to note that the Linux-based RAS cannot execute .NET (NRAS) content, which is DLL based and runs only windows. Therefore if you need to access a RAS on a Linux host you can only use JRAS (Java Class-based).




You refer to a RAS with a RAS Reference.

The RAS Reference is a URL that points the flow to the specific iAction hosted on the RAS server.

The RAS reference is defined in the properties for an operation that uses RAS, which is most of the content in OO. In other words, you need to double-click the operation in the Library or select it in the Design panel and choose Open Operation to see the Operation Properties, which is where the RAS Reference is specified.

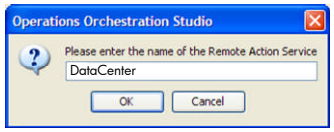
## Creating a RAS Reference

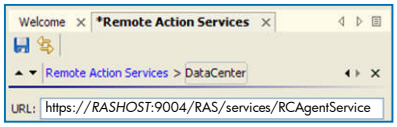
### A RAS Reference consists of:

- Hostname or IP address of the RAS host
- Port number (default: 9004)
- Location of the RAS service:
  - RAS/services/RCAgentService
- Default RAS Reference: RAS_Operator_Path

### To add a RAS Reference

1. Right-click Configuration/Remote Action Service, select New
2. Give the reference a descriptive name, click OK
3. Enter the URL of the RAS




9

RAS References are stored in the Configuration/Remote Action Service area folder in the Library.

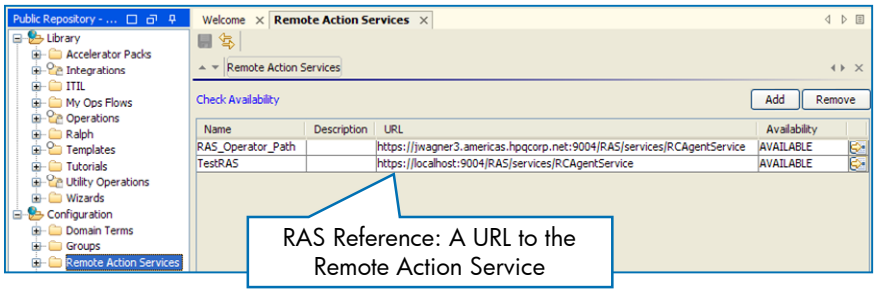
It consists of the hostname or IP address of the RAS host, port number (the default is 9004), and the location of the RAS service which is always RAS/services/RCAgentService.

All Central installations have a default RAS reference named RAS_Operator_Path.

To add a new RAS reference, right-click on the Configuration/Remoate Action Service folder, select New, name the RAS Reference, and provide the URL of the RAS.

## Monitoring RAS References

- OO maintains a RAS list
- To view available RAS references:
  1. Open Library/Configuration, expand Remote Action Services
  2. Default RAS reference: RAS_Operator_Path
  3. You can add or remove additional RAS references
  4. Click Check Availability to update the Availability list




10

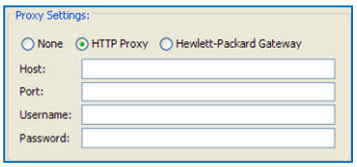
Since RASes are often installed on remote hosts, you need to monitor whether the RAS is available. The host may be down for system maintenance, for example.

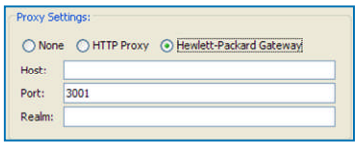
The Remote Action Services list tells you whether a RAS is available or not.

To update the list, click Check Availability.

## Specifying a Proxy Server in a RAS Reference

- If you need to communicate across a firewall or are interoperating with other BSA products, specify a proxy server in your RAS reference
- Select the HTTP Proxy or Hewlett-Packard Gateway button when configuring the RAS Reference




### Proxy Server

- Specify Host, Port, and (if needed) Username and Password

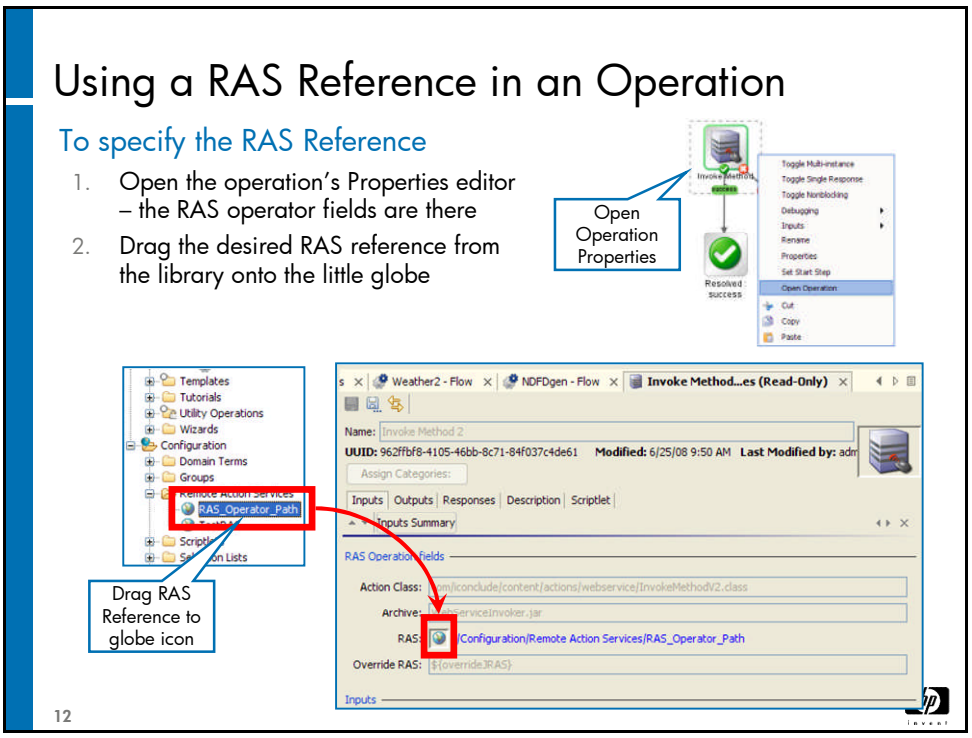
### Hewlett-Packard Gateway

- A mechanism for configuring RAS for HP Server Automation using the SA Gateway
- Contact the SA administrator for details

11

To communicate across a firewall you need to specify a proxy server. Typically you need to provide a proxy host and port numbers, and sometimes a username and password.

The Hewlett-Packard Gateway settings are intended for use with HP Server Automation using the SA Gateway. Contact the SA administrator for details.




Using a RAS Reference in an operation is very simple. Just drag the RAS reference from the library onto the RAS icon in the operation's Properties window.

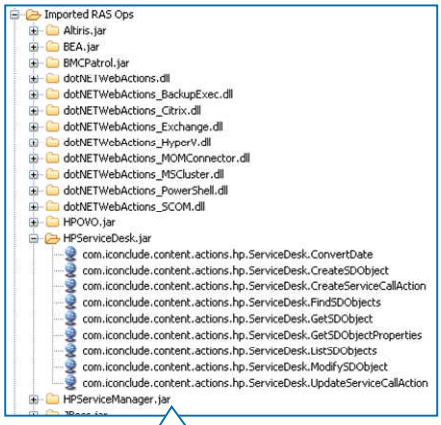
In other words, you need to set up the RAS Reference first, then you can drag and drop it onto an operation's Properties window.

## Creating Operations From a RAS

- You can import all of the operations in a RAS into your local Studio library
- This is useful for:
  - Building flows that use individual RAS operations
  - Reviewing the available operations in a RAS

To import operations from a RAS:

1. Create a new folder in the Library and select it
2. In the File menu select Create Operations from RAS
3. Select the RAS Reference from the pull-down menu
4. Browse imported RAS operations in the Library




- DLLs and JARs in the RAS are imported as folders
- To access individual operations, browse folders or Search

13

You can import all of the operations for a RAS into your local Studio library. This is useful for a number of purposes. For example, you can use the imported operations in flows, or review the functionality that is available in the RAS.

To import a RAS into your library, create a new folder and select it, then in the File menu select Create Operations from RAS.

