Using VMware vRealize Orchestrator Plug-Ins

vRealize Orchestrator 7.5

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

Using VMware vRealize Orchestrator Plug-Ins

Using VMware vRealize Orchestrator Plug-Ins provides information and instructions about configuring and using the standard set of plug-ins installed with VMware® vRealize Orchestrator.

Intended Audience

This information is intended for advanced vSphere administrators and experienced system administrators who are familiar with virtual machine technology and datacenter operations.

Introduction to Orchestrator Plug-Ins

1

With the Orchestrator plug-ins, you can access and control external technologies and applications. Exposing an external technology in an Orchestrator plug-in lets you incorporate objects and functions in workflows and run workflows on the objects of that external technology.

The external technologies that you access by using plug-ins include virtualization management tools, email systems, databases, directory services, and remote control interfaces.

Orchestrator provides a standard set of preinstalled plug-ins, which expose the VMware vCenter Server API, email and authentication capabilities, and other technologies. In addition, the Orchestrator open plug-in architecture lets you to develop plug-ins to access other applications. Orchestrator implements open standards to simplify integration with external systems. For information about developing custom content, see *Developing with VMware vRealize Orchestrator*.

The standard set of plug-ins is automatically installed with the Orchestrator server. You might need to configure some of the plug-ins, for example the vCenter Server plug-in, before start using them.

Plug-ins extend the Orchestrator scripting engine with new object types and methods, and plug-ins publish notification events from the external system that triggers events in Orchestrator and in the plugged-in technology. Plug-ins provide an inventory of JavaScript objects that you can access on the **Inventory** tab of the Orchestrator client. Each plug-in contains packages of workflows and actions that you can run on the objects in the inventory to automate the typical use cases of the integrated product.

This chapter includes the following topics:

- Orchestrator Architecture
- Plug-Ins Installed with the Orchestrator Server
- Access the Orchestrator API Explorer

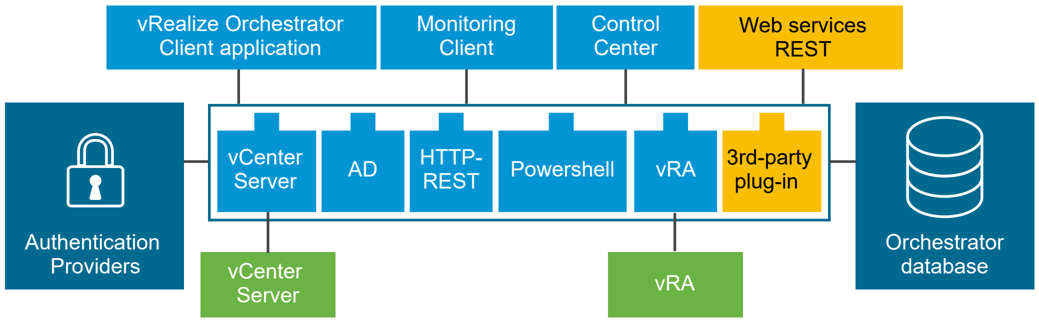
Orchestrator Architecture

Orchestrator contains a workflow library and a workflow engine to allow you to create and run workflows that automate orchestration processes. You run workflows on the objects of different technologies that Orchestrator accesses through a series of plug-ins.

Orchestrator provides a standard set of plug-ins, including a plug-in for vCenter Server and for vRealize Automation, to allow you to orchestrate tasks in the different environments that the plug-ins expose.

Orchestrator also presents an open architecture for plugging in external third-party applications to the orchestration platform. You can run workflows on the objects of the plugged-in technologies that you define yourself. Orchestrator connects to an authentication provider to manage user accounts and to a database to store information from the workflows that it runs. You can access Orchestrator, the objects it exposes, and the Orchestrator workflows through the Orchestrator client interface, or through Web services. Monitoring and configuration of Orchestrator workflows and services is done through the Monitoring Client and Control Center.

Figure 1-1. VMware vRealize Orchestrator Architecture



Plug-Ins Installed with the Orchestrator Server

Orchestrator includes a collection of standard plug-ins. Each plug-in exposes an external product API to the Orchestrator platform. Plug-ins provide inventory classes, additional object types for the scripting engine, and publish notification events from the external system. Each plug-in also provides a library of workflows for automating the typical use cases of the integrated external products.

You can see the list of the installed plug-ins on the **Manage Plug-ins** page in Control Center. For the plug-ins that require configuration, there are separate tabs in the interface.

