## Table of Contents

Lab Overview - HOL-1821-05-CMP - vRealize Orchestrator - Getting Started .................. 3
  Lab Guidance .................................................................................................................. 4

Module 1 - vRealize Orchestrator Overview (30 minutes) ............................................. 11
  Introduction .................................................................................................................... 12
  Introduction to vRealize Orchestrator Client .............................................................. 13
  Running your first workflow ......................................................................................... 18
  Conclusion ..................................................................................................................... 26

Module 2 - Creating a Basic Orchestrator Workflow (30 minutes) ................................. 28
  Introduction .................................................................................................................... 29
  Creating your First Workflow ....................................................................................... 31
  Understanding a Workflow Token ................................................................................ 44
  Conclusion ..................................................................................................................... 48

Module 3 - Introduction to Parameters, Attributes and Scripting Objects (30 minutes) ..... 50
  Introduction .................................................................................................................... 51
  Understanding Input and Output Parameters .............................................................. 53
  Understanding Workflow Attributes .......................................................................... 56
  Using Scripting Objects ............................................................................................... 69
  Conclusion ..................................................................................................................... 78

Module 4 - Using Actions in vRealize Orchestrator (30 minutes) ................................. 80
  Introduction .................................................................................................................... 81
  Overview of actions ...................................................................................................... 83
  Create an Action ........................................................................................................... 85
  Conclusion ..................................................................................................................... 96

Module 5 - Creating and Managing Resource Elements (45 minutes) ......................... 98
  Introduction .................................................................................................................... 99
  Creating Resource Elements ....................................................................................... 101
  Managing Resource Elements ...................................................................................... 107
  Using Resource Elements ............................................................................................ 117
  Conclusion ..................................................................................................................... 129

Module 6 - Creating Advanced Orchestrator Workflows (45 minutes) ......................... 131
  Introduction ................................................................................................................... 132
  Prompting User Input In Workflows ......................................................................... 134
  Using Conditional Execution and Loops ..................................................................... 148
  Bring it together: Sample Advanced Workflow ......................................................... 164
  Asynchronous Workflows ......................................................................................... 198
  Conclusion ................................................................................................................... 207

Module 7 - Error Handling and Troubleshooting (45 minutes) .................................... 209
  Introduction ................................................................................................................... 210
  Error handling and why is it important? .................................................................... 212
  Adding Error Handling to workflows ...................................................................... 213
  Basic Workflow Troubleshooting .............................................................................. 250
  Logging in Workflows ............................................................................................... 264
Lab Overview - HOL-1821-05-CMP - vRealize Orchestrator - Getting Started
Lab Guidance

Note: It will take more than 90 minutes to complete this lab. You should expect to only finish 2-3 of the modules during your time. The modules are independent of each other so you can start at the beginning of any module and proceed from there. You can use the Table of Contents to access any module of your choosing.

The Table of Contents can be accessed in the upper right-hand corner of the Lab Manual.

Use vRealize Orchestrator to simplify the automation of IT tasks. Explore the vRealize Orchestrator development environment and learn important development concepts by building and testing basic workflows, then put those workflows to use inside vRealize Automation to create reusable processes.

Lab Module List:

• **Module 1 - vRealize Orchestrator Overview** (30 minutes) (Basic) Examine and understand the key features and architecture of the vRealize Orchestrator platform. Learn the methods for accessing and interacting with the platform.

• **Module 2 - Creating a Basic Orchestrator Workflow** (30 minutes) (Basic) Learn how to create a basic workflow to resolve a simple yet routine task such as cloning a virtual machine or taking a snapshot.

• **Module 3 - Introduction to Parameters, Attributes and Scripting Objects** (30 minutes) (Advanced) Explore and learn about the objects and constructs that are used to extend the functionality of workflows. These include attributes, scripting objects and decision trees as well as the parameters they use.

• **Module 4 - Using Actions in vRealize Orchestrator** (30 minutes) (Advanced) Reuse your JavaScript code by turning JavaScript Functions into vRealize Orchestrator Actions, then use them repeatedly in multiple workflows. Actions represent individual functions that you use as building blocks in workflows and scripts.

• **Module 5 - Creating and Managing Resource Elements** (45 minutes) (Advanced) Discover how to use external objects such as scripts, XML, or HTML files in your workflows.

• **Module 6 - Creating Advanced Orchestrator Workflows** (45 minutes) (Advanced) Test your Orchestrator skills by in this module where you will create an advanced workflow. Learn about applying best practices to workflow development.

• **Module 7 - Error Handling and Troubleshooting** (45 minutes) (Advanced) Discover the various ways to add error or exception handling to ease in troubleshooting and more resilient workflows. Learn about and create exception binding to your workflows.
Introduction to the Scenario

As you work through the lab's use cases, you'll be assuming roles within a fictional company, Rainpole, and addressing their business and IT challenges.

Rainpole is an IT company that is under pressure to automate many of the internal day-to-day tasks in order to streamline operations, cut down on inconsistencies, and free up teams to work on more proactive tasks. The Rainpole IT Automation team decided to reach out to VMware to help them with their journey to automation. VMware vRealize Orchestrator simplifies the automation of complex IT tasks and integrates with VMware vRealize Suite components to adapt and extend service delivery and operational management, effectively working with existing infrastructure, tools, and processes. You have the most knowledge around existing processes and scripts in place and have been tasked with getting up to speed on vRealize Orchestrator and starting down the path of automating existing processes.
Location of the Main Console

1. The area in the RED box contains the Main Console. The Lab Manual is on the tab to the Right of the Main Console.
2. A particular lab may have additional consoles found on separate tabs in the upper left. You will be directed to open another specific console if needed.
3. Your lab starts with 90 minutes on the timer. The lab can not be saved. All your work must be done during the lab session. But you can click the **EXTEND** to increase your time. If you are at a VMware event, you can extend your lab time twice, for up to 30 minutes. Each click gives you an additional 15 minutes. Outside of VMware events, you can extend your lab time up to 9 hours and 30 minutes. Each click gives you an additional hour.

Alternate Methods of Keyboard Data Entry

During this module, you will input text into the Main Console. Besides directly typing it in, there are two very helpful methods of entering data which make it easier to enter complex data.
Click and Drag Lab Manual Content Into Console Active Window

You can also click and drag text and Command Line Interface (CLI) commands directly from the Lab Manual into the active window in the Main Console.

Accessing the Online International Keyboard

You can also use the Online International Keyboard found in the Main Console.

1. Click on the Keyboard Icon found on the Windows Quick Launch Task Bar.
Click once in active console window

In this example, you will use the Online Keyboard to enter the "@" sign used in email addresses. The "@" sign is Shift-2 on US keyboard layouts.

1. Click once in the active console window.
2. Click on the Shift key.

Click on the @ key

1. Click on the "@" key.

Notice the @ sign entered in the active console window.
Activation Prompt or Watermark

When you first start your lab, you may notice a watermark on the desktop indicating that Windows is not activated.

One of the major benefits of virtualization is that virtual machines can be moved and run on any platform. The Hands-on Labs utilizes this benefit and we are able to run the labs out of multiple datacenters. However, these datacenters may not have identical processors, which triggers a Microsoft activation check through the Internet.

Rest assured, VMware and the Hands-on Labs are in full compliance with Microsoft licensing requirements. The lab that you are using is a self-contained pod and does not have full access to the Internet, which is required for Windows to verify the activation. Without full access to the Internet, this automated process fails and you see this watermark.

This cosmetic issue has no effect on your lab.

Look at the lower right portion of the screen
Please check to see that your lab is finished all the startup routines and is ready for you to start. If you see anything other than "Ready", please wait a few minutes. If after 5 minutes your lab has not changed to "Ready", please ask for assistance.
Module 1 - vRealize Orchestrator Overview
(30 minutes)
Introduction

In this module, you will learn how to navigate the vRealize Orchestrator client interface. Once you're familiar with the client, you will run your first workflow and verify the results.
Introduction to vRealize Orchestrator Client

The vRealize Orchestrator Client is the main interface to develop, see and run workflows. In this exercise, you will familiarize yourself with the vRealize Orchestrator Client user interface.

Launch vRealize Orchestrator Client

Double click on the vRealize Orchestrator Client icon on the desktop.
Log In to vRealize Orchestrator

Use the following credentials

1. Host name: **vra-01a.corp.local:443**
2. User name: **administrator@vsphere.local**
3. Password: **VMware1!**
4. Click **Login**

Review the Three Different Views

When you first log in to the vRealize Orchestrator Client you will be presented with the Run view.
1. Click the dropdown menu at the top of the client and you should also see Design and Administer views available.

This section will cover the differences in the three views.

**Select the Run View**

1. Select the **Run** View

**Run View**

Within this view you can schedule tasks, manage policies, run workflows, access the inventory, and manage permissions. The tabs shown in the screenshot are available on the left side of the client window. Take some time to click on each of the tabs and explore what is available in each.

1. **My Orchestrator** - Summarizes the most recent activities on the Orchestrator server, such as recently modified elements, pending and running workflows, running policies, completed workflows, and workflows that are waiting for user interaction. You can use the My Orchestrator view to perform common administrative tasks, such as running a workflow, importing a package, and setting root access rights.
2. **Scheduler** - Displays a list of all scheduled workflows. The workflows are sorted by name or date, together with their status. You can use the Scheduler view to create, edit, suspend, resume, and cancel scheduled workflows.
3. **Policies** - Displays existing policies. You can use the Policies view to create and apply policies. Policies in vRealize Orchestrator are a series of rules, gauges, thresholds, and event filters that run certain workflows or scripts when specific predefined events occur in Orchestrator or in the technologies that Orchestrator accesses through plug-ins.
4. **Workflows** - Provides access to the Orchestrator workflow library. You can use the Workflows view to view information about each workflow, create, edit, and run workflows, as well as to interact with the workflows.
5. **Inventory** - Displays the objects of the plug-ins that are enabled in Orchestrator. You can use the Inventory view to run workflows on an inventory object.
Select the Design View

1. Click the dropdown menu at the top of the client, and select the Design View

Design view

Within this view you can create and modify workflows and actions. You can also manage resources, configuration elements, and policy templates. The tabs shown in the screenshot are available on the left side of the client. Take some time to click on each of the tabs and explore what is available in each.

1. **Workflows** - Provides access to the Orchestrator workflow library. You can use the Workflows view to view information about each workflow, create, edit, and run workflows, as well as to interact with the workflows.

2. **Actions** - Provides access to the libraries of predefined actions. You can use the Actions view to duplicate actions, export them to a file, or move them to a different module in the actions hierarchical list.

3. **Resources** - Provides access to the list of resource elements. You can use the Resources view to import external objects such as images, sysprep files, HTML templates, XML templates, and custom scripts, and use them as resource elements in workflows.

4. **Configurations** - Provides access to the available configuration elements. You can use the Actions view to create configuration elements to define common attributes across an Orchestrator server.

5. **Packages** - Displays a list of the available packages and where a selected package is used. You can use the Packages view to add, import, export, and synchronize packages.

6. **Inventory** - Displays the objects of the plug-ins that are enabled in Orchestrator. You can use the Inventory view to run workflows on an inventory object.
Select Administer View

1. Click the dropdown menu at the top of the client, and select the **Administer View**

Administer View

Within this view you can access the inventory and manage authorizations and packages. The tabs shown in the screenshot are available on the left side of the client. Take some time to click on each of the tabs and explore what is available in each.

1. **Inventory** - Displays the objects of the plug-ins that are enabled in Orchestrator. You can use the Inventory view to run workflows on an inventory object.
2. **Policy Templates** - Displays a list of the available master policies. You can use the Policy Templates view to create policy templates.
3. **Authorizations** - Displays a list of the available authorization elements. You can use the Authorizations view to create and edit authorization elements.
4. **Packages** - Displays a list of the available packages and where a selected package is used. You can use the Packages view to add, import, export, and synchronize packages.
Running your first workflow

Now that you have familiarized yourself with the vRealize Orchestrator Client, you are going to run your first workflow. This will allow you to apply some of the knowledge you have already gained around vRealize Orchestrator.

Return to the Run View

1. Click the dropdown menu at the top of the client, and select Run from the list to return to the Run view
View the Create a Snapshot Workflow

1. Select the **Workflow** tab
2. Browse the workflows tree to **Library > vCenter > Virtual Machine Management > Snapshot > Create a snapshot**
3. On the right, select the **Schema** tab

Take a moment to look at what the workflow is doing. You will see that it is using an action called `createSnapshot` and the another action called `vim3WaitTaskEnd`. So this workflow will take a snapshot of a virtual machine and then for that task to end before the workflow completes.

3. Click the **Start workflow** Icon
Choose the VM

1. Click on Not Set to browse for the vm

Choose the VM

1. Browse to vCenter Server > https://vcsa-01a.corp.local:443/sdk > Datacenters > RegionA01 > vm > VRM > brownfield-01a
2. Click the Select button
1. **Click Next**

**Enter snapshot creation parameters**

1. Type in **Test** for the snapshot name
2. Type in **My First Workflow** for the description

Leave all the other settings at the default values.