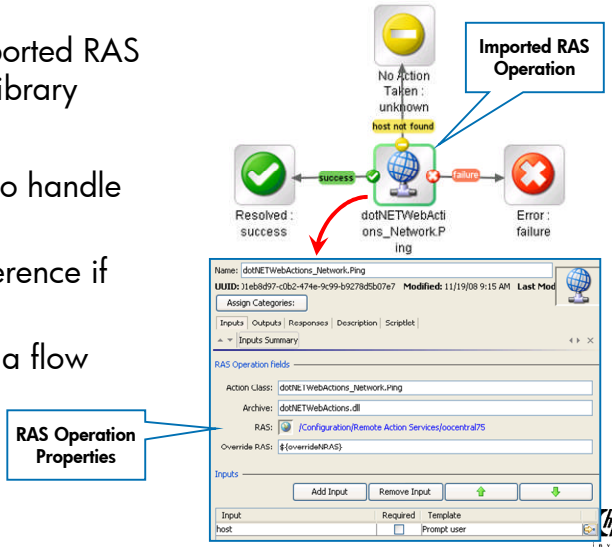
This process takes a few minutes so be patient.

You can then use the imported content in your flows.

## Using Imported RAS Operations in a Flow

- Create a flow folder and new flow
- Copy/paste imported RAS operation from library
- Provide inputs
- Determine how to handle outputs
- Update RAS reference if necessary
- Incorporate into a flow

Example: RAS Ping using imported RAS operation



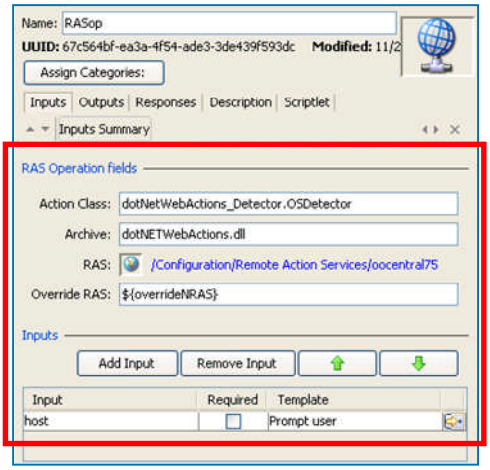

14

The slide outlines how to use imported RAS content in a flow.

It's a good practice to copy an imported operation and paste it into your working flow, then build it into your flow. You definitely would need to do this if you made any changes to the operation's properties.

## Creating a RAS Operation From Scratch

1. Right click in a flow folder, select New/Operation/RAS Operation
2. Select a RAS reference
3. Specify RAS properties:
   - Action Class: Action class from which the operation is created
   - Archive: JAR) file or DLL) file in that contains the action class
   - RAS Reference: The RAS you want to use in Configuration/Remote Action Services – drag/drop
   - Override RAS: Always use the value as shown
4. Specify inputs, outputs, responses
5. Incorporate into a flow




15

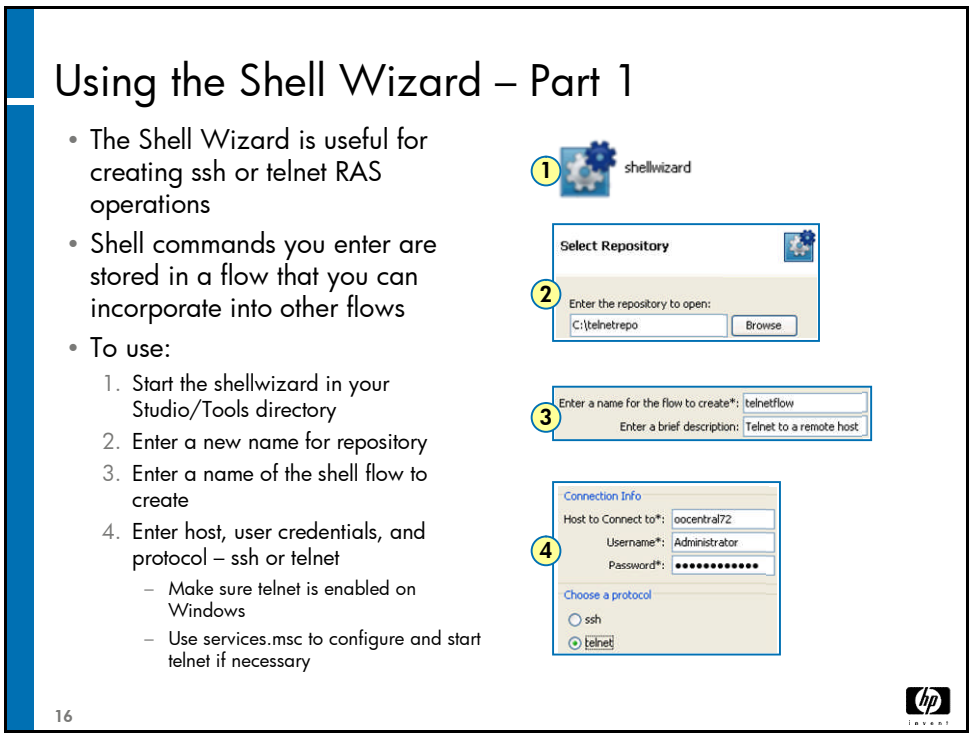
You can also create a RAS operation from scratch.

Create a new operation and select RAS Operation as the type.

A RAS operation requires an iAction that is deployed on the RAS server to be useful. Therefore the key part of creating the RAS operation is to point it to the Java class or DLL that exists on the RAS server.

The RAS Operation UI has fields for specifying the exact location of the iAction that the RAS operation uses.

Once you have set up the RAS operation you can incorporate it into a flow.

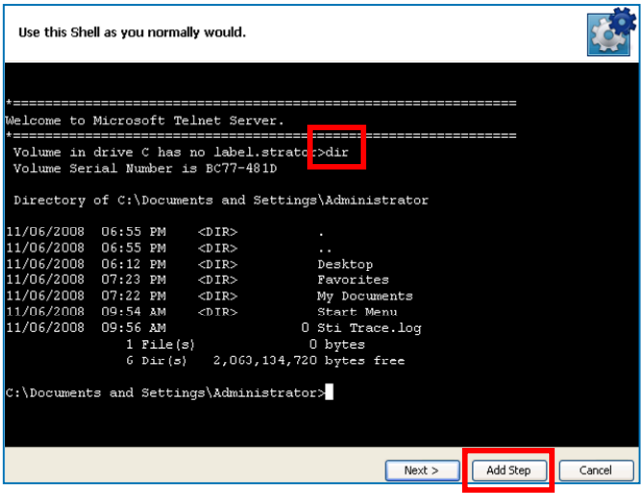



You can use the Shell Wizard to create a RAS operation that executes a telnet or ssh command on a remote host.

Just follow the steps as shown on the slide.

## Using the Shell Wizard – Part 2

5. Shell window opens - enter commands to execute on the remote host, click Add Step after each command - Each step is stored in the flow




17

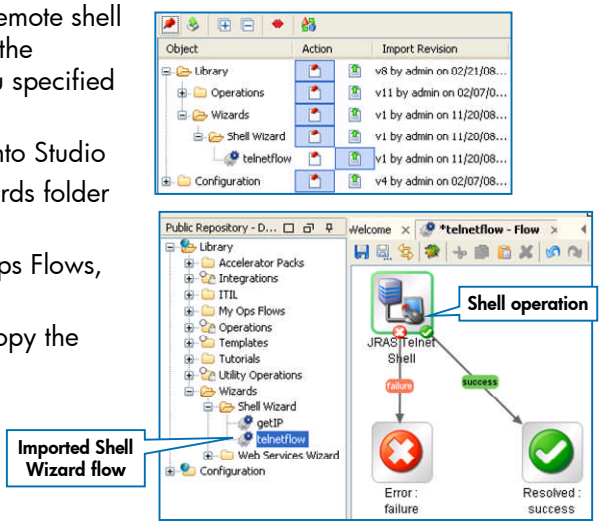
The slide shows the interface when you set up a Telnet operation in the Shell Wizard.

You get a Telnet window where you can add commands. Clicking the Add Step button adds each specified action to the operation, so it is executed automatically when the operation runs in a flow.

This is very useful for running command line-based tasks in flows on remote hosts.