Table 1-1. Plug-ins Installed With Orchestrator

Plug-In	Purpose	Configuration
vCenter Server	Provides access to the vCenter Server API so that you can incorporate all the vCenter Server objects and functions into the management processes that you automate by using Orchestrator.	See Configuring the vCenter Server Plug-In.
Configuration	Provides workflows for configuring the Orchestrator authentication, database connection, SSL certificates, and so on.	None
Library	Provides workflows that act as basic building blocks for customization and automation of client processes. The workflow library includes templates for life cycle management, provisioning, disaster recovery, hot backup, and other standard system management processes. You can copy and edit the templates to modify them according to your needs.	None
SQL	Provides the Java Database Connectivity (JDBC) API, which is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases. The databases include SQL databases and other tabular data sources, such as spreadsheets or flat files. The JDBC API provides a call-level API for SQL-based database access from workflows.	None
SSH	Provides an implementation of the Secure Shell v2 (SSH-2) protocol. Allows remote command and file transfer sessions with password and public key-based authentication in workflows. Supports keyboard-interactive authentication. Optionally, the SSH plug-in can provide remote file system browsing directly in the Orchestrator client inventory.	See Configure the SSH Plug-In.
XML	A complete Document Object Model (DOM) XML parser that you can implement in workflows. Alternatively, you can use the ECMAScript for XML (E4X) implementation in the Orchestrator JavaScript API.	None
Mail	Uses Simple Mail Transfer Protocol (SMTP) to send email from workflows.	Set the default values for the `EmailMessage` object to use. See Define the Default SMTP Connection.
Net	Uses the Jakarta Apache Commons Net Library. Provides implementations of the Telnet, FTP, POP3, and IMAP protocols. The POP3 and IMAP protocols is used for reading email. In combination with the Mail plug-in, the Net plug-in provides complete email sending and receiving capabilities in workflows.	None
Workflow documentation	Provides workflows that you can use to generate information in PDF format about a workflow or a workflow category.	None

Table 1-1. Plug-ins Installed With Orchestrator (continued)

Plug-In	Purpose	Configuration
Enumeration	Provides common Enumerated Types that can be used in workflows by other plug-ins.	None
HTTP-REST	Enables management of REST Web services through an interaction between vRealize Orchestrator and REST hosts.	See Configuring the HTTP-REST Plug-In.
SOAP	Lets you manage SOAP Web services by providing interaction between vRealize Orchestrator and SOAP hosts.	See Configuring the SOAP Plug-In.
AMQP	Lets you interact with Advanced Message Queuing Protocol (AMQP) servers also known as brokers.	See Configuring the AMQP Plug-In.
SNMP	Enables vRealize Orchestrator to connect and receive information from SNMP-enabled systems and devices.	None
Active Directory	Provides interaction between vRealize Orchestrator and Microsoft Active Directory.	See Configuring the Active Directory Plug-In.
Dynamic Types	Lets you define dynamic types and create and use objects of these dynamic types.	See Chapter 19 Using the Dynamic Types Plug-In.
Multi-Node	Contains workflows for hierarchical management, management of Orchestrator instances, and scale-out of Orchestrator activities.	See Chapter 21 Using the Multi-Node Plug-In.
PowerShell	Lets you manage PowerShell hosts and run custom PowerShell operations.	See Chapter 20 Using the PowerShell Plug-In.

Plug-In Components

Each plug-in is a DAR file package. The DAR files are stored in `/var/lib/vco/app-server/plugins` on the Orchestrator Appliance. The components of each plug-in, such as workflow categories and API modules, use different naming conventions.

Table 1-2. Names of Plug-In Components

Plug-In Name in the Configuration UI	DAR File	Workflow Categories	API Module
vCenter Server	`o11nplugin-vsphere.dar`	vCenter	`VC`
vRO Configuration	`o11nplugin-configurator.dar`	Configuration	`Configurator`
Library	`o11nplugin-library.dar`	Locking Orchestrator Troubleshooting	Not applicable.
SQL	`o11nplugin-database.dar`	JDBC SQL	`SQL`
SSH	`o11nplugin-ssh.dar`	SSH	`SSH`
XML	`o11nplugin-xml.dar`	XML	`XML`
Mail	`o11nplugin-mail.dar`	Mail	`Mail`

Table 1-2. Names of Plug-In Components (continued)

Plug-In Name in the Configuration UI	DAR File	Workflow Categories	API Module
Net	`o11nplugin-jakartacommonsnet.dar`	None	`Net`
Workflow documentation	`o11nplugin-wfdocs.dar`	Workflow documentation	Workflow documentation
Common enumerated types	`o11nplugin-enums.dar`	None	`Enums`
Dynamic Types	`o11n-plugin-dynamictypes.dar`	Configuration	`DynamicTypes`
HTTP-REST	`o11nplugin-rest.dar`	Configuration	`REST`
SOAP	`o11n-plugin-soap.dar`	Configuration	`SOAP`
AMQP	`o11n-plugin-amqp.dar`	Configuration	`AMQP`
SNMP	`o11n-plugin-snmp.dar`	Device Management Query Management Trap Host Management	`SNMP`
Active Directory	`o11nplugin-ad.dar`	Computer Configuration Organizational Unit User User Group	`AD`
Orchestrator	`o11nplugin-multi-node.dar`	Servers Configuration Remote Execution Remote Management Tasks Workflows	`VCO`
PowerShell	`o11nplugin-powershell.dar`	Configuration Generate Templates	`PowerShell`