3. Click the **Submit** button
Launch the Chrome Browser

1. From the Windows Quick Launch task bar, click the Google Chrome icon to open the browser

Log in to the vSphere Web Client

1. Click on the RegionA folder and select the vCenter - Region A bookmark
2. Check the box for Use Windows session authentication

This will log you on with the administrator@corp.local credentials
3. Click the **Login** button

**Switch to VMs and Templates View**

1. Mouse over the **Home** icon and select **Home** from the dropdown menu (not shown)
2. Click on **VMs and Templates** under Inventories

**Manage Snapshots**

1. Browse the inventory tree to `vcsa-01a.corp.local > RegionA01 > VRM > brownfield-01a`
2. Right click and browse to **Snapshots > Manage Snapshots...**

**Manage Snapshots: Verify Test**

1. Select the **Test** snapshot
2. Verify the snapshot. It should have **My First Workflow** as its description
3. You no longer need this snapshot. Click the **All Actions** button and choose **Delete Snapshot**

**Manage Snapshots: Delete Test**

1. Click **Yes** to verify you want to delete the snapshot
Manage Snapshots: Verify Delete

You will notice that the Test snapshot is now gone.

Close the Chrome Browser

1. Click the X in the top right hand comer of the browser to close it

Close the vRealize Orchestrator Client

1. Close the vRealize Orchestrator Client by clicking the X in the top right corner

Close the vRealize Orchestrator Client

1. Click the Exit to confirm you want to close the vRealize Orchestrator Client.
Conclusion

vRealize Orchestrator allows you to perform any operation on various systems (VMware products and non-VMware products).

Its main interface is a Java client that allows to develop new workflows and run them. From its inventory a global view of all endpoints connected is provided.

You’ve finished Module 1

Congratulations on completing Module 1.

Looking for additional information on vRealize Orchestrator basics, try one of these:

- Go to http://bit.ly/2veqDW2
- Or use your smart device to scan the QRC Code.

Proceed to any module below which interests you most.

- **Module 2 - Creating a Basic Orchestrator Workflow** (30 minutes) (Basic)
  Learn how to create a basic workflow to resolve a simple yet routine task such as cloning a virtual machine or taking a snapshot.

- **Module 3 - Introduction to Parameters, Attributes and Scripting Objects** (30 minutes) (Advanced)
  Explore and learn about the objects and constructs that are used to extend the functionality of workflows. These include attributes, scripting objects and decision trees as well as the parameters they use.

- **Module 4 - Using Actions in vRealize Orchestrator** (30 minutes) (Advanced)
  Reuse your JavaScript code by turning JavaScript Functions into vRealize Orchestrator Actions, then use them repeatedly in multiple workflows. Actions represent individual functions that you use as building blocks in workflows and scripts.
• **Module 5 - Creating and Managing Resource Elements** (45 minutes) (Advanced) Discover how to use external objects such as scripts, XML, or HTML files in your workflows.

• **Module 6 - Creating Advanced Orchestrator Workflows** (45 minutes) (Advanced) Test your Orchestrator skills by in this module where you will create an advanced workflow. Learn about applying best practices to workflow development.

• **Module 7 - Error Handling and Troubleshooting** (45 minutes) (Advanced) Discover the various ways to add error or exception handling to ease in troubleshooting and more resilient workflows. Learn about and create exception binding to your workflows.

**How to End Lab**

To end your lab click on the **END** button.
Module 2 - Creating a Basic Orchestrator Workflow (30 minutes)
Introduction

This module covers the basics of vRealize Orchestrator workflow creation using a simple Scriptable Task consisting of a few lines of JavaScript code.

Launch vRealize Orchestrator Client

If vRealize Orchestrator Client is already opened, this step and the next one can be ignored.

1. Double click on the vRealize Orchestrator Client icon on the desktop.
Log In to vRealize Orchestrator

Use the following credentials:

1. Hostname: **vra-01a.corp.local:443** (this is the default value)
2. Username: **administrator@vsphere.local**
3. Password: **VMware1!**
4. Click **Login**

Switch to Design View

1. Click the dropdown menu at the top of the client and select **Design**.
Creating your First Workflow

vRealize Orchestrator workflows allow you to perform a variety of tasks. Each workflow can have multiple inputs and return multiple outputs. This lesson covers the creation and execution of a simple workflow.

Creating a Workflow

Let's build a vRealize Orchestrator workflow:

1. Click the **Workflow** tab
2. Select the **HOL-1821** folder
3. Click on the **New workflow** icon

Naming a Workflow

1. Type **My First Workflow** for the Workflow name
2. Click the **Ok** button
1. Select the **Schema** tab

Workflow schemas are made up of elements.

2. Take a moment to look through the list of available elements
1. In the Generic section, Select **Scriptable task**
2. Drag the **Scriptable task** element onto the **Schema**
   (Note: drag the element between the two other elements on the canvas)

The Scriptable Task element is between other elements. The green arrow represents the Start workflow element and the one to the far right is the End workflow element. These 2 elements are mandatory, and must be present for the workflow to run.

### Configuring the New Workflow: Inputs

1. Select the **Inputs** tab
2. Click the **Add parameter** icon to add a parameter with the default name of `arg_in_0`

Parameters within vRealize Orchestrator are basically input and output variables of a workflow in the simplest of terms.

Input parameters are needed for most workflows and are values that are passed into the workflow when it starts via a user, application, another workflow, or even an action.

3. Click on `arg_in_0` to name the parameter (not shown)
4. Type `vm` for the Attribute name to update the name of this parameter
5. Click the **Ok** button
Configuring the New Workflow: Inputs

1. Click on **string** in the Type column to open the "Select a type..." window
2. Type **VC:VirtualMachine** in the Filter box
3. Select the **VC:VirtualMachine** Type (Note: this value should be at the top of the filtered list - it may be necessary to scroll up in order to find it.)
4. Click the **Accept** button

Configuring the New Workflow: Description

1. Double-click on the area under the Description column, and type **My vm** in the Description field
Editing a Scriptable Task

1. Select the **Schema** tab
2. Click the **Scriptable task** to open it in the lower pane
3. Click the **Undock** icon in the corner of the lower pane to open the contents in a larger window
Visual Bindings

1. Select the **Visual Bindings** tab

Bindings allow you to connect workflow input or output to local parameter. This is needed so the local element can process the parameter. In this case, the input parameter is passed to a local parameter so it can be used within the Scriptable Task element that is part of the workflow. It is standard practice to have all parameters bound in a workflow - in fact, if a variable is not bound anywhere, a warning will appear when the workflow is saved. The warning message indicates the parameter that is not bound.

2. Select the *vm* input parameter
3. Drag it over to the **IN** field to create the binding
Local IN Parameters

<table>
<thead>
<tr>
<th>Local Parameter</th>
<th>Source parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vm</td>
<td>vm [in-parameter]</td>
<td>VC:VirtualMachine</td>
<td>My vm</td>
</tr>
</tbody>
</table>

1. Select the IN tab

Notice that now there is a local IN parameter for the Scriptable task.
1. Select the **Scripting** tab
2. **Enter** the following code (if drag and drop is used, make sure to keep the focus in the scripting tab all the time):

```java
System.log("Name: " + " " + vm.summary.config.name);
System.log("Hostname: " + " " + vm.summary.guest.hostName);
System.log("IP Address: " + " " + vm.summary.guest.ipAddress);
System.log("Description for VM: " + " " + vm.summary.config.annotation);
System.log("Number of CPU: " + " " + vm.summary.config.numCpu);
System.log("Amount of memory in MB: " + " " + vm.summary.config.memorySizeMB);
System.log("Number of Network Adapters: " + " " + vm.summary.config.numEthernetCards);
System.log("Number of Disks: " + " " + vm.summary.config.numVirtualDisks);
System.log("Guest OS: " + " " + vm.summary.config.guestFullName);
System.log("Is template: " + " " + vm.summary.config.template);
```
The code will log information available from the API of the virtual machine entered as input into the workflow. The information logged by the workflow includes:

- The virtual machine name
- The virtual machine hostname
- The virtual machine ip address
- The virtual machine description field
- The cpu count for the virtual machine
- The memory count for the virtual machine
- The network adapter count for the virtual machine
- The disk count for the virtual machine
- The virtual machine guest OS full name
- Whether or not the virtual machine is marked as a template

It is also taking this data and formatting it in a much more readable format.

3. Click the Red X button to close the window and return it to the dock
1. Click the **Validate** icon

   This allows vRealize Orchestrator to check the workflow and make sure there are not any issues, such as parameters that are not bound or used.

2. Click the **Close** button
Save and Close

1. Click the **Save and close** button

Running Your Workflow

1. Click the **Workflow** tab
2. Browse to **HOL-1821 > My First Workflow**
3. Click on the **Start workflow** icon

Running Your Workflow

1. Click **Not set** to select the virtual machine to use as input into the workflow
Running Your Workflow

1. Browse to **vCenter Server** > **https://vcs-01a.corp.local:443/sdk** > **Datacenters** > **RegionA01** > **vm** > **brownfield-01a**
2. Click the **Select** button

Running Your Workflow

1. Click the **Submit** button
Running Your Workflow

When the workflow completes, the workflow token shows a green check mark next to it. This indicates successful execution of the workflow.
Understanding a Workflow Token

A workflow token represents an instance of a workflow that is running or has run previously.

Find Workflow Token from the Last Run

1. if necessary, expand the workflow My First Workflow
2. Select the workflow token (with the green check mark) from the previous exercise.
Get the Details of the Workflow Instances

1. Select **Schema** tab
2. If necessary, click and drag the **summary** window up to see the details.
Look at the Variables

1. Click the Variables tab on the lower right hand side of the window

Notice that the value for the vm input is the brownfield-01a virtual machine that was selected as input of the workflow. Variables are Inputs, outputs and attributes, and the values shown are the values at the time of completion of the workflow.

Look at the Logs

1. Click the Logs tab on the lower right hand side of the window
2. Make sure Info is selected

Notice that the script worked and all the values were logged in the format specified in the Scriptable Task.
Look at the Events

1. Select **Events** tab

This tab shows a list of events relevant to the workflow execution, including:

- beginning of the workflow
- end of the workflow
- every call to Server.log, Server.warn, Server.error

Note that Events are stored in database and can be seen from any Orchestrator node and at any time.
Conclusion

A vRealize Orchestrator workflow is very easy to create, and it can be executed once or multiple times. A workflow contains parameters (inputs, outputs, attributes) that can be bound to workflow elements. Each workflow run is called a workflow token.

Discover more about workflows in the next module.

You’ve finished Module 2

Congratulations on completing Module 2.

If you are looking for additional information on Developing workflows, try one of these:

- Go to http://bit.ly/2ugo6gy
- Or use your smart device to scan the QRC Code.

Proceed to any module below which interests you most. [Add any custom/optional information for your lab manual.]

- Module 1 - vRealize Orchestrator Overview (30 minutes) (Basic) Examine and understand the key features and architecture of the vRealize Orchestrator platform. Learn the methods for accessing and interacting with platform.
- Module 3 - Introduction to Parameters, Attributes and Scripting Objects (30 minutes) (Advanced) Explore and learn about the objects and constructs that are used to extend the functionality of workflows. These include attributes, scripting objects and decision trees as well as the parameters they use.
- Module 4 - Using Actions in vRealize Orchestrator (30 minutes) (Advanced) Reuse your JavaScript code by turning JavaScript Functions into vRealize Orchestrator Actions, then use them repeatedly in multiple workflows. Actions represent individual functions that you use as building blocks in workflows and scripts.
• **Module 5 - Creating and Managing Resource Elements** (45 minutes) (Advanced) Discover how to use external objects such as scripts, XML, or HTML files in your workflows.

• **Module 6 - Creating Advanced Orchestrator Workflows** (45 minutes) (Advanced) Test your Orchestrator skills by in this module where you will create an advanced workflow. Learn about applying best practices to workflow development.

• **Module 7 - Error Handling and Troubleshooting** (45 minutes) (Advanced) Discover the various ways to add error or exception handling to ease in troubleshooting and more resilient workflows. Learn about and create exception binding to your workflows.

**How to End Lab**

To end your lab click on the **END** button.
Module 3 - Introduction to Parameters, Attributes and Scripting Objects (30 minutes)
Introduction

This module covers the basics to make a workflow, starting with inputs and outputs, then looking at attributes, and finally scripting objects provided by vRealize Orchestrator plugins.

It explains how all these constructs are interconnected, how values and objects flow between the various elements in a workflow, and how they are used to do real work!

Launch vRealize Orchestrator Client

If vRealize Orchestrator Client is already opened, this step and the next one can be ignored.

1. Double click on the vRealize Orchestrator Client icon on the desktop.
Log In to vRealize Orchestrator

Use the following credentials

1. Hostname: **vra-01a.corp.local:443** (this is the default value)
2. User name: **administrator@vsphere.local**
3. Password: **VMware1!**
4. Click **Login**

Switch to Design View

1. Click the dropdown menu at the top of the client and select **Design**.
Understanding Input and Output Parameters

Input and output parameters are essential to understand in vRealize Orchestrator, as they are used in nearly every available workflow.