## Using the Shell Wizard – Part 3

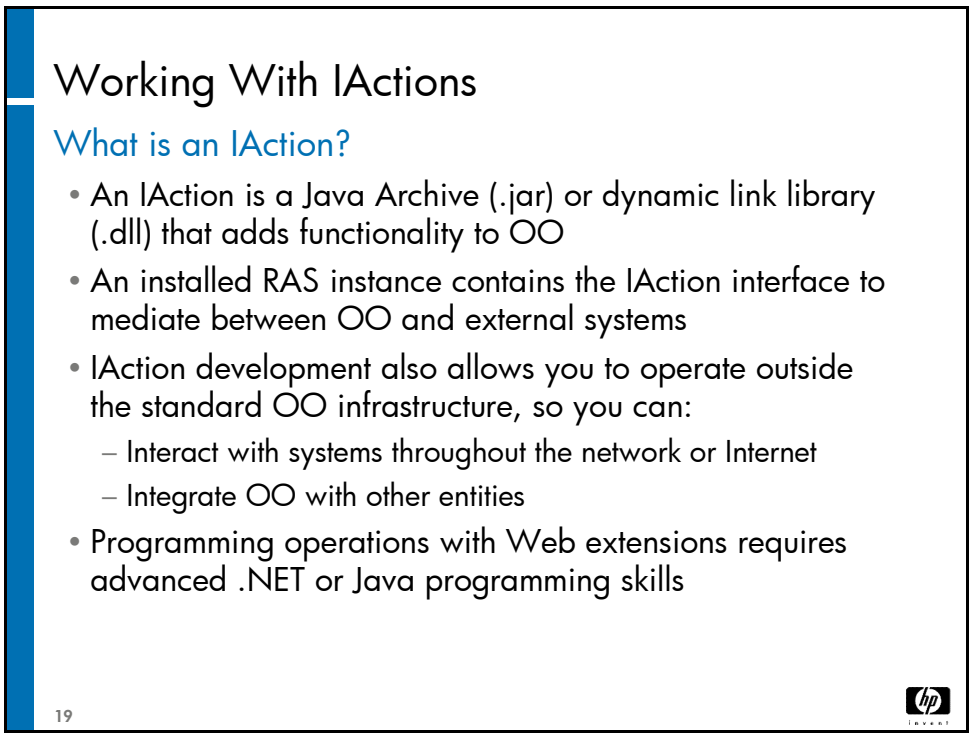
- A flow containing the remote shell operation is created in the repository directory you specified earlier
- Import that repository into Studio
- Flow imports into Wizards folder in the Library
- Copy/paste into My Ops Flows, modify as needed
- Use imported flow or copy the shell operation




18

A flow containing the shell operation is created in the repository directory you specified when you started the Shell Wizard.

When the wizard completes, you import the repo into your library and the new shell content is available for use in your flows.

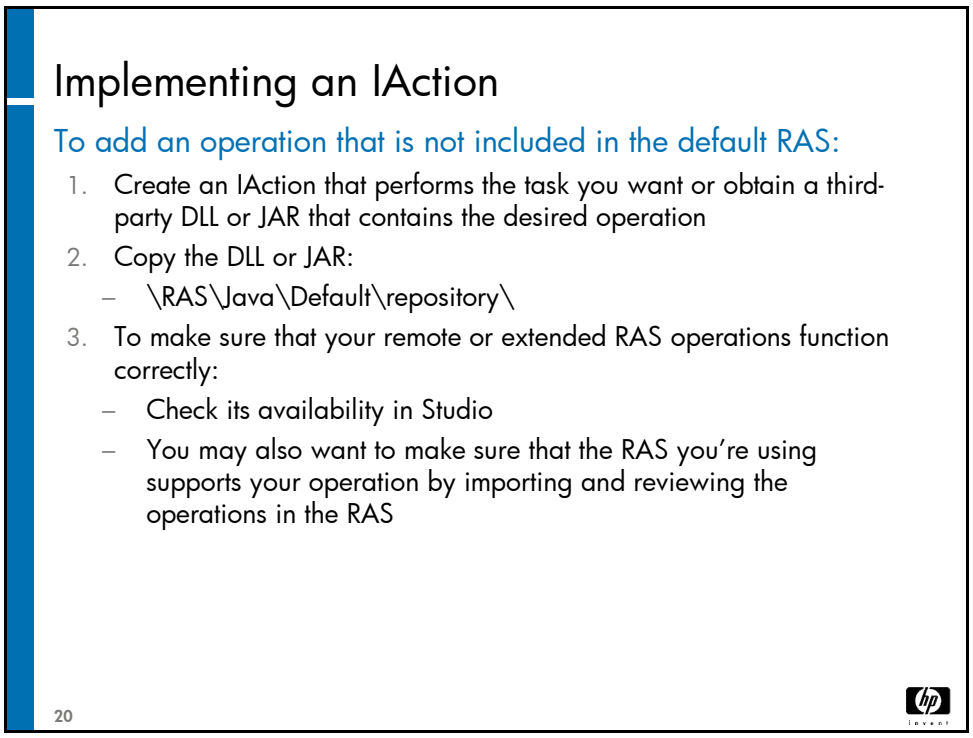



An iAction is a Java Archive (.jar) or Dynamic Link Library (.dll) that adds functionality to OO.

RAS basically handles the communication between operations and the iActions that they execute on remote hosts.

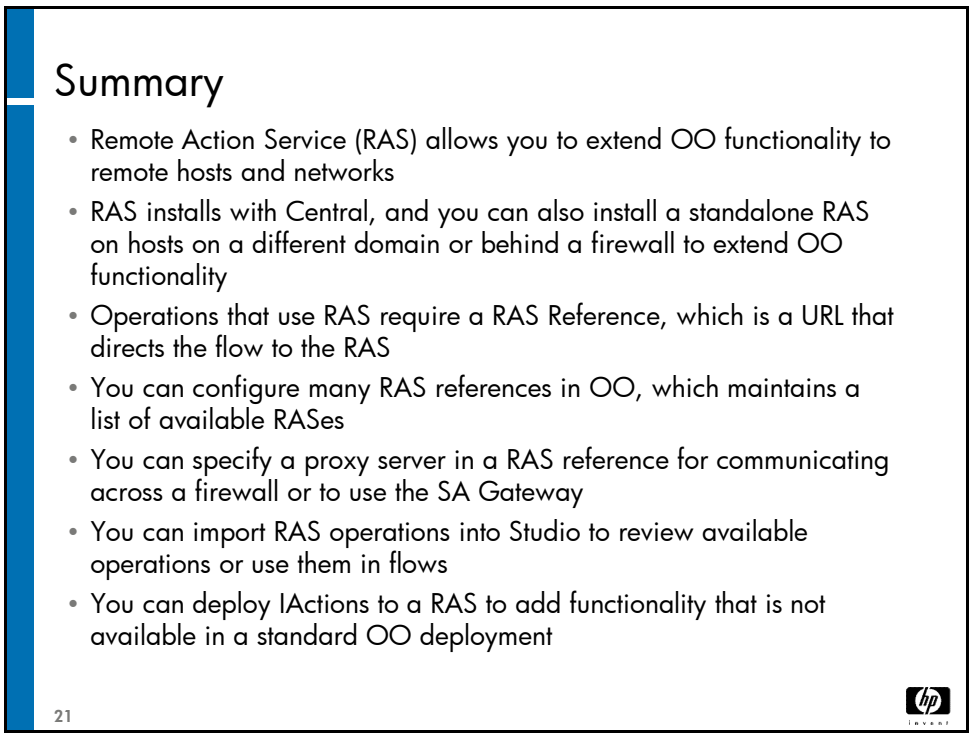
iAction development involves Java or .NET programming techniques and is out of scope for this class.

Refer to the OO SDK for detailed information on developing iActions that can be used in RAS operations.