Access the Orchestrator API Explorer

Orchestrator provides an API Explorer that you can use to search the Orchestrator API and see the documentation for JavaScript objects that you can use in scripted elements.

You can consult an online version of the Scripting API for the vCenter Server plug-in on the Orchestrator documentation home page.

Procedure

1 Log in to the Orchestrator client.

2 Select **Tools > API Explorer**.

Results

The API Explorer appears. You can use it to search all the objects and functions of the Orchestrator API.

What to do next

Use the API Explorer to write scripts for scriptable elements.

Configure the Orchestrator Plug-Ins

2

The default Orchestrator plug-ins are configured only through workflows.

If you want to configure any of the default Orchestrator plug-ins, you need to use the specific workflow from the Orchestrator client.

This chapter includes the following topics:

- Manage the Orchestrator Plug-Ins
- Uninstall a Plug-In

Manage the Orchestrator Plug-Ins

In the **Manage Plug-Ins** page of Control Center, you can view a list of all plug-ins that are installed in Orchestrator and perform basic management actions.

Change Plug-Ins Logging Level

Instead of changing the logging level for Orchestrator, you can change it only for specific plug-ins.

Install a New Plug-In

With the Orchestrator plug-ins, the Orchestrator server can integrate with other software products. The Orchestrator Appliance includes a set of preinstalled plug-ins and you can also install custom plug-ins.

All Orchestrator plug-ins are installed from Control Center. The file extensions that can be used are `.vmoapp` and `.dar`. A `.vmoapp` file can contain a collection of several `.dar` files and can be installed as an application, while a `.dar` file contains all the resources associated with one plug-in.

Disable a Plug-In

You can disable a plug-in by deselecting the **Enable** check box next to the name of the plug-in.

This action does not remove the plug-in file. For more information on uninstalling a plug-in in Orchestrator, see Uninstall a Plug-In.

Uninstall a Plug-In

You can use Control Center to disable a plug-in, but this action does not remove the plug-in file from the Orchestrator Appliance file system. To remove the plug-in file, you must log in to the Orchestrator Appliance and remove the plug-in file manually.

Procedure

1 Delete the plug-in from the Orchestrator Appliance.

   a Log in to the Orchestrator Appliance over SSH as **`root`**.

   b Open the `/etc/vco/app-server/plugins/_VSOPluginInstallationVersion.xml` file with a text editor.

   c Delete the line of code that corresponds to the plug-in that you want to remove.

   d Navigate to the `/var/lib/vco/app-server/plugins` directory.

   e Delete the `.dar` archives that contain the plug-in that you want to remove.

2 Restart the vRealize Orchestrator services.

```
service vco-configurator restart && service vco-server restart
```

3 Log in to Control Center as **`root`**.

4 In the **Manage Plug-Ins** page, verify that the plug-in is removed.

5 Through the Orchestrator client, delete the packages and folders that are related to the plug-in.

   a Log in to the Orchestrator client.

   b Select **Design** from the drop-down menu in the upper-left corner.

   c Click the **Packages** view.

   d Right-click the package that you want to delete, and select **Delete element with content**.

   Note Orchestrator elements that are locked in the read-only state, for example, workflows in the standard library, are not deleted.

   e From the **Tools** menu in the upper-right corner, select **User preferences**.

   The **Preferences** context menu opens.

   f On the **General** page, select the **Delete non empty folder permitted** check box.

   You can now delete an entire folder, including its subfolders and workflows, with a single click.

   g Click the **Workflow** view.

   h Delete the folder of the plug-in that you want to remove.

  i Click the **Actions** view.
  j Delete the action modules of the plug-in that you want to remove.

6 Restart the vRealize Orchestrator services.

Results

You removed all custom workflows, actions, policies, configurations, settings, and resources related to the plug-in.

Using the vCenter Server Plug-In

3

You can use the vCenter Server plug-in to manage multiple vCenter Server instances. You can create workflows that use the vCenter Server plug-in API to automate tasks in your vCenter Server environment.

The vCenter Server plug-in maps the vCenter Server API to the JavaScript that you can use in workflows. The plug-in also provides actions that perform individual vCenter Server tasks that you can include in workflows.