Introduction

Parameters within vRealize Orchestrator are basically input and output variables of a workflow in the simplest of terms.

**Input parameters** are needed for most workflows and are values that are passed into the workflow when it starts via a user, application, another workflow, or even an action. A good example of this might be a workflow that is used to snapshot a virtual machine. An input parameter for such a workflow may be the virtual machine name. Another thing to note about input parameters is that they are read-only, and cannot be changed during workflow execution.

**Output parameters** represent the result of the workflow and can be passed on to other elements when the workflow has completed. For that same virtual machine snapshot workflow, the output parameter is the resulting snapshot. Output parameters are write-only; to read the value of a parameter within a workflow, it cannot be an output parameter. However, this output parameter value is readable for any workflows chained after this one.
Identify Parameters

Time to explore parameters a bit more.

1. Click on the Workflows tab
2. Browse the workflows tree to Library > vCenter>Configuration > Add a vCenter Server Instance
3. Click on Inputs

Here are the input parameters for a workflow used to add vCenter objects to the vRealize Orchestrator inventory. Once a vCenter object is added, it is possible to run other workflows against it - for example, a workflow to clone a virtual machine.

Back to parameters. Observe the different input parameters being gathered. There is a "Type" column, which dictates the type of value the input parameter will accept. For example, the password input parameter type is SecureString, which allows vRealize Orchestrator to take in the string but in a secure way since it is a password and to not show this value in clear text.

The best practice is to format input parameter names in a format called camelCase. camelCase is where the first word in the name starts with a lowercase letter and all proceeding ones start with a capital letter. A good example would be the parameter "sessionPerUser". camelCase is the standard used for parameters in vRealize Orchestrator and what we will be using throughout the lab.

4. Now, click on the Output tab

Are there any output parameters listed? What is their type?
Types of Parameters

Each parameter has a type. This type can be simple type like string, number, array, ... but it can also be complex type such as VC:VirtualMachine, vCACCAFE:CatalogItemRequest or REST:RESTHost. All of these complex types come from plugins that have been added to vRealize Orchestrator.

When an input is chosen, vRealize Orchestrator will only allow objects from that input's type to be selected, in order to avoid user mistakes. A complex object can carry a lot of properties that can be accessed directly from within a scriptable task, which makes it very handy.

Overall, the fact that vRealize Orchestrator accounts for all variable types simplifies development, and makes it more reliable.
Understanding Workflow Attributes

Workflow attributes are one of the key constructs used to carry values between elements in a workflow.

Introduction

Attributes are variables that are generally read/write, and they can be used to transfer data between the elements of a workflow. An attribute can be locked so it is read-only across all workflow elements. This is very useful to define a static value that is needed in several elements of a workflow.

Read-only workflow attributes act as global constants for a workflow. Writeable attributes act as a workflow’s global variables.

Values of attributes can be set and modified as follows:

- Set attribute value in a workflow
- Assign attribute value from a configuration element
- Attribute linked to the output parameter of a workflow element

Identify Attributes

1. Click on the Workflows tab
2. Browse the workflows tree to HOL-1821 > Playing with attributes
3. Click the Edit icon
1. Click on the **General** tab
2. **Scroll down** to the bottom of the General tab

Here are the attributes listed. They can be blank or have a default value, and like Parameters, they have a type.

3. Click on the **checkbox** near **attribute2**

By doing this, the attribute is now read-only, and can be used by any workflow element only as IN.

4. Double-click into the value field for the **attribute1** attribute and type in **my value 1**

This would set the attribute's default value to **my value 1**. It will remain read/write and can used by any workflow element as IN and/or OUT.
Look at the Value of the Attributes

1. Select **Schema** tab
2. Click on **Run** button
3. Once the workflow completes, look at the logs by clicking the **Logs** tab on the right

The log output shows the Business Group Name, and it shows the attribute value twice (because the workflow contains two scriptable tasks that show attribute values.)

Note: if necessary, you can click to the left of the Logs window and drag to expand it.

**Use Configuration Element**

Configuration elements are a list of attributes you can use to configure constants across the entire Orchestrator instance.

All workflows, actions, and policies running in a particular Orchestrator server can use the attributes set in a configuration element. Setting attributes in configuration elements allows you to make the same attribute values available to all the workflows, actions, and policies running on the Orchestrator server.
Set a Configuration Element

1. Click on the **General** tab
2. **Scroll down** to the bottom of the General tab
3. Click on **double arrow button** for the businessGroup attribute
Choose Configuration Element Attribute

1. Type **Module 3** in the Filter box
2. Select the configuration element **Module 3** from the filtered list
3. Select the attribute **businessGroup**
4. Click on **Select** to apply the configuration element to this attribute. You will see a value of "Module 3 : businessGroup" for the businessGroup attribute.
Look at the Value of the Attributes

1. Select **Schema** tab
2. Click on **Run** button
3. Once the workflow has completed, view the log from the new execution (the **Logs** tab should already be selected.)

Notice that the Business group name has changed to **Development** (which is the value contained in the Configuration Element.)
Change the Value of an Attribute in a Workflow Element

1. Scroll to the middle of the schema
2. Select **change attr value** workflow element
3. Click the **Undock** icon to open the lower panel in a new window
1. Select **Visual Binding** tab
2. Drag and Drop **attribute1** from **Out Attributes** box to **OUT** box
3. Notice that **attribute2** is only available as **In Attributes** and not as **Out Attributes** (it is because the attribute has been made read-only in a previous step).
Update Script

1. Select **Scripting** tab
2. Enter the following code:

```javascript
attribute1 = attribute1 + " - my new value";
```
Look at the Value of the Attributes

1. Select **Schema** tab
2. Click on **Run** button
3. Look at the logs

Notice the second time the value of attribute 1 is shown, "my new value" is visible but with null at the beginning. This does not seem to be the expected behavior - let's see in the next steps how to fix that.
1. Scroll to the middle of the schema
2. Select **change attr value** workflow element
3. Click the **Undock** icon to open the lower panel in a new window
Change the Value of an Attribute in a Workflow Element (Continued)

1. Click the **Visual Binding** tab
2. Drag and Drop **attribute1** from **In Attributes** box to **IN** box
3. Click the **X** to close the window and return to the Schema panel
Look at the Value of the Attributes

1. Select **Schema** tab
2. Click on **Run** button
3. Look at the logs

Notice the second time the value of attribute 1 is shown, "my new value 1" has been well concatenated to " - my new value".

Because the attribute "attribute1" was not an IN of the scriptable task, its value was null. After adding it as an IN, its value is available in the scriptable task and properly concatenated.

**Save and Close**

1. Click on **Save and close**
Using Scripting Objects

Scripting objects are JavaScript classes that can be used in any Scriptable task and Actions. They can be browsed from the API explorer and comes from vRealize Orchestrator core as well as the plugins.

Select "Show VM IP Address" workflow

1. Select Workflow tab
2. Select Show VM IP Address workflow
3. Click on Edit icon
Select "show ip" scriptable task

1. Select Schema
2. Select show ip scriptable task
3. Select the Scripting tab

From the scripting tab you can use JavaScript code. You also have the ability to leverage the API explorer to view the vRealize Orchestrator types available and what type of data is available and can be scripted against. In your workflow we will be leveraging the VC:VirtualMachine type.

4. Click the Search API icon
API Explorer: API Search

1. Type VC:VirtualMachine in the Containing text field
2. Click the Search button
3. Click on Name column to sort the result by name.
4. Select VC / VC:VirtualMachine
5. Click Go to selection
6. Click the Close button

Undock Scripting tab
1. Click on **Undock** icon

**API Explorer: Browsing**

![API Explorer Diagram](image)

**VC:VirtualMachine** Type should be selected in the type list, below there is a description of the object type.

1. **Scroll up** if necessary
2. Click the Scripting Object **VcVirtualMachine**

A scripting object is the javascript object that can be used in a scriptable task.
API Explorer: Browsing

1. Click the arrow to expand `VcVirtualMachine`

This shows all properties and methods in the `VcVirtualMachine` scripting object. Take a moment to click on some of the properties. When a property or method is selected, the lower panel displays details about this property or method.

API Explorer: Browsing
1. Select the **guest** property

Click on the **VcGuestInfo** Return Type

Return Types are the value we can pull via the API for the given object type and property.

**Find ipAddress**

1. Expand **VcGuestInfo** by clicking on the arrow.
2. Observe all the available attributes, **ipAddress** is listed as one of the information

**Show IP address**

If we summarize, we started at the **VC:VirtualMachine** object which points toward the Scripting class **VcVirtualMachine**, followed by **guest**, then **ipAddress**. From a javascript standpoint, it is written like this: **VM.guest.ipAddress**

1. Enter the following code:

   ```javascript
   System.log(VM.guest.ipAddress);
   ```

This code write the IP address of the machine in the log.
Close the window

1. Click on the X icon to close the window and dock it back.
Launch Workflow

1. Click on **Run** button
2. Click on **Not Set** (partially obscured here)
3. Expand and Select **vSphere vCenter Plugin** > 
   https://vcsa-01a.corp.local:443/sdk > Datacenters > RegionA01 > vm > puppet-01a
4. Click **Select**
5. Click **Submit**
Find the IP Address

The IP Address of the VM selected when you ran the workflow is shown in the logs.
Conclusion

Inputs, outputs, and attributes are the variables available in vRealize Orchestrator. They can be passed between workflow elements within a workflow, or between workflows, and their type allows for consistency across a process.

You've finished Module 3

Congratulations on completing Module 3.

If you are looking for additional information on Key Concepts of workflows, try one of these:

- Go to http://bit.ly/2t5w3pM
- Or use your smart device to scan the QRC Code.

Proceed to any module below which interests you most. [Add any custom/optional information for your lab manual.]

- **Module 1 - vRealize Orchestrator Overview** (30 minutes) (Basic) Examine and understand the key features and architecture of the vRealize Orchestrator platform. Learn the methods for accessing and interacting with platform.
- **Module 2 - Creating a Basic Orchestrator Workflow** (30 minutes) (Basic) Learn how to create a basic workflow to resolve a simple yet routine task such as cloning a virtual machine or taking a snapshot.
- **Module 4 - Using Actions in vRealize Orchestrator** (30 minutes) (Advanced) Reuse your JavaScript code by turning JavaScript Functions into vRealize Orchestrator Actions, then use them repeatedly in multiple workflows. Actions represent individual functions that you use as building blocks in workflows and scripts.
- **Module 5 - Creating and Managing Resource Elements** (45 minutes) (Advanced) Discover how to use external objects such as scripts, XML, or HTML files in your workflows.
• **Module 6 - Creating Advanced Orchestrator Workflows** (45 minutes) (Advanced) Test your Orchestrator skills by in this module where you will create an advanced workflow. Learn about applying best practices to workflow development.

• **Module 7 - Error Handling and Troubleshooting** (45 minutes) (Advanced) Discover the various ways to add error or exception handling to ease in troubleshooting and more resilient workflows. Learn about and create exception binding to your workflows.

### How to End Lab

To end your lab click on the **END** button.
Module 4 - Using Actions in vRealize Orchestrator (30 minutes)
Introduction

This module explain in detail how Actions are created, used, and why they are so important to the development of clean workflows. They are a key component in vRealize Orchestrator, allowing you to easily reuse a piece of code in multiple workflows and even in other actions.

Launch vRealize Orchestrator Client

If vRealize Orchestrator Client is already opened, this step and the next one can be ignored.

1. Double click on the vRealize Orchestrator Client icon on the desktop.
Log In to vRealize Orchestrator

Use the following credentials

1. Host name: vra-01a.corp.local:443
2. User name: administrator@vsphere.local
3. Password: VMware1!
4. Click Login

Switch to Design View

1. Click the dropdown menu at the top of the client and select Design.
Overview of actions

Actions in vRealize Orchestrator are generally pieces of code that can be reused often. In another language they could be referred to as functions.

The main difference between an Action and a workflow is that Actions can only return one variable and a workflow can return many. Actions are also built entirely using JavaScript, and there is no visual programming like with a workflow.

Browse Out of the Box Actions

1. Select the Actions tab

See a list of what look to be folders listed - these are referred to as modules.

Modules are basically a way to group actions, like a folder but only one level is allowed - there is no possibility to create "submodules." The naming convention is quite different and looks like a URL. This is done as a way to track each module's purpose and its author or provider, i.e. com.vmware.library.vc.networking is a module with actions associated with vCenter networking provided by VMware.
Browse Out of the Box Actions

1. Scroll down and open the module `com.vmware.library.vc.basic`
2. Select the `cloneVM` action
3. On the right hand side select the scripting tab

The scripting tab of an action gives details about the parameters used and the JavaScript code being used. Notice that tabs like schema or visual bindings do not exist for an action.

Take a few minutes to look through the other modules and actions available.

Every module and action starting with com.vmware.library in the name is available out of the box with vRealize Orchestrator. VMware has you covered for a lot of things that can be used right away just by dragging and dropping an action into a workflow.
Create an Action

Now that you’ve learned more about actions, let’s create a module with a basic action in it.