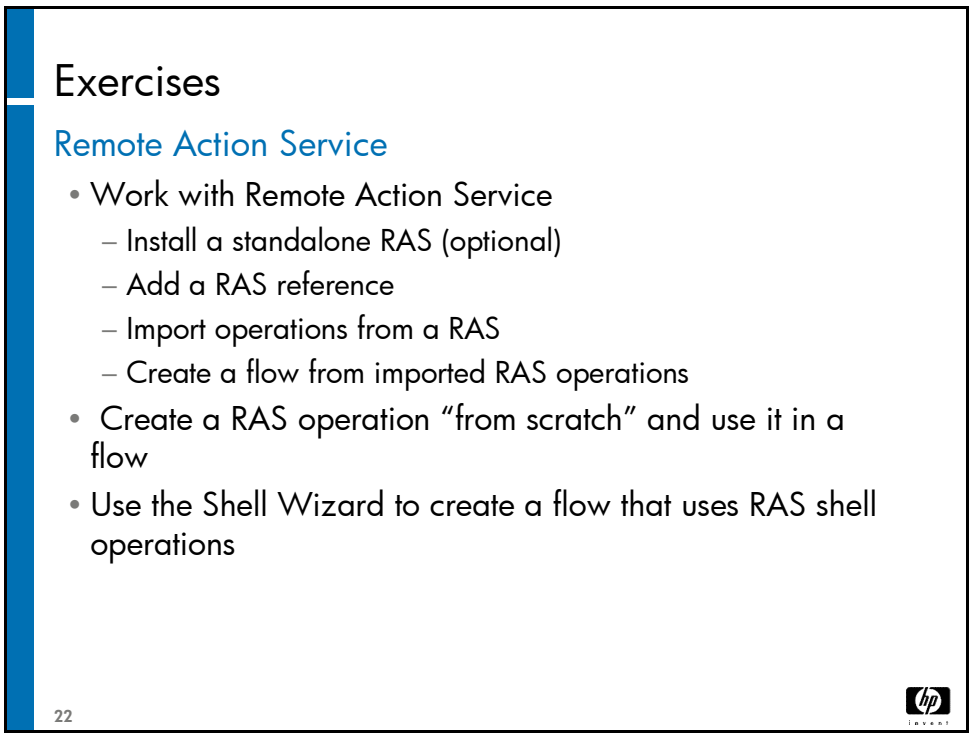



To implement an iAction, you copy the DLL or JAR file to the \RAS\Java\Default\repository directory on the Remote Action Server.

Once deployed you create a RAS operation that points to the content.

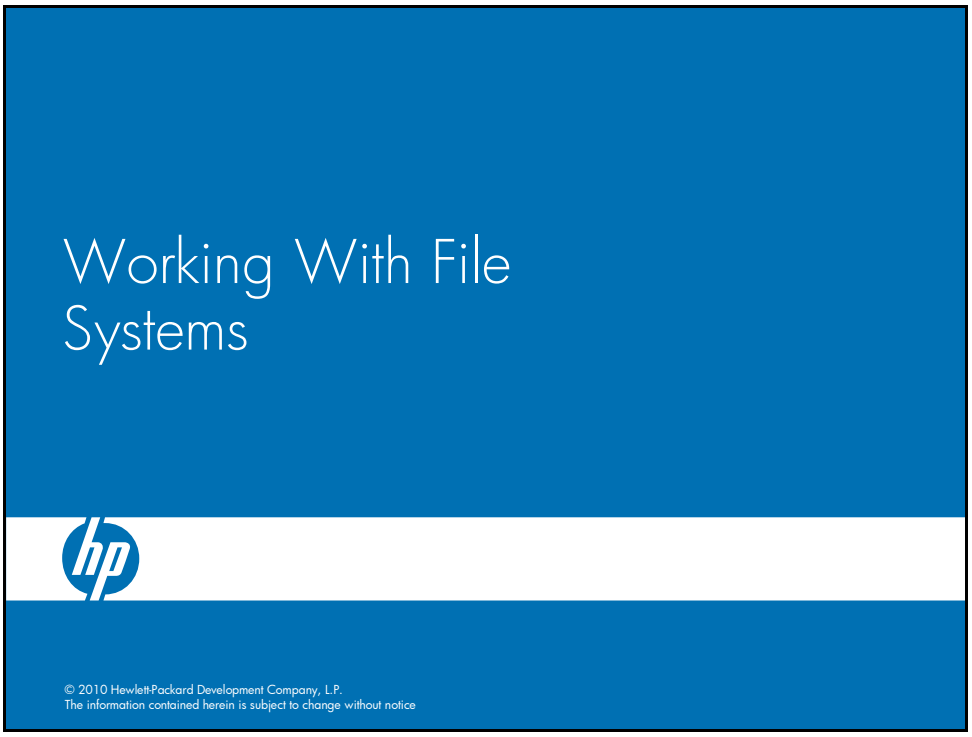



Review the summary.

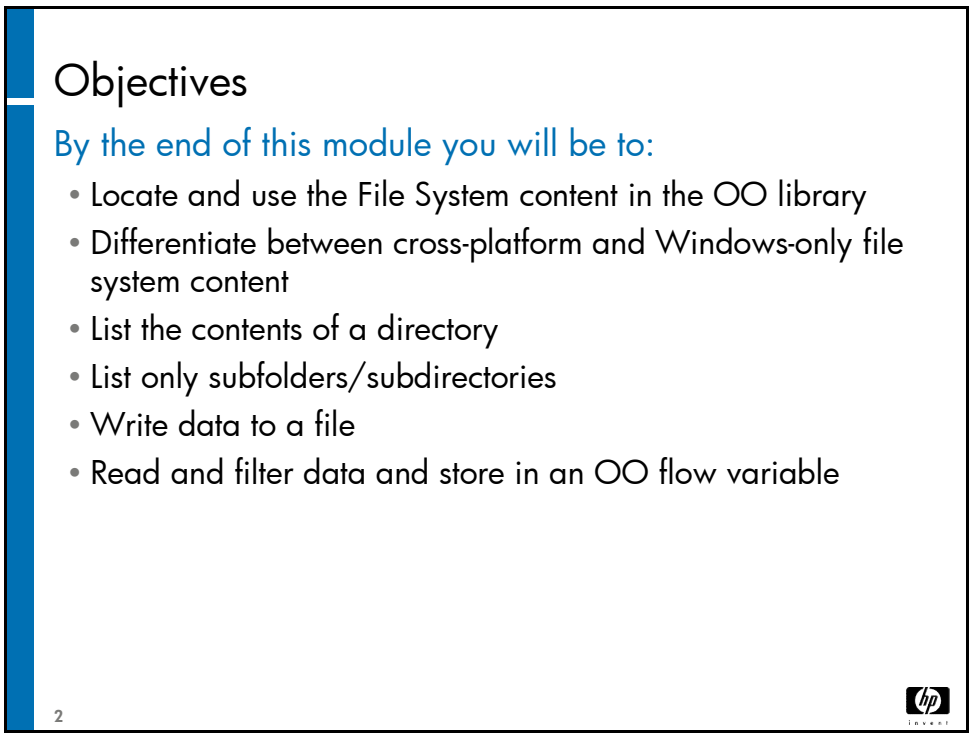



In the exercise for this module you will work with Remote Action Service, as shown on the slide.

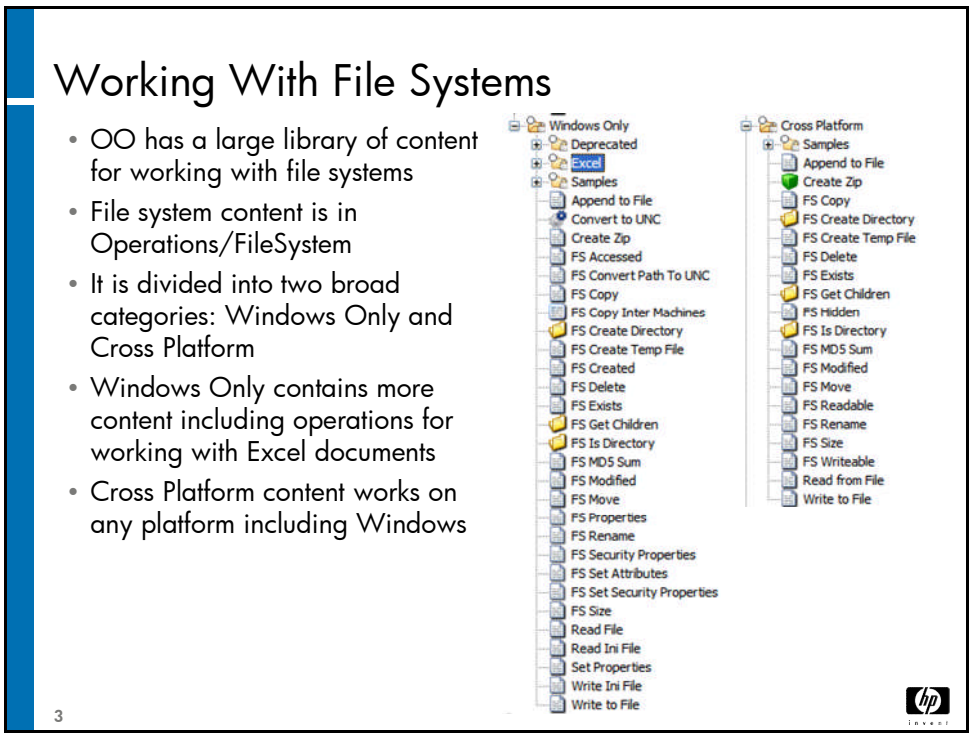
# Working With File Systems




In this section you will learn how to use content in the OO library for working with file systems.