The vCenter Server plug-in provides a library of standard workflows that automate vCenter Server operations. For example, you can run workflows that create, clone, migrate, or delete virtual machines.

The vCenter Server plug-in includes the Policy-Based Management (PBM) and the Storage Montoring Service (SMS) APIs as scripting objects in the Orchestrator scripting API. The Storage Policy-Based Management policies and components appear in the Orchestrator **Inventory** tab.

This chapter includes the following topics:

Configuring the vCenter Server Plug-In

Before managing the objects in your vSphere inventory by using Orchestrator and to run workflows on the objects, you must configure the vCenter Server plug-in and define the connection parameters between Orchestrator and the vCenter Server instances you want to orchestrate.

You can configure the vCenter Server plug-in by running the vCenter Server configuration workflows from the Orchestrator client.

To manage the objects in your vSphere inventory by using the vSphere Web Client, make sure that you configure the Orchestrator server to work with the same vCenter Single Sign-On instance to which both vCenter Server and vSphere Web Client are pointing. You must also ensure that Orchestrator is registered as a vCenter Server extension. You register Orchestrator as a vCenter Server extension when you specify a user (by providing the user name and password), who has the privileges to manage vCenter Server extensions.

Configuration Workflows

The Configuration workflow category of the vCenter Server plug-in contains workflows that let you manage the connections to vCenter Server instances.

You can access these workflows from **Library > vCenter > Configuration** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add a vCenter Server instance	Configures Orchestrator to connect to a new vCenter Server instance so that you can run workflows over the objects in the vSphere infrastructure.
List the vRealize Orchestrator extensions of vCenter Server	Lists all vRealize Orchestrator extensions of vCenter Server.
Register Orchestrator as a vCenter Server extension	Registers the Orchestrator instance as a vCenter Server extension.
Remove a vCenter Server instance	Removes a vCenter Server instance from the Orchestrator inventory. You will no longer be able to orchestrate this vCenter Server instance.
Update a vCenter Server instance	Updates the connection to a vCenter Server instance. For example, if the IP address of your vCenter Server system changes, you must update the connection parameters to the vCenter Server instance so that you can manage your vSphere inventory with Orchestrator.
Unregister a vCenter Server extension	Unregisters a vSphere Web Client extension.

Configure the Connection to a vCenter Server Instance

You can configure the connections to vCenter Server instances by running the vCenter Server configuration workflows in the Orchestrator client.

Procedure

1 Log in to the Orchestrator client as an administrator.

2 Click the **Workflows** view in the Orchestrator client.

3 In the workflows hierarchical list, expand **Library > vCenter > Configuration** and navigate to the **Add a vCenter Server instance** workflow.

4 Right-click the **Add a vCenter Server instance** workflow and select **Start workflow**.

5 Enter the IP address or the DNS name of the machine on which the vCenter Server instance you want to add is installed.

Note The hostname that you enter is case-sensitive.

6 Retain the default port value, **443**.

7 Retain the default location of the SDK to use to connect to your vCenter Server instance.

8 Select whether you want to manage the vCenter Server instance through Orchestrator, and click **Next**.

9 Select whether you want to ignore certificate warnings for the vCenter Server instances that you want to add.

If you choose to ignore certificate warnings, the vCenter Server instance certificate is accepted silently and the certificate is added to the trusted store.

10 Select the method that you want to use to manage user access on the vCenter Server system.

Option	Description
Share a unique session	Allows Orchestrator to create only one connection to vCenter Server. In the **User name** and **Password** text boxes, enter the credentials for Orchestrator to use to establish the connection to the vCenter Server host. The user that you select must be a valid user with privileges to manage vCenter Server extensions and a set of custom defined privileges. Orchestrator uses these credentials to monitor the VirtualCenter Web service, typically to run Orchestrator system workflows.
Session per user	Creates a new session to vCenter Server. This action might rapidly use CPU, memory, and bandwidth. Select this option only if your vCenter Server is in an Active Directory domain or if vCenter Server Single Sign-On is enabled. The user that you select must be a valid user with privileges to manage vCenter Server extensions.

The user account that you select is also used by the policy engine to collect statistical and other data. If the user that you select does not have enough privileges, the policy engine cannot access the necessary parts of the vCenter Server inventory and cannot collect the necessary data.

11 (Optional) Enter the user domain.

You must specify the user domain name only when you select to use a shared session.

Note Fill this text box if session per user is selected.

12 (Optional) Enter the URLs for the vSphere storage management endpoints.

You can configure the Policy-Based Management (PBM) endpoint, the Storage Monitoring Service (SMS) endpoint, or both.