Creating a Module

1. Select the **Actions** tab (or scroll up to the top of the Actions list if necessary)
2. Right click **Administrator @ vra-01a.corp.local**
3. Select **New module**

Naming a Module

1. Type **com.rainpole.vc.vm** for the Module name

Remember, modules are a way to group actions. One thing that makes them different from workflow folder is that only one level is allowed (there is no submodule.) Also, their naming convention tends to look like a Java class and includes information on what is included in the module.

For example, the name used (vc.vm) is included in the end to reflect that the module is going to contain modules related to vCenter virtual machines.
2. Click the Ok button

Creating an Action

1. Select the `com.rainpole.vc.vm` module (it should be selected by default after the previous step)
2. Click the Add Action icon

Remember, Actions in vRealize Orchestrator are generally pieces of code that can be reused often.
Naming an Action

1. Type `osNameToGuestOS` for the Action name
2. Click the **Ok** button

In vRealize Orchestrator it is standard practice to name items using camelCase. camelCase means that the first word in the name starts with a lower case letter and all proceeding ones start with capital letters. Also the action name should be descriptive. In this module the action takes a OS friendly name and converts it virtual machine guest OS identifier that orchestrator and vSphere understand.

Configuring the Action

1. Select the **Scripting** tab
2. Click **Add parameter** icon
This will be an input parameter that the action will take in as a string

3. Click on **arg0**
4. Type **osName** for the Attribute name

Notice that it is camelCase for the naming convention

5. Click the **Ok** button
6. Click the **void** link, this is for the Return Type

The Return Type is where the format of the output will be set.

**Return Type**

1. Type **VC:VirtualMachineGuest** in the Filter field
2. Select the **VC:VirtualMachineGuestOsIdentifier** type
3. Click the **Accept** button
Scripting

vRealize Orchestrator is JavaScript based, and actions are based on it. Unlike with workflows, there are no visual programming options available.

This JavaScript code takes an input parameter string that is for a Windows or CentOS OS and convert it to a type understood by Orchestrator that identifies a virtual machine's guest OS type. Take a few minutes to look over the code.

```javascript
if (osName == "Windows" || osName == "Windows 2012" || osName == "Windows 2012 R2") {
    return VcVirtualMachineGuestOsIdentifier.windows8Server64Guest;
} else if (osName == "CentOS" || osName == "CentOS 7" || osName == "CentOS 6" || osName == "CentOS 5") {
    return VcVirtualMachineGuestOsIdentifier.centos64Guest;
}
```
1. Make sure **Scripting** tab is selected
2. **Type** the code into the script window.
3. Click the **Save and close** button

**Using an Action in a Workflow**

![Image of workflow tool interface]

Add this action to a workflow to test, and make sure it takes a string value and converts it to the correct type and value.

1. Select the **Workflow** tab
2. Right click the **HOL-1821** folder
3. Select **New workflow**

**Naming a Workflow**

![Image of workflow naming interface]

1. Type **My First Action Workflow** for the Workflow name
2. Click the **Ok** button
1. Select the Schema tab
2. Select All Actions on the left hand side of the screen
3. Browse to the action osNameToGuestOS - remember that it's in the com.rainpole.vc.vm folder
4. Drag and drop the action on the Schema between the Start and End workflow elements
5. Click the Setup button
Promote Workflow Parameters

Accept the defaults on Promote Workflow Input/output Parameters.

vRealize Orchestrator is smart enough to create the right binding for Input and Output parameters that were defined in the action element. This step can be skipped and binding can be done manually otherwise.

1. Click the **Promote** button

Save and Close

1. Click the **Save and close** button
Running the Workflow

1. Select the **Workflow** tab
2. Browse to **HOL-1821 > My First Action Workflow**
3. Click the **Start workflow** icon
Common Parameters

1. Type **Windows 2012 R2** for the guest os name
2. Click the **Submit** button

Verifying the Workflow Run

1. Ensure you are on the **Schema** tab, then click and drag the **summary** window up to see the details.
Verifying the Workflow Run

1. Select the **Variables** tab

The workflow returned a Type of VC:VirtualMachineGuestOsIdentifier and the correct value for the Windows.

If you like, you can run the workflow again with other values like CentOS as the name of the Operating System.
Conclusion

*Actions are like "functions" in other programming languages. They are meant to be reused across multiple workflows, or even in other actions. They are also very fast to execute.*

You've finished Module 4

Congratulations on completing Module 4.

If you are looking for additional information on Developing Actions, try one of these:

- Or use your smart device to scan the QRC Code.

Proceed to any module below which interests you most. [Add any custom/optional information for your lab manual.]

- **Module 1 - vRealize Orchestrator Overview** (30 minutes) (Basic) Examine and understand the key features and architecture of the vRealize Orchestrator platform. Learn the methods for accessing and interacting with platform.
- **Module 2 - Creating a Basic Orchestrator Workflow** (30 minutes) (Basic) Learn how to create a basic workflow to resolve a simple yet routine task such as cloning a virtual machine or taking a snapshot.
- **Module 3 - Introduction to Parameters, Attributes and Scripting Objects** (30 minutes) (Advanced) Explore and learn about the objects and constructs that are used to extend the functionality of workflows. These include attributes, scripting objects and decision trees as well as the parameters they use.
- **Module 5 - Creating and Managing Resource Elements** (45 minutes) (Advanced) Discover how to use external objects such as scripts, XML, or HTML files in your workflows.
- **Module 6 - Creating Advanced Orchestrator Workflows** (45 minutes) (Advanced) Test your Orchestrator skills by in this module where you will create...
an advanced workflow. Learn about applying best practices to workflow development.

- **Module 7 - Error Handling and Troubleshooting** (45 minutes) (Advanced)
  Discover the various ways to add error or exception handling to ease in troubleshooting and more resilient workflows. Learn about and create exception binding to your workflows.

**How to End Lab**

To end your lab click on the **END** button.
Module 5 - Creating and Managing Resource Elements (45 minutes)
Introduction

Resource Elements are a very important and useful construct in vRealize Orchestrator. From storing scripts, to HTML or XML templates, or even images, they can be easily used from within a workflow.

Each aspect of resource element management, from creation to deletion, is covered from an UI standpoint (using vRealize Orchestrator Client) and from a code standpoint (using javascript code in a workflow) in this module.

Launch vRealize Orchestrator Client

If vRealize Orchestrator Client is already opened, this step and the next one can be ignored.

1. Double click on the vRealize Orchestrator Client icon on the desktop.
Login to vRealize Orchestrator Client

Use the following credentials

1. Host name: **vra-01a.corp.local:443**
2. User name: **administrator@vsphere.local**
3. Password: **VMware1!**
4. Click **Login**
Creating Resource Elements

Let’s see how to create resource elements from the vRealize Orchestrator Client, and also directly from a workflow.

Switch to Design View

1. Click the dropdown menu at the top of the client and select **Design**.

Import Resource Element

To start, you will import an HTML page into vRealize Orchestrator as a resource element. It can be used later on in a workflow to send customized emails.
1. Select **Resources** tab
2. Select **HOL-1821-5** folder
3. Click on **Import Resources...** button
4. Browse the folder **C:\hol\HOL-1821-05-CMP**
5. Select **email-template.html**
6. Click on **Open** button

**View the Content of the Resource**

vRealize Orchestrator allows certain types of files to be viewed (such as txt, csv, html and a few other types.) These files can be viewed in the Resources section of the Design view.

1. Select the resource from the left panel
2. Click on **Viewer** tab
Create a New Workflow

Now let's see how it is possible to create a Resource Element from within a workflow.

1. Click on **Workflows** tab
2. Right-Click on folder **HOL-1821**
3. Click on **New Workflow**

**Name the New Workflow**

4. Enter "Create Resource Element" as the workflow name
5. Click **OK**
Add Scriptable Task

1. Click to **Schema** tab
2. Drag and drop the **Scriptable task** item to the arrow in the schema
3. Click the **Undock** icon to open the Scriptable task pane in a new window
Add Javascript Code to Create Resource Element

1. Click on **Scripting** tab
2. Copy the following code:

   ```javascript
   var myMimeAttachment = new MimeAttachment();
   myMimeAttachment.content = "vro generated content\r\nhol-1821-5";
   myMimeAttachment.mimeType = "text/plain";
   Server.createResourceElement("HOL-1821-5", "my-vro-generated-resource.txt", myMimeAttachment);
   ```

3. Click on the X to close the window, and then click on **Save and close** (not shown)
Launch the Workflow

1. Select the workflow **Create Resource Element** that was just created
2. Click on **Start Workflow** button

Wait for the workflow to finish (green check mark)

View the Newly Created Resource Element

1. Click on **Resources** tab
2. Expand **HOL-1821-5** folder
3. Click on **my-vro-generated-resource.txt** Resource Element
4. Click on **Viewer** tab

The content matches what was in the code from the previous step. This can be easily changed to integrate user inputs or data from other systems.
Managing Resource Elements

After a Resource Element is added to vRealize Orchestrator, it can be updated. This can be done from the Client or from a workflow.

Update Resource with vRealize Orchestrator Client

1. Click on Resources tab
2. Select the resource to update
3. Click the Edit button
Change Version of the Resource

Each object in vRealize Orchestrator is versioned (Workflows, Configuration Elements, Resource Elements, and so on.) However, when an update is done from the Client, the version number is not automatically incremented. That is why it needs to be done manually before uploading the new content.

1. Select **General** tab
2. Click on the last number in the Version row
3. Add a **comment** for the version change
4. Click **OK**
5. Click on **Save and close** (Not shown)
Update the Resource Element with vRealize Orchestrator Client

1. Select `email-template.html` Resource Element
2. Right-Click and select **Update resource...**
3. Browse the folder `C:\hol\HOL-1821-05-CMP`
4. Select `email-template-2.html`
5. Click **Open**
6. Click **Update Resource** when prompted (not shown in the screenshot)
Update Resource with Workflow

1. Click on Workflows tab
2. Right-Click on folder HOL-1821
3. Click on New Workflow

Name the New Workflow

4. Enter "Update Resource Element" as workflow name
5. Click OK

Add Input

1. Select Inputs tab
2. Click on **Add parameter** button

**Assign Parameter Type**

1. Click on **arg_in_0**
2. Type **resource**
3. Click **OK**
4. Click on **string**
5. Type **ResourceElement** in the search box
6. Select **ResourceElement** from the list
7. Click **Accept** and the parameter should show a Name of **resource** and a Type of **ResourceElement**
Add Scriptable Task

1. Click to **Schema** tab
2. Drag and Drop **Scriptable task** item to the arrow in the schema
3. Click the **Undock** icon to open the lower panel in a new window
Bind Input

1. Click on the **Visual Binding** tab
2. Drag and drop **resource** from **In Parameters** to **IN**

Add Javascript Code to Create Resource Element

```javascript
var mimeAttachment = resource.getContentAsMimeAttachment();
mimeAttachment.content += "\r\n\nthis is new content";
resource.setContentFromMimeAttachment(mimeAttachment);
```
1. Click on **Scripting** tab
2. Copy the following code:

   ```javascript
   var mimeAttachment = resource.getContentAsMimeAttachment();
   mimeAttachment.content += "\r\n\r\nthis is new content";
   
   resource.setContentFromMimeAttachment(mimeAttachment);
   
   3. Click on the X to close the window, and then click **Save and close** (not shown)

**Launch the Workflow**

![Workflow launch image]

1. Select the workflow **Create Resource Element** that was just created
2. Click on **Start Workflow** button
**Fill Parameters**

1. Click on **Not Set** [Not shown]
2. Type **my-vro**
3. Select **my-vro-generated-resource.txt** by **double clicking** it - without a double click, the resource will not be selected!
4. Click **Submit**

Wait for the workflow to finish (green checkmark.)
View the Newly Created Resource Element

1. Click on **Resources** tab
2. Expand **HOL-1821-5** folder
3. Click on **my-vro-generated-resource.txt** Resource Element
4. Click on **Viewer** tab

The content has been successfully updated.
Using Resource Elements

Now let's use a Resource Element in a workflow. Previously, an email template was added as a resource element. Let's see how it can be used to send an actual email.

Create a New Workflow

1. Click on Workflows tab
2. Right-Click on folder HOL-1821
3. Click on New Workflow

Name the New Workflow

4. Enter Use Resource Element as workflow name
5. Click OK
Add Input

1. Select **Inputs** tab
2. Click on **Add parameter** button
Assign Type

1. Click on arg_in_0
2. Type resource
3. Click OK
4. Click on string
5. Type ResourceE in the Filter box
6. Select ResourceElement from the list
7. Click Accept and the parameter should show a Name of resource and a Type of ResourceElement
Add Input

1. Select Inputs tab
2. Click on Add parameter button

Assign Type

1. Click on arg_in_0
2. Type VM
3. Click Ok
4. Click on string
5. Type VC:VirtualMachine in the Filter box
6. Select **VC:VirtualMachine** from the list
7. Click **Accept** and the second parameter should show a Name of **VM** and a Type of **VC:VirtualMachine**

**Add Scriptable Task**

1. Click to **Schema** tab
2. Drag and Drop **Scriptable task** item to the arrow in the schema

**Add Workflow**

1. Enter **send em** in the search box to find the Send Email to Dev User workflow
2. Drag and drop the workflow **Send Email to Dev User** after the scriptable task
Bind Workflow Inputs

1. Click on Setup...
2. Click on Value (at the top)
3. Click on Promote

Select Scriptable Task Item
1. Click on **Scriptable task** workflow element
2. Click the **Undock** icon to open the lower pane in a new window

**Bind Input**

Ensure that the **Visual Binding** tab is selected.

1. Drag and Drop **resource** from **In Parameters** to **IN**
2. Drag and Drop **VM** from **In Parameters** to **IN**
3. Drag and Drop **subject** from **Out Attributes** to **OUT**
4. Drag and Drop **content** from **Out Attributes** to **OUT**
Add Javascript Code to Create Resource Element

1. Click on **Scripting** tab
2. Copy the following code:

```javascript
subject = "Machine info " + VM.name;
content = resource.getContentAsMimeAttachment().content;
content = content.replace("{{name}}", VM.name);
content = content.replace("{{cpu}}", VM.summary.config.numCpu);
content = content.replace("{{memory}}", VM.summary.config.memorySizeMB + " MB");
content = content.replace("{{ipAddress}}", VM.guest.ipAddress);
```

3. Click on the X to close the window, and then click **Save and close** (not shown)
Launch the Workflow

1. Select the workflow Create Resource Element that was just created
2. Click on Start Workflow button

Enter Resource Parameter

1. For resource parameter (first one), click on Not Set (Not shown)
2. Type email-te
3. Select email-template.html (making sure to double-click)
Enter VM Parameter

1. For VM parameter (second one), click on Not Set (Not shown)
2. Expand and Select vSphere vCenter Plugin > [https://vcsa-01a.corp.local:443/sdk](https://vcsa-01a.corp.local:443/sdk) > Datacenters > RegionA01 > vm > puppet-01a
3. Click Select
4. Click Submit

Wait for the workflow to finish (green checkmark.)

Launch Chrome
1. In the Windows Quick Launch task bar, click on the **Google Chrome** icon

**Launch Webmail**

1. Click on the **Webmail** bookmark in the **HOL Admin** folder

**Log In to devuser Mailbox**

1. Enter username: **devuser@rainpole.com**
2. Enter password: **VMware1!**
3. Click on **Login**

### View Email from Workflow

1. Click Refresh until an email appears in the inbox (This can take up to a minute)
2. Select the email in the email list
3. Check the value in the email

Information from the machine has been populated into the email template!

This show how a static vRO resource object can be easily used to generate dynamic content.

When finished reading the new email message, close the **Chrome** window (not pictured)
Conclusion

Resource Elements are very useful to store templates (HTML or XML,) scripts, or even images. It is very easy to use them from a workflow, to add content to them, and to use them in other systems like Emails, REST calls, and more.

You've finished Module 5

Congratulations on completing Module 5.

If you are looking for additional information on Resource Elements, try one of these:

• Go to http://bit.ly/2ukKQx4
• Or use your smart device to scan the QRC Code.

Proceed to any module below which interests you most. [Add any custom/optional information for your lab manual.]

• **Module 1 - vRealize Orchestrator Overview** (30 minutes) (Basic) Examine and understand the key features and architecture of the vRealize Orchestrator platform. Learn the methods for accessing and interacting with platform.

• **Module 2 - Creating a Basic Orchestrator Workflow** (30 minutes) (Basic) Learn how to create a basic workflow to resolve a simple yet routine task such as cloning a virtual machine or taking a snapshot.

• **Module 3 - Introduction to Parameters, Attributes and Scripting Objects** (30 minutes) (Advanced) Explore and learn about the objects and constructs that are used to extend the functionality of workflows. These include attributes, scripting objects and decision trees as well as the parameters they use.

• **Module 4 - Using Actions in vRealize Orchestrator** (30 minutes) (Advanced) Reuse your JavaScript code by turning JavaScript Functions into vRealize Orchestrator Actions, then use them repeatedly in multiple workflows. Actions represent individual functions that you use as building blocks in workflows and scripts.
• **Module 6 - Creating Advanced Orchestrator Workflows** (45 minutes) (Advanced) Test your Orchestrator skills by in this module where you will create an advanced workflow. Learn about applying best practices to workflow development.

• **Module 7 - Error Handling and Troubleshooting** (45 minutes) (Advanced) Discover the various ways to add error or exception handling to ease in troubleshooting and more resilient workflows. Learn about and create exception binding to your workflows.

**How to End Lab**

To end your lab click on the **END** button.
Module 6 - Creating Advanced Orchestrator Workflows (45 minutes)
Introduction

vRealize Orchestrator can be used to automate complex processes that require various steps, branching, and even looping.

In this module, conditional execution, loops, and asynchronous presentation are covered using real-life examples in relation with vRealize Automation. This module begins with details on how to implement these concepts in a workflow, and it is followed by a real-life example on how to request a vRealize Automation catalog item, wait for the provisioning to finish, and retrieve the newly created machine.

This type of use case is particularly useful for automated blueprint testing, or for provisioning load testing.

Note: Use drag and drop to enter code in scriptable tasks to avoid any mistakes.

Launch vRealize Orchestrator Client

If vRealize Orchestrator Client is already opened, this step and the next one can be ignored.

1. Double click on the vRealize Orchestrator Client icon on the desktop.
Login to vRealize Orchestrator Client

Use the following credentials

1. Host name: *vra-01a.corp.local:443*
2. User name: *administrator@vsphere.local*
3. Password: *VMware1!*
4. Click **Login**
Promoting User Input In Workflows

vRealize Orchestrator workflows allow you to validate input parameters. This way, the values have to match certain criteria before the workflow starts. In this lesson, basic validation types are covered including mandatory, min/max number, and dropdown.

Create a New Workflow

1. Make sure Design view is selected
2. Select Workflow tab
3. Right Click on HOL-1821 workflow folder
4. Click on New workflow

Name Workflow

1. Presentation Tutorial
2. Ok
1. Enter the name of the workflow: **Presentation Tutorial**
2. Click **OK**

### Add Input Parameters

1. Click on the **Inputs** tab

Three parameters will be added in total.

### Add Input Parameter: name

1. Click the **Add parameter** icon to add the first parameter to the list
2. Click the default parameter name **arg_in_0** to rename it
3. In the Choose attribute name... window, enter **name** for Attribute name
4. Click **Ok**
Add Input Parameter: list

1. Click the Add parameter icon to add the second parameter to the list
2. Click the default parameter name `arg_in_0` to rename it
3. In the Choose attribute name... window, enter `list` for Attribute name
4. Click Ok

Add Input Parameter: num

1. Click the Add parameter icon to add the second parameter to the list
2. Click the default parameter name `arg_in_0` to rename it
3. In the Choose attribute name... window, enter `num` for Attribute name
4. Click Ok
1. Click the **Add parameter** icon to add the third parameter to the list
2. Click the default parameter name **arg_in_0** to rename it
3. In the Choose attribute name... window, enter **num** for Attribute name
4. Click **Ok**

**Change num Attribute Type**

1. Click the **string** listing for the third parameter **num**
2. In the Select a type... window, select **number**
3. Click **Accept** to change the Type entry for this parameter to **number**
Add Property to name Parameter

1. Click on the **Presentation** tab
2. Select the **(string) name name** entry
3. In the pane on the right, click on the **Add property...** icon to add a property to this parameter

Make name Parameter Mandatory

1. In the Properties... window, select **Mandatory input**
2. Click **Ok**
Add List of Values to list Parameter

1. Select (string) list list entry on the left, and click the Add property... icon on the right (not shown)
2. In the Properties... window, select Predefined answers
3. Click Ok
Add Default Values

1. Click on the default value of Not set to open the Array of string window
2. Click the upper left corner of the Array of string window, and drag it down and to the right to resize it. Click the title bar of the window to move it up until the Cancel and Accept buttons are visible
3. Enter 3 total values into the New value field, clicking Insert value after each one. The values in order are Small, Medium, and Large
4. Click Accept
Add Minimum Value for num Parameter

1. Click the (number) num entry on the left, and click the Add property... icon on the right (not shown)
2. In the Properties... window, select Minimum number value
3. Click Ok
Add Maximum Value for num Parameter

1. Click the **(number) num** entry on the left, and click the **Add property...** icon on the right (not shown)
2. In the Properties... window, select **Maximum number value**
3. Click **Ok**

These properties have default values of 0.0 for Minimum number value, and 100.0 for Maximum number value. These default values will suffice for this exercise.
Save and Close the Workflow

1. Click **Save and close** to close the workflow

Continue Despite Validation Errors

Unlike the previous modules, attempting to save and close this workflow will result in a Validation warning. This is because the 3 parameters defined in the previous steps are not actually used by this workflow. Click the **View details** button if you wish to see the specific information.
1. Since we know that we can proceed despite this warning, click **Continue anyway**

**Run the Workflow**

1. Click the **Start workflow...** icon
Observe Behavior for Mandatory Parameter

The name parameter has been designated as mandatory, as noted by the red * next to the parameter name. Attempting to skip the name parameter will generate an error.

1. Click the **name** text box, but do not enter any value
2. Click another field (such as the **num** text box) and observe the error message at the top of the parameter list
3. Enter a name, and the error message will be dismissed.
Observe Behavior for List of Values

The list parameter has a predefined set of values. For this reason, the list parameter on this screen is a dropdown menu of the values defined previously, and only these values may be chosen.

1. Choose a value for the list parameter from the dropdown menu
Observe Behavior for Numerical Range

The num parameter has both a minimum and a maximum defined value, with the minimum of 0.0 and the maximum of 100.0.

1. Enter a **number greater than 100.0** and observe the error message at the top of the parameter list.
2. Click on **Cancel** button

**To go Further**

Many more presentation properties are available, not limited to but including hide/show, dropdown list from Actions, and choices related to other fields. Feel free to take more time to explore all of the possibilities.
Using Conditional Execution and Loops

Conditional execution and loops are key constructs in vRealize Orchestrator. In a typical programming language, they are usually identified by a keyword such as for, while, if, then, else, and so on. In this lesson, the same constructs are used, but by dragging and dropping available workflow elements onto the schema of a workflow.

Create a New Workflow

1. Make sure Design mode is selected
2. Select Workflow tab, if it is not already selected from the previous exercise
3. Right Click on HOL-1821 workflow folder
4. Click on New workflow

Name New Workflow

1. Enter the name My First Loop
1. Enter a name for the workflow: **My First Loop**
2. Click **Ok**

### Add Decision box

1. Select **Schema** tab
2. Select the **Generic** section from the list on the left
3. Click the **Decision** object, and drag it onto the arrow between the Start and End objects, noting the new branch added by the Decision object

A decision box is the equivalent of an IF, ELSE conditional branch. The green arrow is the path if the condition is true, and the red dotted arrow is the path if the condition is false.
Create New Decision Parameter

1. Select the Decision tab in the lower left corner
2. Click Not set (NULL) to open the Chooser... window
3. In the new window, click Create parameter/attribute in workflow
Configure Decision Parameter

1. In the Name field, enter **counter**
2. Change the Type to **number**
3. Enter a **0 (zero)** in **Value** field
4. Click **Ok**
Configure Decision

1. In the Decision tab in the lower pane, click the dropdown menu and change the value to **greater or equals**
2. Enter a value of **10** in the blank text field

The counter attribute defined in the previous step is used to define how the **Decision** behave:

- If the counter value is **less than** 10, the path will be the **red dotted** one.
- If the value is **greater than or equal** to 10, the path will be the **green** one.

Add Increase counter Object

1. Select **Basic** category from the list on the left
2. Click on the **Increase counter** object and drag it into the workflow, making sure to drag it to the dotted red line under the Decision object
3. Click the **Setup...** button in the upper right to proceed with the binding of the parameters of the element.
1. Click **Promote** to import all parameters from the Increase counter object to the workflow

This automatically binds the attribute **counter** to the IN and the OUT parameters of the element.
Delete Workflow Branch

In order to make a loop, the second end of the workflow must be deleted. An End element can't be deleted by selecting it, so instead, the arrow preceding the element must be deleted.

1. Right-click on the blue arrow under the Increase counter object
2. Click on Delete

The End element should disappear with the arrow.
1. Select **Generic** category
2. Drag and drop **Scriptable task** element to **Increase counter** element in the schema, which will add the task to the right
3. Click the **Undock** icon to open the Scriptable task object in a new window
Bind Parameter to Scriptable Task

1. Click the **Visual Binding tab**
2. Drag and drop *counter* from **In Attributes** to **IN**
Add Script to Log Counter Value

1. Select the **Scripting** tab
2. Enter the following code:

   ```java
   System.log("Current counter: " + counter);
   ```

3. Click the **X** to close the window and return to the Schema pane
Add Next Item to Scriptable Task

1. Move the mouse over the **Scriptable task**, then over the **blue arrow** that appears along the right side of the object
2. Drag and drop the **blue arrow** to the **Decision** box

(Note: if necessary, click the dropdown menu above the schema to increase the zoom % from the present 50.)

The **loop** is created. It is very similar to the following code in other languages:

```java
for(counter=0; counter <= 10; counter++){
    log("Current counter: "+counter);
}
```
Run the Workflow

1. Click on **Run** button
2. Click the Logs tab to view the output of the workflow execution
3. Observe the log output showing the counter value increasing until it reaches 10, at which time the workflow ends
Create an Infinite Loop

Sometimes, an error in developing a workflow can lead to an infinite loop. Let's see how it can happen.

1. Click on the **Increase counter** object in the workflow schema
2. Select the **OUT** tab in the lower pane
3. Click the **counter** parameter to select it
4. Click on the **Remove parameter** button
Run and cancel infinite loop

With the counter output parameter removed, the value of the counter is never updated. The condition in the decision will thus always be false (it is never greater or equal than 10), and this workflow will result in an infinite loop.
To observe this, run the workflow.

1. Click **Run** to execute the workflow
2. Observe the log, which will continue to show new messages with a counter value remaining at 0
Cancel Workflow

This workflow will continue to run indefinitely. In order to stop the workflow execution, the only way is to cancel the workflow.

1. Click the **Cancel** icon
2. In the Cancel Workflow window, click the **Yes, cancel it** button
1. Click on **Save and close** button

This exercise is complete. Workflow elements available out-of-the-box allow very easily to have conditional execution and to create visual loops in any workflows. As always, with loops there is a risk of an infinite loop with an improperly configured workflow, but it can be stopped by canceling the workflow manually.
Bring it together: Sample Advanced Workflow

Now let's take everything we've learned so far, and use it to create an advanced workflow within vRealize Orchestrator. This workflow provisions a vRealize Automation machine and waits for the provisioning task to complete. Then it retrieves the Item.

Remember, **Drag and Drop** can be used to enter code from HOL manual to workflow **Scriptable tasks**.

Create a New Workflow

1. Make sure **Design** mode is selected
2. Select **Workflow** tab, if it is not already selected from the previous exercise
3. Right Click on **HOL-1821** workflow folder
4. Click on **New workflow**
Name New Workflow

1. Enter "Create Machine and Wait" as workflow name
2. Click OK

Reset Schema Zoom

After completion of the previous exercise, the schema section may be set to a lower zoom level than usual. Before starting, the zoom can be changed.

1. Click on the Schema tab
2. Click on the dropdown menu in the top menu, and change the value to 100
Add an Existing Workflow

To begin, an existing out-of-the-box workflow is leveraged to considerably save time and effort.

1. Select **All Workflows** from the list on the left
2. Open the **administrator @ vra-01a.corp.local** menu (not shown) and navigate to **Library -> vRealize Automation -> Requests**
3. Click on the **Request a catalog item with provisioning request** workflow, and drag it onto the arrow between the start and end objects
4. Click the **Setup...** button in the “Do you want to add the activity's parameters...” message that appears above the schema
Bind Parameters

1. For the request parameter under Promote Workflow Output Parameters, change the Mapping Type to Local variable
2. Click Promote

Create a Loop to Wait

Now that the request has been configured to request a vRealize Automation catalog item, a loop is required to wait for the end of the provisioning of the machine. In addition, a timeout will be implemented to fail the workflow if the machine takes too much time to provision.
Add Decision Box

1. Click on the **Generic** menu item in the list on the left
2. Click on **Custom decision** and drag it onto the schema. Make sure to drag the Custom decision object onto the arrow to the right of the workflow, as shown

Name Custom Decision

1. Click on the newly added **Custom decision** object
2. In the lower pane, for Name enter **Request done?**
Bind Decision Inputs

1. Select the **IN** tab in the lower pane (note that upon changing tabs, the Custom decision object name has changed)
2. Click the **Bind to workflow parameter/attribute** icon
3. In the Chooser... window that opens, click the checkbox next to **request**
4. Click **Select**

Configure Decision

1. Select **Scripting** tab
2. Enter the following code:

   ```java
   return (request.getExecutionStatus().value() == "STOPPED")
   ```
By setting this condition, the green path will be used only when the execution of the request is stopped (which means the provisioning has either succeeded or failed.)

Add Sleep Box

1. Select **Basic** category
2. Drag and Drop **Sleep** workflow element on the **false** (red dotted line) of the decision box, below the Request done? custom decision
3. Click the **Setup...** button
Configure Sleep Box

1. Click on Value radio button
2. Enter value 20 (for 20 seconds)
3. Click on Promote button

This sleep box allows the workflow to wait 20 seconds between each request status check.
Add Increase Counter Element

1. Click on Basic category
2. Drag and Drop Increase counter workflow element after the Sleep element added previously (but do not click the Setup... button this time)

A counter is required to manage a timeout in the loop.

Set Source Parameter for Input

1. Select IN tab
2. Click on not set for the unique Local Parameter "counter"
3. Click on the link Create parameter/attribute in workflow in the Chooser... window (not shown)
Create Attribute Value for Counter

1. Enter 0 (zero) in the Value field
2. Click on Ok button
Set Source Parameter for Output

1. Select **Out** tab
2. Click on **not set** for the unique Local Parameter "counter"

Set Source Parameter for Output (Continued)
1. Select attribute **counter** in the list
2. Click on **Select** button

**Remove end of the workflow**

1. Right click on the **blue arrow** after the counter element
2. Click on **Delete**

Note: If clicking on the arrow is difficult, it is possible to change the zoom level using the dropdown at the top, as done at the start of this exercise.
Add Decision Element

1. Select **Generic** category
2. Drag and drop the **Decision** element on top of the **Increase counter** workflow element.
   The Decision element will be added to the right of the Increase counter element.

Rename Decision

1. Select the **Info** tab in the lower pane
2. For Name, enter **no timeout?**
Configure Decision Box

1. Select **Decision** tab
2. Click on **Not set (NULL)** link
3. In the Chooser... window, select **counter** attribute from the list
4. Click on **Select** button

Set Timeout Threshold

1. Select **smaller or equals** from the dropdown
2. Enter **30** in the text box (30 x 20sec sleep = 10min timeout)
Add Scriptable Task

1. Select **Generic** category, if it is not already selected
2. Drag and drop the **Scriptable task** element onto the **Decision** workflow element. The new Scriptable task will appear below the Decision element.

**Arrange Workflow Schema**

---

**HOL-1821-05-CMP**

Page 178
1. Click and drag the **Scriptable task** element above the decision box. Note that the arrow follows the element as you drag it to the new location.

**Bind IN & OUT Parameters**

1. Click on **Scriptable task**, if it is not already selected
2. Click the **Undock** icon in the lower right, to exand the lower pane into a new window
1. Select **Visual Binding** tab
2. Drag and drop **request** attribute from **In Attributes** to **IN**
3. Drag and drop **request** attribute from **Out Attributes** to **OUT**
Add Javascript

1. Select **Scripting** tab
2. Enter the following code (if drag and drop is used, make sure to keep the focus in the scripting tab all the time):

   ```javascript
   //Refresh request
   var host = vCACCAFEEntitiesFinder.getHostForEntity(request);
   request = vCACCAFEEntitiesFinder.getCatalogItemRequest(host, request.id);
   //Log state and execution status
   System.log(request.getStateName() + " (" + request.getExecutionStatus().value() + ")");
   ```

3. Click the **X** to close the window and return to the schema pane

   This step updates the `request` object to make sure that it contains an updated version with the latest status. Depending of the plugin used, this step may or may not be required. There is no golden rule, so the only way to know if this is needed or not is to test the behavior.
Link to Next Item

1. Move your mouse over **Scriptable task**, then on the **blue arrow**
2. Drag and drop the **blue arrow** to the decision box "Request done?"

Add Exception End

1. Select **Generic** category, if it is not already selected
2. Scroll to find **Throw exception**
3. Drag and drop **Throw exception** to the decision box **no timeout?**
Add Attribute for Exception Binding

1. Select Exception tab
2. Click on the link Not set next to Throw exception binding
3. In the first window that appears, click on the link Create parameter/attribute in workflow to open a second window
4. Click OK, as all of the defaults in this window will suffice
1. Scroll to the top of the Generic category to find **Scriptable task**
2. Drag and drop **Scriptable task** to the **red dotted arrow** between the decision box and the exception
Add IN Parameters

1. In the lower pane, select the **IN** tab
2. Click on **Bind to workflow parameter/attribute** button
3. Mark the checkbox for **request**, **sleepTime**, **counter** attributes
4. Click **Select**
Add OUT Parameters

1. Select the **OUT** tab
2. Click on **Bind to workflow parameter/attribute** button
3. Mark the checkbox for **errorCode** attributes
4. Click **Select**

Add javascript

```javascript
errorCode = "Request " + request.getRequestNumber ()
+ " reached the timeout of "
+ (counter * sleepTime) + " seconds.";
```
1. Select **Scripting** tab
2. Enter the following code:

```java
errorCode = "Request " + request.getRequestNumber() + " reached the timeout of " + (counter * sleepTime) + " seconds.";
```

### Add Scriptable Task to Get Resource Item

1. Drag and drop an additional **Scriptable task** element onto the **arrow** between the decision box and the end of the workflow
Add IN Parameters

1. In the lower pane, select the **IN** tab
2. Click on **Bind to workflow parameter/attribute** button
3. Mark the checkbox for **request** attribute
4. Click **Select**
Add OUT Parameters

1. Select the OUT tab
2. Click on **Bind to workflow parameter/attribute** button
3. In the new window, click on **Create parameter/attribute in workflow** to open an additional window
4. Enter `resource` in the textbox **name**
5. Enter `CatalogResource` in the **Filter** textbox
6. Select `vCACCAFE:CatalogResource` from the results
7. Select **Create workflow OUTPUT PARAMETER with the same name** radio button
8. Click **Ok**
Add javascript

1. Select **Scripting** tab
2. Enter the following code (if drag and drop is used, make sure to keep the focus in the scripting tab all the time):

```javascript
//Get Consumer Resource Service (to query items)
var host = vCACCAFEEntitiesFinder.getHostForEntity(request);
var service = host.createCatalogClient().getCatalogConsumerResourceService();

//Prepare Filter
var filter = new Array();filter[0] = vCACCAFEFilterParam.equal("request/id",
vCACCAFEFilterParam.string(request.getId()));
filter[1] = vCACCAFEFilterParam.equal("resourceType",
vCACCAFEFilterParam.string("Infrastructure.Virtual");
var query = vCACCAFEDataQuery.query().addFilter(filter);