Review the objectives – basically you will learn how to read, write, and list external file systems with content in the OO library.




Operations Orchestration has a large amount of content in the library for dealing with file systems. Just browsing the names of the operations tells you a lot about what they do – most of these are basically self-explanatory.

The important thing to note is the presence of the Windows Only and Cross Platform folders.

The Windows Only folder contains more content, with many operations for working specifically with Windows files like properties, Ini files, etc.

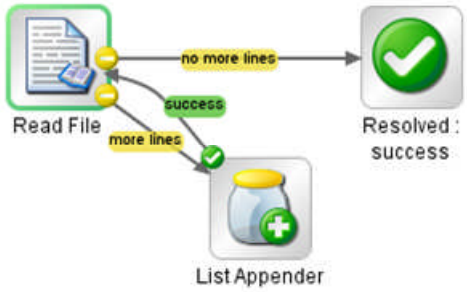
Additionally the Windows Only content supports windows authentication.

Many users prefer to use the Cross Platform content when possible because it is simpler to use, as fewer inputs, and bypasses the Windows authentication. Obviously you would use what ever content category is appropriate for your application, but generally speaking the Cross Platform content is a little easier to implement.

It's a good idea to look at the Samples folder for each platform category.

## Reading a File

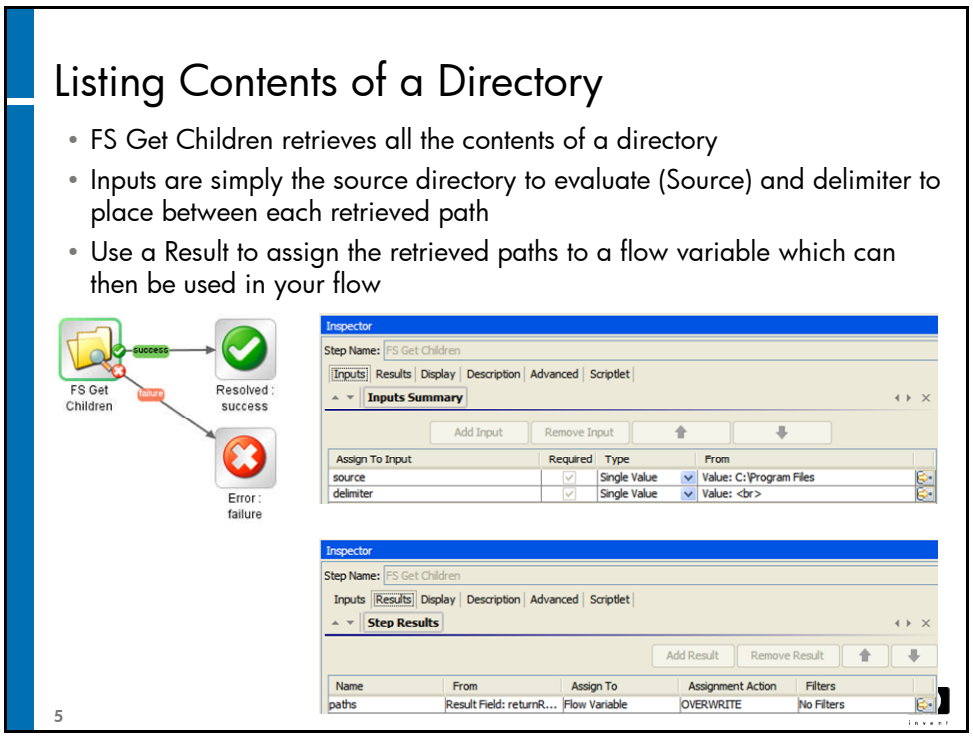
- Read File iteratively reads a file line-by-line
- As an iterative operation you need to tie the more lines response to some other step – List Appender in this case
- Read File also has a Filter allowing you to include only those lines that contain the filter you specify




4

This example shows one of the simplest tasks, reading from a file.

The Read File operation iteratively reads each line of the file. To make the entire file availbale in a flow variable, you can simply append each line in List Appender. Optionally you can set filters in Read File to append only those lines that match the filtering criteria.



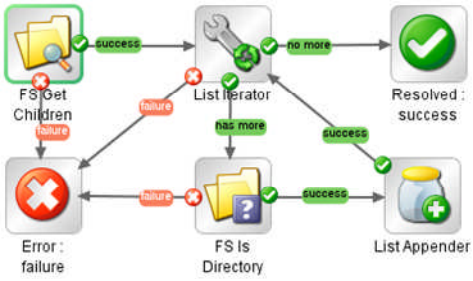

The FS Get Children operation lists the contents of a directory. This is not an iterative operation, it simply provides a directory listing.

You simply specify a source directory to list, and a delimiter to place between each item retrieved in the listing.

Use a result to assign the retrieved paths to a flow variable. The named result should be based on the Result Field returnResult.

## List Only Subfolders of a Directory

- Another useful operation is FS Is Directory which acts as a filter which passes only paths that are directories
- The List Directory Subfolders flow:
  - Gets all paths and stores in a result named paths
  - Uses List Iterator to iterate through the list
  - Each item in the list is sent to FS Is Directory
  - If the path is a directory it is added to the list, otherwise it is ignored
  - The result is a listing of subdirectories in a parent directory




6

This example shows the use if FS Is Directory, which essentially acts as a filter that passes only those items that are directories. In the example above, FS Get Children compiles the list and the List Iterator goes through the list one item at a time, filtering each entry through FS Is Directory which passes only those items that are directories to List Appender, which compiles the list.

In other words, this simple flow returns a list of directories rather than the entire contents of a source directory.

## Writing to a File

- The Write List to File flow writes the compiled list of directories to a file
- The cross-platform version of Write to File does not require username/password, the Windows version does

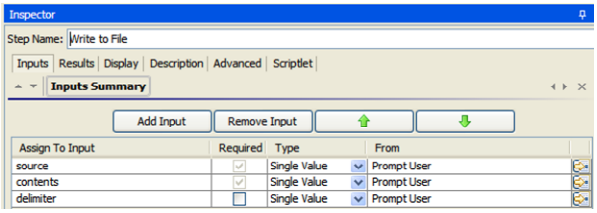

7

This example shows the directory listing flow from the previous slide but with an added step to write the list to a file. Writing to a file is very simple, you simply specify the full path to the file name you want to write and provide the flow variable content to write to the file, in this case the compiled list of directories.

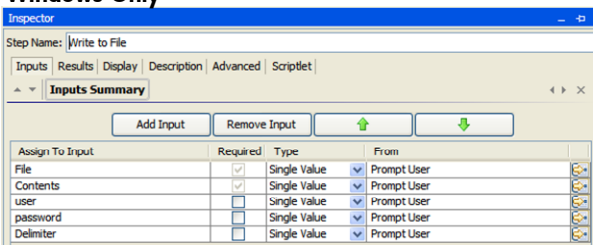
## Write to File Inputs

- Source: Full pathname of the file to write
- Contents: data to write to file
- Delimiter: Newline (\n, \r\n, etc)
- User/password: Windows account (Windows only)

**Cross Platform**




**Windows Only**




8

This slide shows the differences between the inputs for the Cross Platform and Windows Only versions of Write to File.