13 Click **Submit** to run the workflow.

Results

After the workflow runs successfully, the vCenter Server instance and all vSphere objects that belong to it appear in the **Inventory** view.

vCenter Server Plug-In Scripting API

The vCenter Server scripting API contains classes, with their respective attributes, methods, and constructors that allow interaction between vRealize Orchestrator and vCenter Server. You can use the API to develop custom workflows.

For a list of available API objects, see https://www.vmware.com/support/orchestrator/doc/vro-vsphere65-api/index.html.

Using the vCenter Server Plug-In Inventory

The vCenter Server plug-in exposes all objects of the connected vCenter Server instances in the Inventory view. You can use the **Inventory** tab to add authorization elements or to run workflows on vCenter Server objects.

If you enable the **Use contextual menu in inventory** option from the **Inventory** tab of the User preferences tool, all of the workflows that you can run on the selected inventory object appear in a pop-up menu.

Performance Considerations for Querying

With the vCenter Server plug-in for vRealize Orchestrator, you can query the vCenter Server inventory for specific objects.

Querying Methods

For querying, you can either use the `vCSearchIndex` managed object, or the object finder methods that are included in the plug-in inventory, such as `getAllDatastores()`, `getAllVirtualMachines()`, `findAllForType()`, and others.

Performance

By default, both methods return the queried objects without including any of their properties, unless you specify a set of properties as an argument for the method parameters in the search query.

Note You must always use query expressions with the `getAll...()` and `findAll...()` finder objects to prevent the Orchestrator client from filtering large sets of returned objects, which might affect the overall performance of the Orchestrator server.

You can use two types of expressions for querying the vCenter Server inventory.

Type of Expression	Description
Name expressions	You can specify a name as an argument for a query parameter. **Note** The objects are filtered by the specified name argument according to the name of the plug-in object as it is appears in the vCenter Server plug-in inventory.
XPath expressions	You can use expressions based on the XPath query language. For more information, see Using XPath Expressions with the vCenter Server Plug-In.

When you invoke a vCenter Server inventory object with custom properties, each reference to this object, in a workflow or an action, sends a query to the vCenter Server, which generates a noticeable performance overhead. To optimize performance and avoid serializing and deserializing the object multiple times within a workflow run, it is best to use a shared resource to store the object, instead of storing it as a workflow attribute, an input, or an output parameter. Such shared resource can be a configuration element or a resource element.

Using XPath Expressions with the vCenter Server Plug-In

You can use the finder methods in the vCenter Server plug-in to query for vCenter Server inventory objects. You can use XPath expressions to define search parameters.

The vCenter Server plug-in includes a set of object finder methods such as `getAllDatastores()`, `getAllResourcePools()`, `findAllForType()`. You can use these methods to access the inventories of the vCenter Server instances that are connected to your Orchestrator server and search for objects by ID, name, or other properties.

For performance reasons, the finder methods do not return any properties for the queried objects, unless you specify a set of properties in the search query.

You can consult an online version of the Scripting API for the vCenter Server plug-in on the Orchestrator documentation home page.

Important The queries based on XPath expressions might impact the Orchestrator performance because the finder method returns all objects of a given type on the vCenter Server side and the query filters are applied on the vCenter Server plug-in side.

Using XPath Expressions with the vCenter Server Plug-In

When you invoke a finder method, you can use expressions based on the XPath query language. The search returns all the inventory objects that match the XPath expressions. If you want to query for any properties, you can include them to the search script in the form of a string array.

The following JavaScript example uses the `VcPlugin` scripting object and an XPath expression to return the names of all datastore objects that are part of the vCenter Server managed objects and contain the string **`ds`** in their names.

```
var datastores = VcPlugin.getAllDatastores(null, "xpath:name[contains(.,'ds')]");
for each (datastore in datastores){
     System.log(datastore.name);
 }
```

The same XPath expression can be invoked by using the `Server` scripting object and the `findAllForType` finder method.

```
var datastores = Server.findAllForType("VC:Datastore", "xpath:name[contains(.,'ds')]");
for each (datastore in datastores){
     System.log(datastore.name);
 }
```

The following script example returns the names of all host system objects whose ID starts with the digit **`1`**.

```
var hosts = VcPlugin.getAllHostSystems(null, "xpath:id[starts-with(.,'1')]");
for each (host in hosts){
     System.log(host.name);
}
```

The following script returns the names and IDs of all data center objects that contain the string **`DC`**, in upper- or lower-case letters, in their names. The script also retrieves the **`tag`** property.

```
var datacenters = VcPlugin.getAllDatacenters(['tag'], "xpath:name[contains(translate(., 'DC', 'dc'),
'dc')]");
for each (datacenter in datacenters){
     System.log(datacenter.name + “ ”  +  datacenter.id);
}
```

Access the vCenter Server Plug-In Workflow Library

You must use the Orchestrator client or the vSphere Web Client to access the elements from the vCenter Server plug-in workflow library.