// get items
var items = service.getResourcesList(new vCACCAFEPageOdataRequest(query));
resource = items[0];
```

This code allows the task to retrieve the newly provisioned item from the request.

**Run the Workflow**

1. Click the **Run** button to execute the workflow. This will open the Start Workflow window and prompt for input.
Enter Workflow Inputs

1. For Description, enter **automated creation**
2. Click on **Not set** under Catalog item to open a new window
Choose a vRealize Automation Catalog Item to Request

1. In the Select window, navigate to vRealize Automation -> vRA - DevMgr [https://vra-01a.corp.local] [vsphere.local] -> Catalog and Select CentOS
2. Click on Select to close this window
3. In the Start Workflow window, click on Submit (not shown) to begin the workflow execution
Monitor the Workflow

The workflow execution can take 5 to 15 minutes to complete (depending on the load on the HOL platform).

Observe how the schema is flowing, and the log messages as they appear in the Logs tab.

Continue to the next step before the workflow finishes.

Look at the variables
Observe how variables are changing, especially the counter:

1. On the right, click on the **Variables** tab
2. View the **counter** variable, and its current value

**Open Chrome Browser**

1. In the Windows Quick Launch task bar, click the **Google Chrome** icon to open the browser

**Log In to vRealize Automation**

1. Log in to vRealize Automation as **devmgr** with password **VMware1!**
2. Click **Sign in**
Observe Status of Catalog Request

1. Navigate to the Requests tab
2. Note the request for a CentOS item, with the description of automated creation as noted upon requesting in the workflow.

The request will show a Status of In Progress until provisioning of the machine is complete. When it is complete, the vRealize Automation request will show a Status of Successful or Failed.

Look at the Final Variables

Return to vRealize Orchestrator, and observe the status of the completed workflow. Note that in the Variables tab, the resource parameter is now set to a value of the machine’s name (this name was generated as part of the vRealize Automation blueprint requested by this workflow).
Look at the Log Again

The workflow log shows a final status of **Successful (STOPPED)**.

**Destroy Machine in vRealize Automation**

1. Switch to vRealize Automation in the Chrome browser (not shown,) and navigate to the **Items** tab
2. Select the line containing the **CentOS** item (do not click on the Name itself, or you will be taken to the Item Details screen)
3. Click **Actions**
4. From the dropdown menu, select **Destroy**

**Destroy Machine in vRealize Automation (Continued)**

![Image of vRealize Automation interface](image)

1. Click **Submit** to submit the destroy request
2. Click **OK** (not shown) when notified of successful request submission

This exercise is complete. Interacting with vRealize Automation is easy using out-of-the-box workflows and basic concepts, such as loops. This allows to additional automation on top of the machine provisioning by creating a new machine, returning a reference to the newly created machine, and then executing additional actions with the machine such as configuration management, software installation, and more. For more examples of additional options beyond machine creation, see the other vRealize Automation hands-on labs.
Asynchronous Workflows

Asynchronous workflows allow multiple tasks within a workflow to be launched without waiting for an individual task to complete! This is a good way to trigger multiple actions in parallel, or to track different parts of a process as separate workflow tokens.

Create a New Workflow

1. Make sure **Design** mode is selected
2. Select **Workflow** tab, if it is not already selected from the previous exercise
3. Right Click on **HOL-1821** workflow folder
4. Click on **New workflow**

Name the New Workflow

1. Asynch
1. Enter the name of the workflow: **Asynch**
2. Click **Ok**

**Add Asynchronous Workflow**

1. Click on the **Schema** tab
2. Select the **Generic** category from the list on the left, if it is not already selected
3. Scroll down until **Asynchronous workflow** element appears.
4. Click on **Asynchronous workflow** and drag it to the arrow between the start and end elements. Once this is done, a Search box will pop up.

**Choose Workflow**

1. Enter **Create Machine and** in the Search field
2. In the results, select the **Create Machine and Wait** workflow (created in the previous exercise) by **double-clicking**

**Bind Parameters**

1. Click the **Setup...** button in the upper right corner of the schema after choosing the Create Machine and Wait workflow
2. In the Promote Workflow Input/Output Parameters window, click **Promote** to accept all of the default values

**Run the Workflow**

1. Click **Run** to execute the workflow and open the Start Workflow window
Enter Workflow Input

Notice that the input prompt is the same as the previous exercise. The same workflow is executed, but this time it is done from a parent workflow and in an asynchronous manner.

1. For Description, enter **automated creation**
2. Click **Not set** under Catalog item
Choose a vRealize Automation Catalog Item to Request

1. In the Select window, navigate to vRealize Automation -> vRA - DevMgr [https://vra-01a.corp.local] [vsphere.local] -> Catalog and Select CentOS
2. Click Select to close this window
3. In the Start Workflow window, click Submit (not shown) to begin the workflow execution

Save and Close

1. Click on Save and close to close this workflow
Observe Variables for Completed Workflow

1. In the Workflows list, expand the Asynch workflow to observe the recent run
2. Note the green checkmark indicating that this workflow execution was successful. Click on the specific Asynch item to view the results
3. Select the Variables tab on the right
4. Observe the wfToken attribute, it is set to Create Machine and Wait and represent the workflow asynchronously executed.

Look at Other Workflow

1. Expand the Create Machine and Wait workflow to observe recent activity for that workflow. Note that this workflow is in the HOL-1821 folder.
2. Locate the most recent run of this workflow. Note that it is still running, while the Asynch workflow has already completed.

Open Chrome Browser
1. In the Windows Quick Launch task bar, click the Google Chrome icon to open the browser

Log In to vRealize Automation

1. Log in to vRealize Automation as devmgr with password VMware1!
2. Click Sign in
Destroy Machine in vRealize Automation

1. Navigate to the **Items** tab
2. Select the line containing the **CentOS** item (do not click on the Name itself, or you will be taken to the Item Details screen). Note that the item takes couple of minutes to appear in the item list.
3. Click **Actions**
4. From the dropdown menu, select **Destroy**
1. Click **Submit** to submit the destroy request
2. Click **OK** (not shown) when notified of successful request submission

The exercise is complete. To go further, it is possible to use a loop to wait for the end of the asynchronous workflow by monitoring the workflow token state value.
Conclusion

vRealize Orchestrator can be used to automate complex processes requiring polling for status, launching multiple tasks at the same time (or asynchronously,) and using advanced input presentation. In conjunction with vRealize Automation, it can be very useful to automate processes around machine provisioning by using the power of vRealize Automation for the creation and the flexibility of Orchestrator for input parameters or third party integrations.

You've finished Module 6

Congratulations on completing Module 6.

If you are looking for additional information on vRO Coding Guidelines, try one of these:

- Or use your smart device to scan the QRC Code.

Proceed to any module below which interests you most.

- **Module 1 - vRealize Orchestrator Overview** (30 minutes) (Basic) Examine and understand the key features and architecture of the vRealize Orchestrator platform. Learn the methods for accessing and interacting with platform.
- **Module 2 - Creating a Basic Orchestrator Workflow** (30 minutes) (Basic) Learn how to create a basic workflow to resolve a simple yet routine task such as cloning a virtual machine or taking a snapshot.
- **Module 3 - Introduction to Parameters, Attributes and Scripting Objects** (30 minutes) (Advanced) Explore and learn about the objects and constructs that are used to extend the functionality of workflows. These include attributes, scripting objects and decision trees as well as the parameters they use.
• **Module 4 - Using Actions in vRealize Orchestrator** (30 minutes) (Advanced) Reuse your JavaScript code by turning JavaScript Functions into vRealize Orchestrator Actions, then use them repeatedly in multiple workflows. Actions represent individual functions that you use as building blocks in workflows and scripts.

• **Module 5 - Creating and Managing Resource Elements** (45 minutes) (Advanced) Discover how to use external objects such as scripts, XML, or HTML files in your workflows.

• **Module 7 - Error Handling and Troubleshooting** (45 minutes) (Advanced) Discover the various ways to add error or exception handling to ease in troubleshooting and more resilient workflows. Learn about and create exception binding to your workflows.

### How to End Lab

To end your lab click on the **END** button.
Module 7 - Error Handling and Troubleshooting (45 minutes)
Introduction

Error handling and logging are very important concepts in vRealize Orchestrator for the creation of reliable workflows, to assist with workflow troubleshooting, and ultimately to make infrastructure more robust.

In this module, error handling, troubleshooting, and logging are covered. This lesson highlights how easy it is to catch errors, execute remediation actions, and finally to automate everything in a reliable fashion using vRealize Orchestrator.

Launch vRealize Orchestrator Client

If vRealize Orchestrator Client is already opened, this step and the next one can be ignored.

1. Double click on the vRealize Orchestrator Client icon on the desktop.
Login to vRealize Orchestrator Client

Use the following credentials

1. Host name: **vra-01a.corp.local:443**
2. User name: **administrator@vsphere.local**
3. Password: **VMware1!**
4. Click **Login**
Error handling and why is it important?

Exception handling catches any errors that occur when a schema element runs. Exception handling defines how the schema element behaves when the error occurs.

All elements in a workflow, except for decisions and start and end elements, contain a specific output parameter type that serves only for handling exceptions. If an element encounters an error during its run, it can send an error signal to an exception handler. Exception handlers catch the error and react according to the errors they receive. If the exception handlers you define cannot handle a certain error, you can bind an element's exception output parameter to an Exception element, which ends the workflow run in the failed state.

This is important because when your workflows do fail, exception handling allows them to fail as gracefully as possible and it provides meaningful error messages.
Adding Error Handling to workflows

To learn more about how to handle errors in vRealize Orchestrator workflows, let's build a basic workflow and add error handling to it.

Adding New Workflow

1. Right click the HOL-1821 workflow folder
2. Select New workflow

Naming a Workflow

1. Type My Workflow Error Handling for the workflow name
2. Click the Ok button
Adding the Create a simple virtual machine Workflow

1. Select the **Schema tab**
2. Click **All Workflows** and browse to **Library > vCenter > Virtual Machine management > Basic > Create simple virtual machine**
3. Drag the **Create simple virtual machine** between the Start and End workflow elements
4. Click **Setup**

By clicking setup, this is allowing vRealize Orchestrator to bind the inputs/outputs from Create simple virtual machine to the parent workflow.
Adding the Create a simple virtual machine Workflow

Look over the input and output parameters that are being added to the workflow.

1. Click **Promote**
Adding the System Error Element to a Workflow

1. Browse the **Log** slider and Select **System Error**
2. Drag the **System Error** Element on to the Create simple virtual machine workflow

By dragging the System Error Element onto the workflow, you are linking it to the workflow. Notice how a red dotted line is created from the workflow to the system error. Also, the System Error element is connected to a separate End workflow element. This shows that if the the workflow errors out, it will be logged and the workflow will end.

3. Click **Setup**

By clicking setup, this is allowing vRealize Orchestrator to add the inputs/outputs from System Error to the master workflow. This allows you to avoid having to enter them manually, further simplifying the error handling process.
Adding the System Error Element to a workflow

1. Select the **Value** radio button for the **text** parameter
2. Select **Input value** for the text parameter
3. Type **The workflow has failed, please contact support.** in the Enter Text field

The text parameter allows you to supply the end user with more information when the workflow fails.

4. Click the **Ok** button
5. Click **Promote**
Moving Parameters to Attributes

1. Select the **Inputs** tab
2. Select the **vmGuestOS** row (do not click on the parameter name itself, or you will open a new window to rename the parameter)
3. Click the **Move as attribute** icon
4. Repeat the above tasks for these additional parameters: **vmFolder, vmResourcePool, vmHost, vmNetwork, and vmDatastore**

Configuring Attributes

1. Select the **General** tab
2. Scroll to the bottom

Configuring Attributes

The reason you have moved many of these parameters to attributes is so that you can configure preset values, and end users are not burdened with entering these values at provisioning time. Attributes are read/write parameters that are constants for the duration of the workflow.

1. Click Not Set to set the value for vmGuestOS

Configuring Attributes

1. Type **centos** in the Search field and press **Enter**
2. Double click **centos64Guest** to select it
Configuring Attributes

1. Click **Not Set** to set the value for vmFolder
2. Browse to **vCenter > https://vcsa-01a.corp.local:443/sdk > Datacenters > RegionA01 > vm**
3. Click the **Select** button
Configuring Attributes

1. Click **Not Set** to set the value for `vmResourcePool`.
2. Browse to **vCenter > https://vcsa-01a.corp.local:443/sdk > Datacenters > RegionA01 > host > RegionA01-COMP01 > Resources**.
3. Click the **Select** button.
Configuring Attributes

1. Click **Not Set** to set the value for vmHost.
2. Browse to **vCenter > https://vcsa-01a.corp.local:443/sdk > Datacenters > RegionA01 > host > RegionA01-COMP01 > esx-01a.corp.local**
3. Click the **Select** button.
Configuring Attributes

1. Click Not Set to set the value for vmNetwork.
2. Browse to vCenter > https://vcsa-01a.corp.local:443/sdk > Datacenters > RegionA01 > network > none (since we will not require any virtual machine network connectivity for this exercise).
3. Click the Select button.
Configuring Attributes

1. Click **Not Set** to set the value for vmDatastore
2. Browse to **vCenter > https://vcsa-01a.corp.local:443/sdk > Datacenters > RegionA01 > datastore > RegionA01-ISCSI01-COMP01**
3. Click the **Select** button
Running a workflow

1. Select the **Schema** Tab
2. Click the **Run Workflow** icon

You are going to do a test run of the workflow to see the effects of adding the System Error element.
Providing Inputs for a workflow

1. Type **Finance** for the virtual machine name
2. Type **5** for the amount of storage
3. Type **512** for the amount of memory
4. Type **.5** for the number of CPUs

Note you are putting .5 for the CPU count (which is invalid) to test the results of a failed workflow run with the added System Error element.

5. Select **Yes** for making the disk thin-provisioned
6. Click the **Submit** button
Reviewing the logs

1. Click the **Logs** tab

Note that the workflow failed, and a few things were logged:

- The VMWare provided error
- The additional user friendly text you provided

This is great, but what if you want to change an incorrect CPU count to 1 CPU if the user provided an invalid value, and proceed with the workflow? Basically catch the error, verify if it is related to CPU count, change the value, and try the build again. This is a good example of error handling, and you will be building this into your workflow next.
Adding a Decision Element

1. Browse Generic and Select **Decision**
2. Drag the **Decision** Element on to the line between the Create simple virtual machine workflow and the System Error Element

By dragging the **Decision** Element onto the line it is linked to the other elements. The red dotted line is created from the workflow to a End Workflow element. The Decision Element is connected to the System Error element with a green line.

As explained in **Module 6**, Decision Elements takes an input, if it matches the criteria it will use the green path, if not, it will use the red dotted path.
Adding a Decision Element

If the Decision Element is false it needs to be connected to the System Error, so let's do some cleanup.

1. Right click the red dotted line between the Decision Element and a End Workflow Element
2. Select Delete

This will delete the red dotted line and the End Workflow Element that is connected to it.
Adding a Decision Element

If the Decision Element is false it needs to be connected to the System Error, so let's do some cleanup.

1. Right click the **green line** between the **Decision** Element and the **System Error** Element
2. Select **Delete**
Adding a Decision Element

Finally you now can connect the red dotted line, or false to the System Error Element

1. Hover over the **Decision** Element and drag the **red arrow** to the **System Error** Element
Editing the Decision Element

Now you will want to give the decision some criteria to look for.

1. Click on the **Decision** Element to open the edit window in the lower pane
Selecting decision criteria

Now that the decision element has a string value to check, you need to tell it what to look for.

1. Click the drop down and select contains
2. Type numCPUs in the value field
The decision is set to look for the errorCode parameter that includes the string value "numCPUs".

**Moving a parameter to an attribute**

1. Select the **Inputs** tab
2. Select the **vmNbOfCpus** parameter row
3. Click the **Move as attribute** icon

Input parameters are read-only and cannot be changed once an input value is given, attributes are read/write. The value of this parameter may need to be updated based on the outcome of the decision element.
Adding an Input Parameter

Let's add an input parameter to have the end user provide the CPU count.

1. Select the Inputs tab
2. Click the Add a parameter icon
3. Click on arg_in_0 to provide a name (not shown)
4. Type vmCPU for the Attribute name
5. Click the Ok button
6. Click on string to change the type (details on next step)

Parameter type needs to be changed to number because this value will be passed to vmNbOfCpus attribute, which is also of type number and type must match in vRO.
Adding an Input Parameter

1. Select **number**
2. Click the **Accept** button

Adding an Input Parameter

1. Double click on the Description field of the vmCPU parameter and type **Virtual Machine CPU count**
Adding a Scriptable Task

1. Select the **Schema** tab
2. Browse Generic and Select **Scriptable task**
3. Drag the **Scriptable task** Element on to the line between the Start workflow element and the Create simple virtual machine workflow

You will use this Scriptable task to convert the `vmCPU` parameter to the `vmNbOfCpus` attribute.

Naming a Scriptable Task

1. Select the **Info** tab
2. Type **Input CPU count** for the Name
Adding an Input Parameter

1. Select the **IN** tab
2. Click the **Bind a workflow parameter** button
3. Select the **vmCPU** parameter
4. Click the **Select** button
Adding an Output Parameter

1. Select the **OUT** tab
2. Click the **Bind a workflow parameter** button
3. Select the **vmNbOfCpus** parameter
4. Click the **Select** button

Adding a Script

1. Select the **Scripting** tab
2. Type the following in the scripting window