## HTTP Get to File

- Other content in the OO library writes to disk
- HTTP Client Get to File does an HTTP Get operation and writes the results to disk

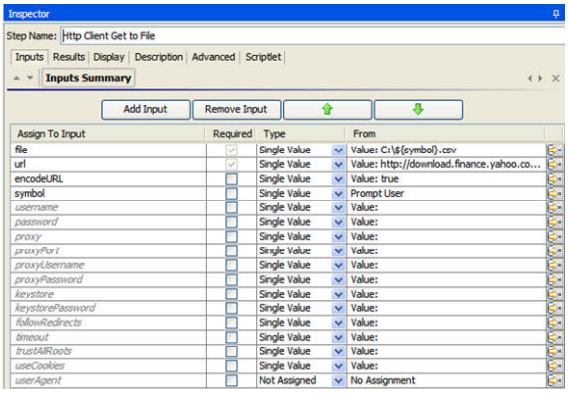

9

In this example, you are using an HTTP Get To File to perform an HTTP Get operation and write the result to a file. In the lab exercise you will use this flow to retrieve a stock quote from Yahoo Finance and write it to a file. You will need a clear Internet connection for this flow to work.

# Inputs

Example: Get a stock quote, write to file

- File: File name to write – C:\${symbol}.csv
- URL: Location to retrieve data
- Encode URL: true
- Symbol: Prompts user to enter a stock symbol
- Unused inputs are removed

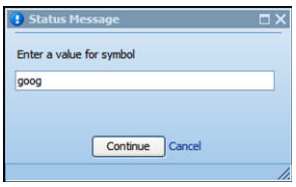


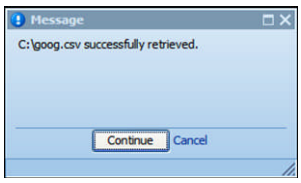

10

Here are the inputs required to get the stock quote with HTTP Client Get to File – the only inputs used are the file name to write (${symbol}.csv), the URL to retrive today's quote for ${symbol}, the symbol input to retrieve, and the encoding option (set to true). Unused inputs are simply removed.

## Running the Flow

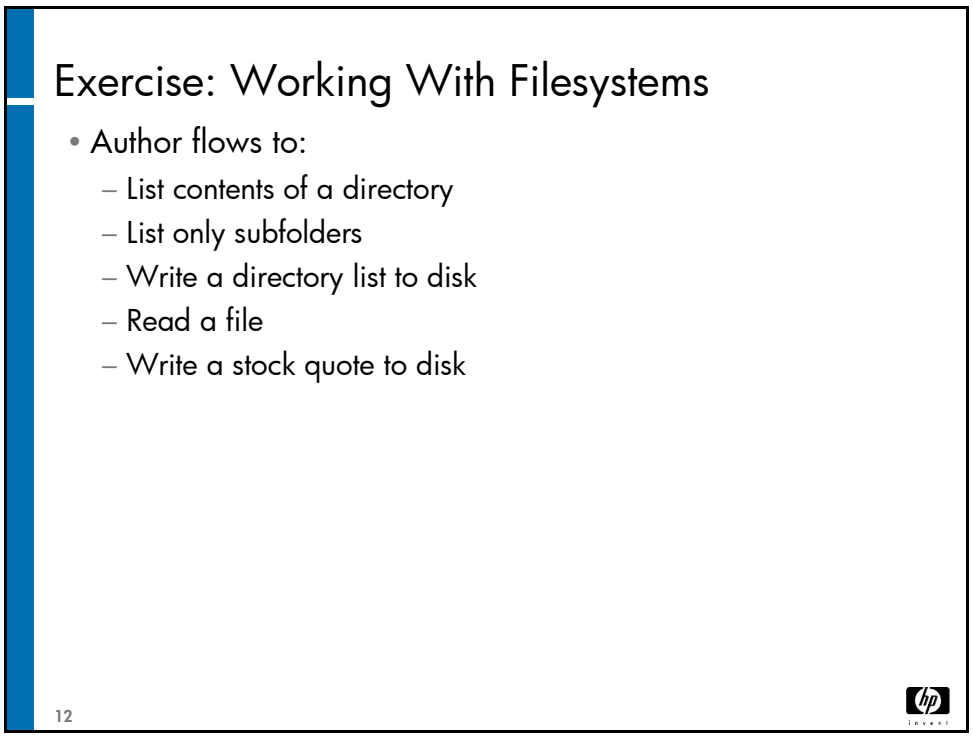
- Enter a stock symbol – stored in ${symbol}
- HTTP Client Get to File does an HTTP Get on this URL:
  - http://download.finance.yahoo.com/d/quotes.csv?s=**${symbol}**&f=sl1d1t1c1ohgv&e=.csv
- File is saved to C:\${symbol}.csv



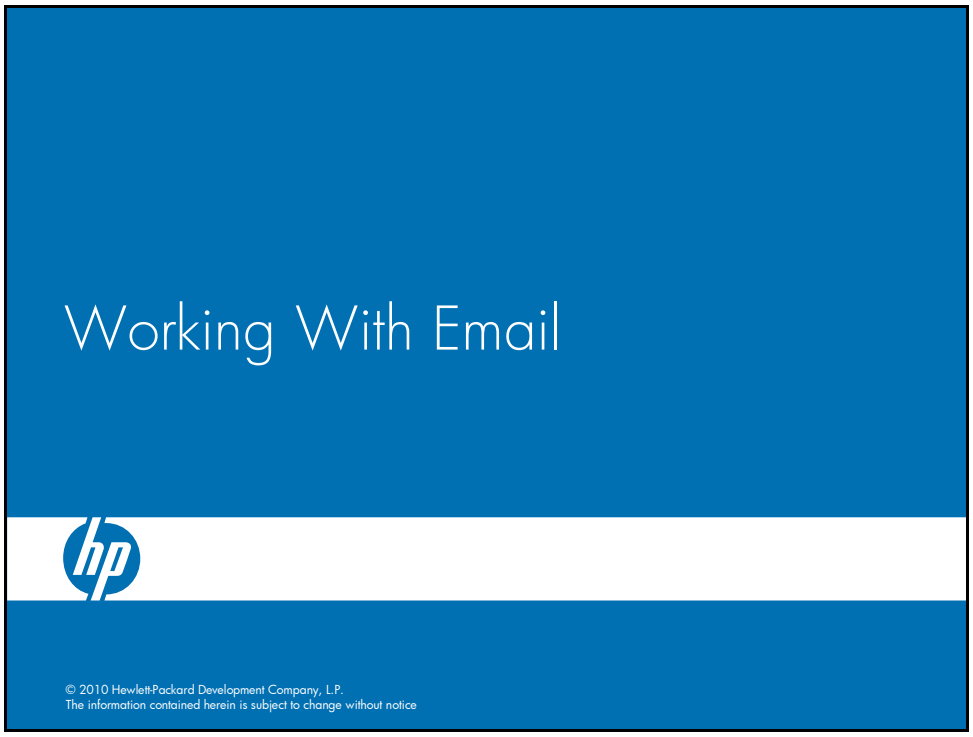

11

When you run the flow you enter a stock symbol, and the flow gets the stock quote and writes it to a file named symbol.csv in the location you specify.

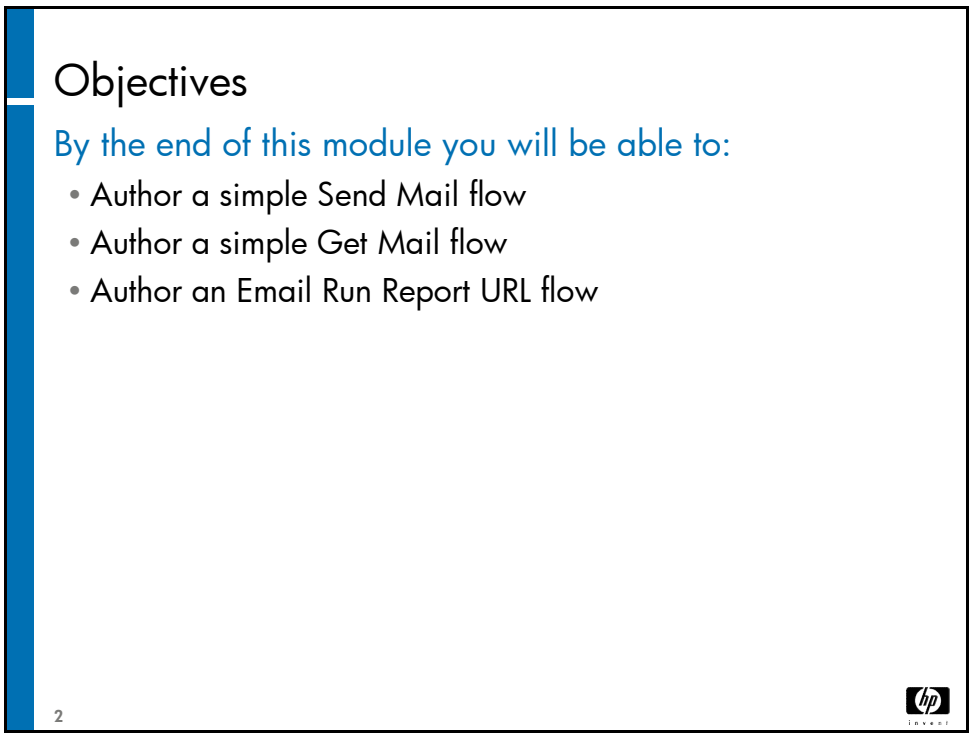



In the exercise you will write a number of flows that deal with file systems.