Prerequisites

- Configure a connection to a vCenter Server instance.
- Verify that the user account you are logged in with has the necessary permissions to run vCenter Server workflows.

Procedure

1 In the Orchestrator client, select **Design** or **Run** from the drop-down menu in the left upper corner.

2 Click the **Workflows** view in the Orchestrator client left pane.

3 Expand the hierarchical list to **Library > vCenter**.

What to do next

Review the workflow library.

vCenter Server Plug-In Workflow Library

The vCenter Server plug-in workflow library contains workflows that you can use to run automated processes related to the management of vCenter Server.

- Batch Workflows

  Batch workflows populate configuration elements or run workflows on a selected vCenter Server object.

- Cluster and Compute Resource Workflows

  With the cluster and compute resource workflows, you can create, rename, or delete a cluster. You can also enable or disable high availability, Distributed Resource Scheduler, and vCloud Distributed Storage on a cluster.

- Configuration Workflows

  The Configuration workflow category of the vCenter Server plug-in contains workflows that let you manage the connections to vCenter Server instances.

- Custom Attributes Workflows

  With custom attributes workflows, you can add custom attributes to virtual machines or get a custom attribute for a virtual machine.

- Datacenter Workflows

  With datacenter workflows, you can create, delete, reload, rename, or rescan a datacenter.

- Datastore and Files Workflows

  With the datastore and files workflows, you can delete a list of files, find unused files in a datastore, and so on.

- Datacenter Folder Management Workflows

  With datacenter folder management workflows, you can create, delete, or rename a datacenter folder.

- Host Folder Management Workflows

  With host folder management workflows, you can create, delete, or rename a host folder.

- Virtual Machine Folder Management Workflows

  With virtual machine folder management workflows, you can create, delete, or rename a virtual machine folder.

- Guest Operation Files Workflows

  With the guest operation files workflows, you can manage files in a guest operating system.

- Guest Operation Processes Workflows

  With guest operation processes workflows, you can get information and control the running processes in a guest operating system.

- Power Host Management Workflows

  With power host management workflows you can reboot or shut down a host.

- Basic Host Management Workflows

  With the basic host management workflows, you can put a host into maintenance mode and make a host exit maintenance mode. You can also move a host to a folder or a cluster, and reload data from a host.

- Host Registration Management Workflows

  With the host registration management workflows, you can add a host to a cluster, disconnect, or reconnect a host from a cluster, and so on.

- Networking Workflows

  With networking workflows you can add a port group to distributed virtual switch, create a distributed virtual switch with a port group, and so on.

- Distributed Virtual Port Group Workflows

  With the distributed virtual port group workflows, you can update or delete a port group, and reconfigure the port group.

- Distributed Virtual Switch Workflows

  With distributed virtual switch workflows, you can create, update or delete a distributed virtual switch, and create, delete, or update a private VLAN.

- Standard Virtual Switch Workflows

  With the standard virtual switch workflows you can create, update, or delete a standard virtual switch, and create, delete, or update port groups in standard virtual switches.

- Networking Virtual SAN Workflows

  With Virtual SAN workflows, you can configure Virtual SAN network traffic.

- Resource Pool Workflows

  With the resource pool workflows you can create, rename, reconfigure or delete a resource pool, and get resource pool information.

- Storage Workflows

  With the storage workflows, you can perform storage-related operations.

- Storage DRS Workflows

  With the storage DRS workflows, you perform storage-related operations, such as creating and configuring a datastore cluster, removing a datastore from cluster, adding storage to a cluster, and others.

- Storage VSAN Workflows

  With the Virtual SAN workflows, you can manage non-SSD disks and disk groups in a Virtual SAN cluster.

- Basic Virtual Machine Management Workflows

  With the basic virtual machine management workflows, you can perform basic operations on virtual machines, for example, create, rename or delete a virtual machine, upgrade virtual hardware, and others.

- Clone Workflows

  With clone workflows, you can clone virtual machines with or without customizing the virtual machine properties.

- Linked Clone Workflows

  With the linked clone workflows, you can perform linked clone operations such as restoring a virtual machine from a linked clone, creating a linked clone, or others.

- Linux Customization Clone Workflows

  With Linux customization workflows, you can clone a Linux virtual machine and customize the guest operating system.

- Tools Clone Workflows

  With the tools clone workflows, you can obtain customization information about the operating system of the virtual machine, information required to update a virtual device, and others.

- Windows Customization Clone Workflows

  With the Windows customization clone workflows, you can clone Windows virtual machines and customize the guest operating system.

- Device Management Workflows

  With the device management workflows, you can manage the devices that are connected to a virtual machine or to a host datastore.