```plaintext
vmNbOfCpus = vmCPU;
```
This script will convert the vmNbOfCpus attribute to the value of the input parameter vmCPU.

**Adding a Scriptable Task**

Now you want to add additional Scriptable task to change the value of the vmNbOfCpus attribute to 1 when it is received from the decision element.

1. Select the **Schema** tab  
2. Browse Generic and Select **Scriptable task**  
3. Drag the **Scriptable task** Element on to the decision element and it will be added to the schema with a green line that comes from the decision element.

This links the scriptable task to the decision element. When the criteria is met in the decision element, the workflow continues on to the scriptable task.
Naming a Scriptable Task

1. Select the **Info** tab
2. Type **Change CPU Count to 1** for the Name

Adding an Output Parameter

1. Select the **OUT** tab
2. Click the **Bind a workflow parameter** button
3. Select the **vmNbOfCpus** parameter
4. Click the **Select** button

**Adding a Script**

1. Select the **Scripting** tab
2. Type the following in the scripting window

```
vmNbOfCpus=1;
```

This script sets the value of vmNbOfCpus to 1.
Linking a Scriptable Task

1. Select the Schema tab
2. Hover over the scriptable task and drag the blue arrow from Change CPU Count to 1 to the Create a simple virtual machine workflow

Save and Close

1. Click the Save and close button
Testing the Workflow

1. Browse to **HOL-1821 > My Workflow Error Handling**, if it is not already selected
2. Click the **Start workflow** icon
Providing Inputs

1. Type FinanceDB for the virtual machine name
2. Type 10 for the virtual disk size
3. Type 512 for the memory size
4. Select Yes for making the disk thin-provisioned
5. Type .5 for the virtual machine CPU count
6. Click the Submit button

You are testing the workflow just like before, only this time you have built in some error handling to correct an invalid CPU count value. The workflow should complete successfully this time.
Checking the Logs

1. In the window to the right make sure the **Schema** tab is selected
2. Click the **Logs** tab
3. Review the logs

What do you see? You should see that an error was thrown because there was an invalid CPU count, but your error handling task caught it and the workflow completed successfully.

Launching the Chrome Browser

1. From the Windows Quick Launch task bar, open the **Google Chrome** web browser.
Logging into the vSphere Web Client

1. Click on the Region A vCenter bookmark in the RegionA folder
2. Check the box for **Use Windows session authentication**

This will log you on with the administrator@corp.local credentials

3. Click the **Login** button (This may take a minute)
1. Locate the virtual machine FinanceDB in the inventory and click on the Summary tab (not shown)
2. Scroll down until the VM Hardware section is visible
3. Verify the CPU entry showing 1 CPU allocated to the FinanceDB virtual machine
Delete the Virtual Machine

1. Select FinanceDB virtual machine (if it is not already the case)
2. Click on Actions icon to show the dropdown list
3. Click on Delete from Disk
4. Click on Yes in the Confirm Delete dialog box (not shown)

Close the Chrome Browser

1. Click the X in the top right hand corner of the browser to close it
Basic Workflow Troubleshooting

vRealize Orchestrator provides various ways to investigate workflow failures. It can be done from the vRealize Orchestrator Client, or directly from the logs. In this lesson, troubleshooting from the Client is covered as well as very basic usage of Log Insight with the vRealize Orchestrator management pack.

Launch "workflow failure 1"

1. Switch to Design mode
2. Select Workflows tab
3. Expand HOL-1821
4. Select Workflow failure 1
5. Click Start workflow... icon
Observe the Failure

1. The workflow token from the failed run can be seen by the red x icon
2. Select Variables tab
3. Look at the Exception message

This is the first clue to understand what happened. Most of the time this message should be clear enough to have an understanding of what caused the workflow failure.
Find the Element that Failed

1. Select **Schema** tab
2. **Expand** the upper panel by **drag and dropping** the separator bar
3. Look at the **highlighted** item

The workflow element highlighted in red is where the exception happened.
Look at the Code

1. Double-click on the **Scriptable task** that was previously highlighted in red

Look at the code (continued)
1. The workflow **Workflow failure 1** should be now selected
2. Select **Scripting** tab
3. Look at the code, the first line is just a log, so the variable **sizes** must be involved in the failure

**Look at the Variable Value at Failure Time**

1. Click back on the **workflow instance** that failed
2. Lower pane should be expanded by drag and dropping the separator bar
3. Select **Variables** tab
4. Look at the variable **sizes**

This variable, which is an array, has not been initialized (Not Set in value). It explains why the exception message says "Cannot call method "push" of null" - in this case, null is the array.
Get More Details

1. Select **Logs** tab
2. Select **Debug** from the dropdown
3. Logs before the exception can help narrow down the exception and the root cause. That's why logging is very important. In the next exercise, we will review logging in more detail.
4. The exception message is written to the error log. It also helps to understand where this happened: In the workflow "Workflow failure 1", in the item 1, names "Scriptable task", at line 1.
5. It shows the full variable stack, so it is easy to see what the value of a variable was at the time of the exception.
Launch Workflow failure 2

1. Right-click on **Workflow failure 2**
2. Click on **Start workflow...**
Observe the failure

1. Select **Schema**
2. The workflow element where the exception occurred is highlighted.
3. In this case, the workflow that failed is not the root workflow, so it is possible to see the workflow stack.
Switch to Root Workflow

1. Click on **Workflow failure 2** button
2. The workflow element in **Workflow failure 2** which failed is highlighted
3. The variables are the variables from **Workflow failure 2**

This can be useful to track down the propagation of a bad value.
Switch Back to Child Workflow

1. Click on **Workflow failure 1** button
2. The variables are the variables from **Workflow failure 1**

This helps to understand how variables have flown from the root workflow to the child workflow.
Look at the Logs

1. Expand the lower pane by dragging and dropping the separator bar
2. Select **Logs** tab
3. This shows the variables for root workflow (Workflow failure 2)
4. This shows the variables for child workflow (Workflow failure 1)

In case of more workflows, all variables of all workflows are shown.

Open Chrome Browser

1. In the Windows Quick Launch task bar, click the **Google Chrome** icon to open the browser
Log In to Log Insight

1. Click on **vRealize Log Insight** link
2. Enter credentials
   - Username: admin
   - Password: VMware1!
3. Click on **Login** button
Select vRO Dashboard

1. Click on **Dashboards** tab
2. Scroll down to find **VMware - Orchestrator - 7.0.1+** content pack
3. Expand **VMware - Orchestrator - 7.0.1+** content pack
4. Click on **Workflows failures** dashboard
5. Select **Last 24 hours of data** from the dropdown
Look at the Dashboard

1. Scroll down the dashboard
2. Look at the **FAILED workflows grouped by WORKFLOW NAME** widget
Logging in Workflows

Despite multiple methods for workflow troubleshooting described in the previous exercise, logging is still very important in order to understand errors, and also to audit a workflow result.

Create a New Workflow

1. Right-Click on folder **HOL-1821**
2. Click on **New Workflow**

Note: If you do not see the HOL-1821 folder or the New workflow right click menu option, ensure that the client is in the **Design** view and that the **Workflows** tab is selected.
Name New Workflow

1. Enter "Logging" as workflow name
2. Click OK

Add Scriptable Task

1. Select Schema
2. Select Generic category
3. Drag and Drop Scriptable task to the blue arrow
Undock Lower Panel

1. Select **Scriptable task** workflow element
2. Click on little window button
1. Select **Scripting**
2. Enter the following code:

```javascript
System.log("INFO logging");
System.warn("WARN logging");
System.error("ERROR logging");
System.debug("DEBUG logging");

Server.log("INFO event");
Server.warn("WARN event");
Server.error("ERROR event");
```

3. Click on the X to close the window
Save and Close

1. Click on **Save and close** button (lower right)

Run the Workflow

1. Right-click on the workflow **Logging** (just created)
2. Click on **Start workflow...**
Look at the Logs

1. Select **Schema** tab
2. Select **Logs** tab
3. Make sure the dropdown has **Info** selected
4. Look at the message
Changing Log Level View

1. Click on the dropdown and select **Debug**

More log messages should appear!

**Message Details**

Each message in the log consists of the following items:

1. Time of the log execution, localized using current computer time
2. Flag indicating type of logs

   - \( I = \text{INFO} \)
   - \( W = \text{WARNING} \)
   - \( E = \text{ERROR} \)
   - \( D = \text{DEBUG} \)

3. Text decoration change depending of type of logs

   - black = INFO
   - black bold = WARNING
   - red bold = ERROR
   - black italic = DEBUG
Remember that these log messages have been written by this part of the code:

```java
System.log("INFO logging");
System.warn("WARN logging");
System.error("ERROR logging");
System.debug("DEBUG logging");
```

These logs can also be found in Server.log file in the vRO instance. By default, DEBUG logs are not logged in this file.

In a vRO cluster, logs are visible only when connected to the vRO instance where the workflow instance actually ran.

Log Insight should be used to have a consolidated view at all times.

**Look at the Events**

1. Select **Events** tab
2. Look at the message

**Events Details**

Each event consists of the following information:
1. Message (what was written in the code)
2. Type of event
3. Time of event

Events are stored in database and therefore must be used carefully.

**Close vRealize Orchestrator**

1. Click the red X in the upper right corner to close the vRealize Orchestrator Client window
2. Click **Exit** when prompted
Conclusion

In addition to providing an easy way to develop workflows, third party integrations, and lifecycle extensibility to vRealize Automation, with vRealize Orchestrator it is easy to troubleshoot workflow failures and to include failure path in the workflow itself. This make every process automated through vRealize Orchestrator much more reliable.

You've finished Module 7

Congratulations on completing Module 7.

If you are looking for additional information on Debugging workflows, try one of these:

- Go to http://bit.ly/2t53Ijm
- Or use your smart device to scan the QRC Code.

Proceed to any module below which interests you most. [Add any custom/optional information for your lab manual.]

- **Module 1 - vRealize Orchestrator Overview** (30 minutes) (Basic) Examine and understand the key features and architecture of the vRealize Orchestrator platform. Learn the methods for accessing and interacting with platform.
- **Module 2 - Creating a Basic Orchestrator Workflow** (30 minutes) (Basic) Learn how to create a basic workflow to resolve a simple yet routine task such as cloning a virtual machine or taking a snapshot.
- **Module 3 - Introduction to Parameters, Attributes and Scripting Objects** (30 minutes) (Advanced) Explore and learn about the objects and constructs that are used to extend the functionality of workflows. These include attributes, scripting objects and decision trees as well as the parameters they use.
- **Module 4 - Using Actions in vRealize Orchestrator** (30 minutes) (Advanced) Reuse your JavaScript code by turning JavaScript Functions into vRealize Orchestrator Actions, then use them repeatedly in multiple workflows.
represent individual functions that you use as building blocks in workflows and scripts.

• **Module 5 - Creating and Managing Resource Elements** (45 minutes) (Advanced) Discover how to use external objects such as scripts, XML, or HTML files in your workflows.

• **Module 6 - Creating Advanced Orchestrator Workflows** (45 minutes) (Advanced) Test your Orchestrator skills by in this module where you will create an advanced workflow. Learn about applying best practices to workflow development.

**How to End Lab**

To end your lab click on the **END** button.
Conclusion

Thank you for participating in the VMware Hands-on Labs. Be sure to visit http://hol.vmware.com/ to continue your lab experience online.

Lab SKU: HOL-1821-05-CMP

Version: 20170920-130201