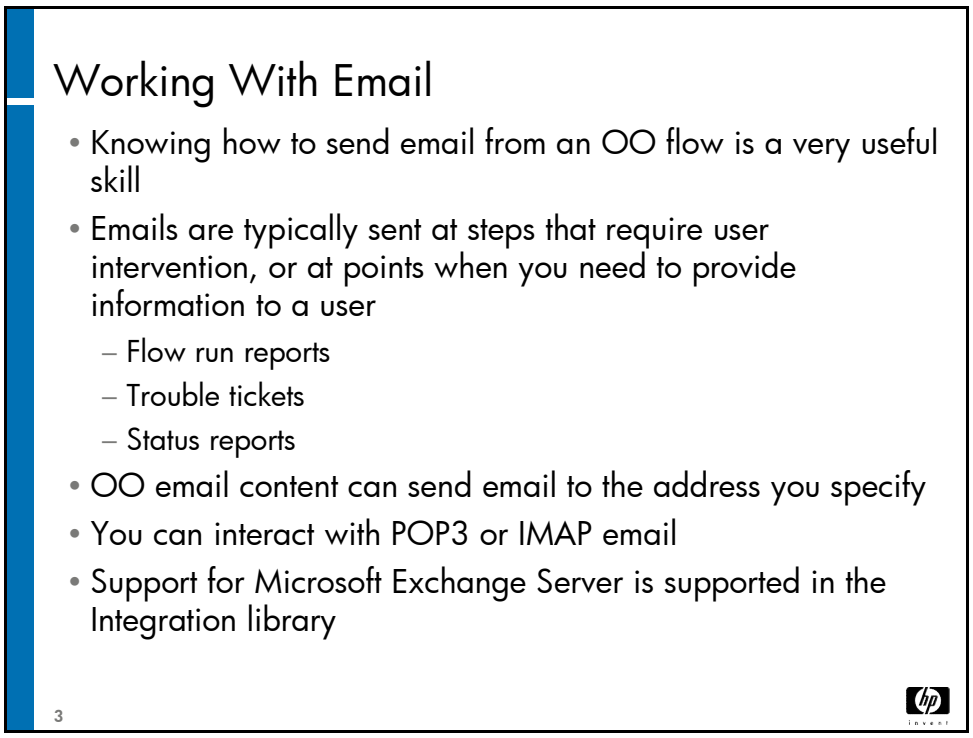
# Working With Email




This section covers another simple but very useful tool, sending emails from your flows, including details on the flow run itself.

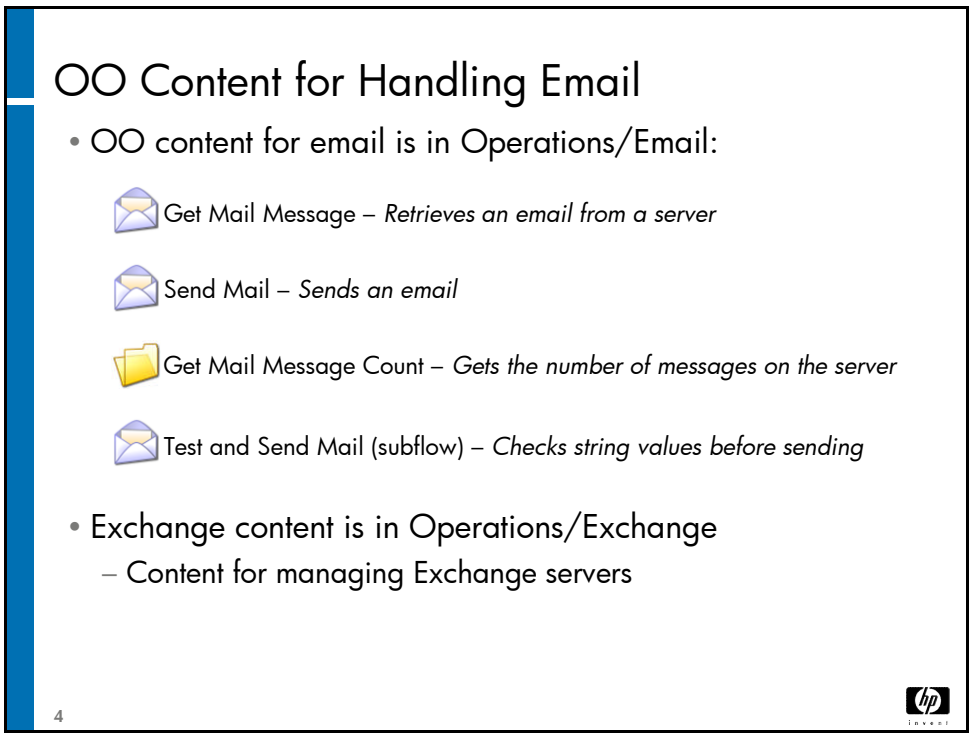



In this section and the accompanying exercise you will author the flows shown on the slide.




Working with email is a very important skill for flow authors. Using email allows you to notify users of a particular condition at any point in a flow run. Email can be sent to anyone you specify, and support for Microsoft Exchange Server is supported in the Integration library.

For the exercises in this section you will need access to a mail server. A link to a free mail server is provided in the lab guide for this section, along with installation and configuration instructions. If you are working on the training VM provided for this class, a mail server is already installed and configured.




In Operations/Email you will find the content for handling email. The commonly used operations are shown on the slide. Exchange Server is not covered in this class but you can find out more by using Studio Help.

## Example: Send an Email

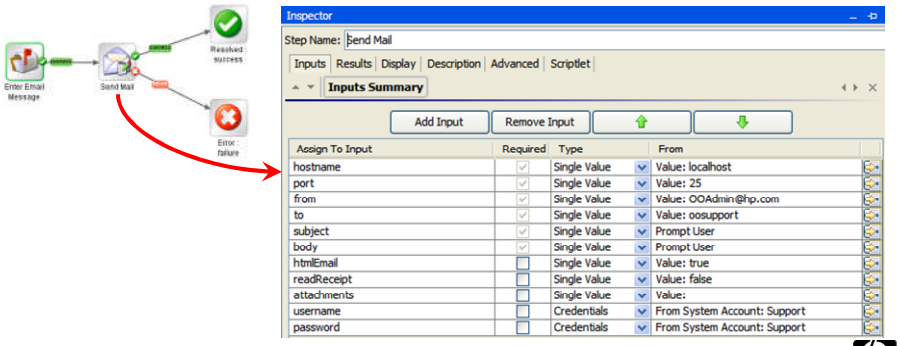
- The Simple Send Email flow:
  - Prompts the user for an email message and subject
  - Sends the email to the specified server and email address


5

This simple flow captures an email subject and body and sends it to the recipient specified in the Send Mail step.

## Configuring Send Mail Inputs

- Hostname: The Mail Server host
- Port: The SMTP port number
- From/to: The from/to address on the email
- Username/password: Stored in OO system accounts



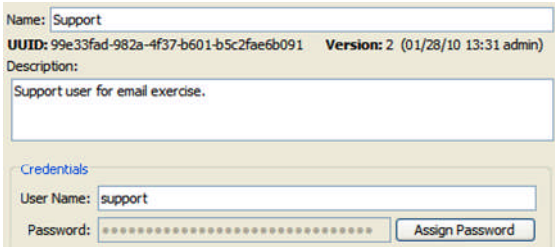

6

The slide shows the Send Mail inputs. Note that many inputs are not required. The inputs should look familiar – you need the name of a mail host, a port number, a From and To address, a subject and body, and other mail options as shown on the slide.