- Move and Migrate Workflows

  With the move and migrate workflows, you can migrate virtual machines.

- Other Workflows

  With the workflows from the Others category, you can enable and disable Fault Tolerance (FT), extract virtual machine information, and find orphaned virtual machines.

- Power Management Workflows

  With the power management workflows, you can power on and off virtual machines, reboot the guest operating system of a virtual machine, suspend a virtual machine, and others.

- Snapshot Workflows

  With snapshot workflows, you can perform snapshot-related operations.

- VMware Tools Workflows

  With VMware Tools workflows, you can perform VMware Tools-related tasks on virtual machines.

Batch Workflows

Batch workflows populate configuration elements or run workflows on a selected vCenter Server object.

You can access the batch workflows from **Library > vCenter > Batch** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Fill batch configuration elements	Populates the configuration elements that the Run a workflow on a selection of objects workflow uses. Performs the following tasks: ■ Resets the `BatchObject` and `BatchAction` configuration elements. ■ Fills the `BatchObject` configuration element with all the workflows that have only one input parameter. ■ Fills the `BatchAction` configuration element with all the actions that have no input parameters or one input parameter and that have an array as the `returnType`.
Run a workflow on a selection of objects	Runs a workflow on a selection of vCenter Server objects, taking one action as input. This is the action that retrieves the list of objects on which to run the workflow. To return the objects without running the selected workflow, run the workflow in simulation mode.

Cluster and Compute Resource Workflows

With the cluster and compute resource workflows, you can create, rename, or delete a cluster. You can also enable or disable high availability, Distributed Resource Scheduler, and vCloud Distributed Storage on a cluster.

You can access the cluster and compute resource workflows from **Library > vCenter > Cluster and Compute Resource** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add DRS virtual machine group to cluster	Adds a DRS virtual machine group to a cluster.
Add virtual machines to DRS group	Adds a virtual machine list to an existing DRS virtual machine group.
Create cluster	Creates a cluster in a host folder.
Delete cluster	Deletes a cluster.
Disable DRS on cluster	Disables DRS on a cluster.
Disable HA on cluster	Disables high availability on a cluster.
Disable vCloud Distributed Storage on cluster	Disables vCloud Distributed Storage on a cluster.
Enable DRS on cluster	Enables DRS on a cluster.
Enable HA on cluster	Enables high availability on a cluster.
Enable vCloud Distributed Storage on cluster	Enables vCloud Distributed Storage on a cluster.
Remove virtual machine DRS group from cluster	Removes a DRS virtual machine group from a cluster.

Workflow Name	Description
Remove virtual machines from DRS group	Removes virtual machines from a cluster DRS group.
Rename cluster	Renames a cluster.

Configuration Workflows

The Configuration workflow category of the vCenter Server plug-in contains workflows that let you manage the connections to vCenter Server instances.

You can access these workflows from **Library > vCenter > Configuration** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add a vCenter Server instance	Configures Orchestrator to connect to a new vCenter Server instance so that you can run workflows over the objects in the vSphere infrastructure.
List the Orchestrator extensions of vCenter Server	Lists all Orchestrator extensions of vCenter Server.
Register Orchestrator as a vCenter Server extension	Registers the Orchestrator instance as a vCenter Server extension.
Remove a vCenter Server instance	Removes a vCenter Server instance from the Orchestrator inventory. You cannot orchestrate this vCenter Server instance any longer.
Update a vCenter Server instance	Updates the connection to a vCenter Server instance. For example, if the IP address of your vCenter Server system changes, you must update the connection parameters to the vCenter Server instance so that you can manage your vSphere inventory with Orchestrator.
Unregister a vCenter Server extension	Unregisters a vCenter Server extension.

Custom Attributes Workflows

With custom attributes workflows, you can add custom attributes to virtual machines or get a custom attribute for a virtual machine.

You can access the custom attributes workflows from **Library > vCenter > Custom Attributes** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add custom attribute to a virtual machine	Adds a custom attribute to a virtual machine.
Add custom attribute to multiple virtual machines	Adds a custom attribute to a selection of virtual machines.
Get custom attribute	Gets a custom attribute for a virtual machine in vCenter Server.

Datacenter Workflows

With datacenter workflows, you can create, delete, reload, rename, or rescan a datacenter.

You can access the datacenter workflows from **Library > vCenter > Datacenter** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Create datacenter	Creates a data center in a data center folder.
Delete datacenter	Deletes a data center.
Reload datacenter	Forces vCenter Server to reload data from a data center.
Rename datacenter	Renames a data center and waits for the task to complete.
Rescan datacenter HBAs	Scans the hosts in a data center and initiates a rescan on the host bus adapters to discover new storage.