If you frequently send email to specific users, consider using System Accounts to store their information.

## Using System Accounts

- It's a common practice to store email accounts in OO System Accounts
- To create a System Account:
  - Expand Configuration/System Accounts
  - Right-click System Accounts, select New
  - Enter the username and password (passwords are encrypted)
  - Subsequently the username and password can be used in email (and other) steps




7

This slide shows setting up a System Account for an email user.

## Example: Sending a Flow Run URL

- Emailing a flow run report is a common and useful task in OO
- This flow sends the recipient a URL to a flow run report
- Generate Run URL is in Integrations/Hewlett-Packard/Operations Orchestration
- It creates a flow variable named **IC_ReportURL** that can be included in the email body


8

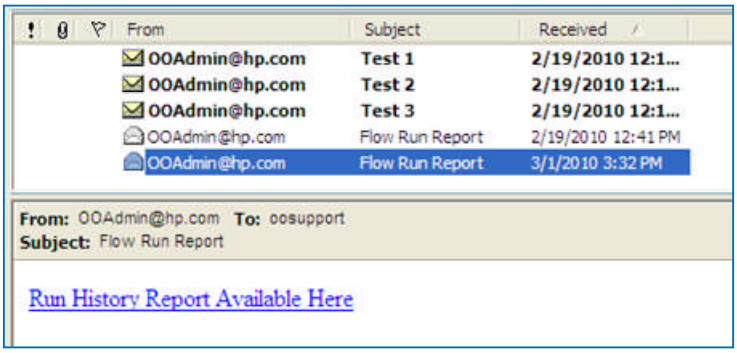
A Flow Run URL is a link to the flow run report in Central that is available at the completion of the flow run.

This is a very valuable tool to use, allowing you to send the flow run report to anyone who might need to see it.

Note that if you run this operation in a flow in Studio you will not see the actual flow run report – the flow must be run from Central to be able to view the actual run report. You can, however, test and debug the flow in Studio even though the actual run report URL is not provided.

## Viewing the Report

- The Run URL is provided to the user in an email
- Clicking the link takes the user directly to the flow run report in Central
- Requires logging into Central if the user is currently not logged in




9

The recipient of the flow run report receives an email with a hyperlink to the run report in Central.

The user will need to log into Central to view the report.

## Getting an Email

- You can also get an email from the mail server and handle it in OO
- Get Mail Message gets the message (or subject only) whose number is provided
- You can also use a Get Mail Message Count operation to get all numbers on a server
- Together these operations are a simple way to get message numbers and associated subjects which can then be used in an OO flow

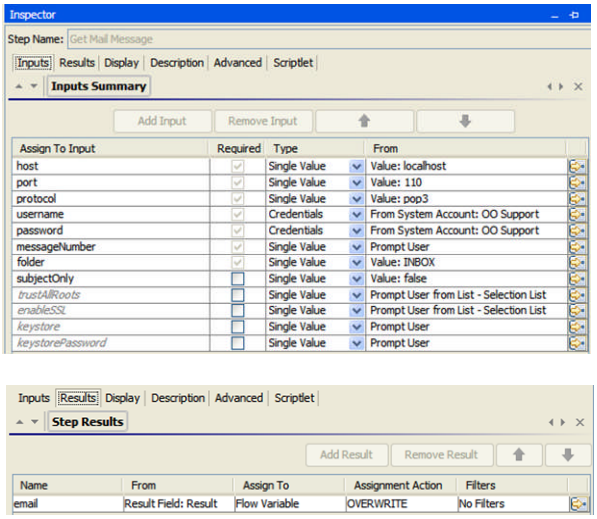

10

You can also retrieve emails using content in the OO library. Get Mail Messages retrieves either an entire message or the subject only, and Get Mail Message Count retrieves message numbers on the mail server.

Together these two operations allow you to summarize email server status an provide summaries or entire emails to users when the flow is run.

## Get Mail Message Inputs and Results

- Unused inputs can be removed
- To assign the email text to a flow variable use a Result based on the Result Field **Result**




11

The slide shows the inputs for Get Mail Message – these are very similar to the other email content in the library, requiring a mail host and port, protocol, accounts, etc.

To assign email text to a flow variable, use a Result based on the Result Field: Result. The email flow variable can then be filtered or used in other ways in your flow.

## Example: Simple OO Email Client

- This flow serves as a simple email client in OO
  - Get Mail Message Count retrieves the number of messages, shown to the user in the Display step
  - The Loop step compiles a list of message numbers
  - Get Mail Message prompts the user to select a message
  - The text of the message is displayed in the Resolved step

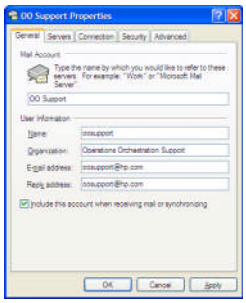

12

This example shows a simple email client implemented in OO. This flow gets the message count, presents a menu of message numbers to the user, then retrieves the select message and displays it to the user.

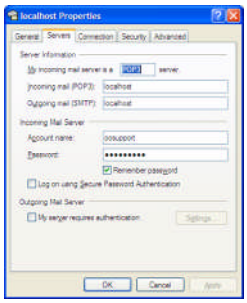
## Configuration Considerations

- You need to know the ports your mail server uses as well as the protocols (IMAP/POP3)
- You may need to configure your email client with the System Accounts you are using in OO
- Outlook Express configuration example – System Account OOSupport:

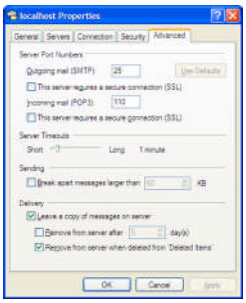
**General tab**



**Servers tab**



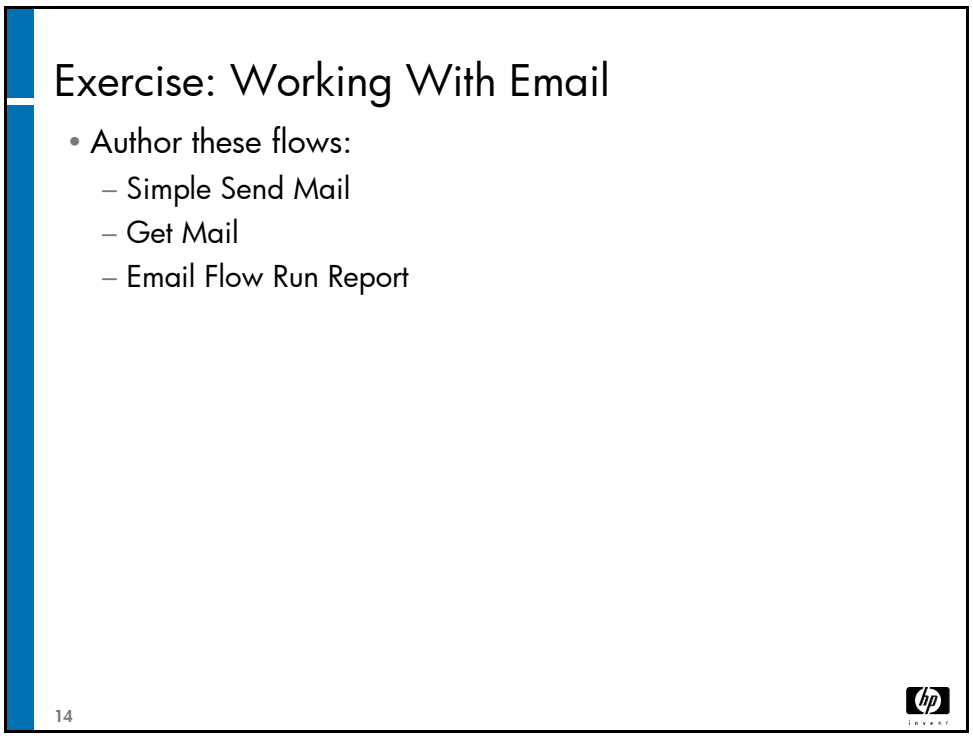
**Advanced tab**



13

This slide reviews some of the configuration options to consider if you install your own email server and are configuring a client to interoperate with it.

In a production you will be interoperating with an established mail server – contact the system administrator for details.




In the exercise you will author flows that handle a variety of email tasks.