Datastore and Files Workflows

With the datastore and files workflows, you can delete a list of files, find unused files in a datastore, and so on.

You can access the datastore and files workflows from **Library > vCenter > Datastore and Files** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Delete all files	Deletes a list of files.
Delete all unused datastore files	Searches all datastores in the vCenter Server environment and deletes all unused files.
Export unused datastore files	Searches all datastores and creates an XML descriptor file that lists all unused files.
Find unused files in datastores	Searches the vCenter Server environment for all unused disks (`*.vmdk`), virtual machines (`*.vmx`), and template (`*.vmtx`) files that are not associated with any vCenter Serverinstances registered with Orchestrator.
Get all configuration, template, and disk files from virtual machines	Creates a list of all virtual machine descriptor files and a list of all virtual machine disk files, for all datastores.
Log all datastore files	Creates a log for every virtual machine configuration file and every virtual machine file found in all datastores.
Log unused datastore files	Searches the vCenter Server environment for unused files that are registered on virtual machines and exports a log of the files in a text file.
Upload file to datastore	Uploads a file to an existing folder on a specific datastore. The uploaded file overwrites any existing file with the same name in the same destination folder.

Datacenter Folder Management Workflows

With datacenter folder management workflows, you can create, delete, or rename a datacenter folder.

You can access the datacenter folder management workflows from **Library > vCenter > Folder management > Datacenter folder** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Create datacenter folder	Creates a data center folder.
Delete datacenter folder	Deletes a data center folder and waits for the task to complete.
Rename datacenter folder	Renames a data center folder and waits for the task to complete.

Host Folder Management Workflows

With host folder management workflows, you can create, delete, or rename a host folder.

You can access the host folder management workflows from **Library > vCenter > Folder management > Host folder** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Create host folder	Creates a host folder.
Delete host folder	Deletes a host folder and waits for the task to complete.
Rename host folder	Renames a host folder and waits for the task to complete.

Virtual Machine Folder Management Workflows

With virtual machine folder management workflows, you can create, delete, or rename a virtual machine folder.

You can access the virtual machine folder management workflows from **Library > vCenter > Folder management > VM folder** in the **Workflow** view of the Orchestrator client.

Workflow Name	Description
Create virtual machine folder	Creates a virtual machine folder.
Delete virtual machine folder	Deletes a virtual machine folder and waits for the task to complete.
Rename virtual machine folder	Renames a virtual machine folder and waits for the task to complete.

Guest Operation Files Workflows

With the guest operation files workflows, you can manage files in a guest operating system.

You can access the guest operation files workflows from **Library > vCenter > Guest operations > Files** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Check for directory in guest	Verifies that a directory exists in a guest virtual machine.
Check for file in guest	Verifies that a file exists in a guest virtual machine.
Copy file from guest to Orchestrator	Copies a specified file from a guest file system to an Orchestrator server.
Copy file from Orchestrator to guest	Copies a specified file from an Orchestrator server to a guest file system.
Create directory in guest	Creates a directory in a guest virtual machine.
Create temporary directory in guest	Creates a temporary directory in a guest virtual machine.

Workflow Name	Description
Create temporary file in guest	Creates a temporary file in a guest virtual machine.
Delete directory in guest	Deletes a directory from a guest virtual machine.
Delete file in guest	Deletes a file from a guest virtual machine.
List path in guest	Shows a path in a guest virtual machine.
Move directory in guest	Moves a directory in a guest virtual machine.
Move file in guest	Moves a file in a guest virtual machine.

Guest Operation Processes Workflows

With guest operation processes workflows, you can get information and control the running processes in a guest operating system.

You can access the guest operation files workflows from **Library > vCenter > Guest operations > Processes** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Get environment variables from guest	Returns a list with environmental variables from a guest. An interactive session returns the variables of the user who is currently logged in.
Get processes from guest	Returns a list with the processes running in the guest operating system and the recently completed processes started by the API.
Run program in guest	Starts a program in a guest operating system.
Kill process in guest	Terminates a process in a guest operating system.

Power Host Management Workflows

With power host management workflows you can reboot or shut down a host.

You can access the power host management workflows from **Library > vCenter > Host management > Power** in the **Workflows** view of the Orchestrator client.

Reboot host

Reboots a host. If the Orchestrator client is connected directly to the host, it loses the connection to the host and does not receive an indication of success in the returned task.

Shut down host

Shuts down a host. If the Orchestrator client is connected directly to the host, it loses the connection to the host and does not receive an indication of success in the returned task.

Basic Host Management Workflows

With the basic host management workflows, you can put a host into maintenance mode and make a host exit maintenance mode. You can also move a host to a folder or a cluster, and reload data from a host.

You can access the basic host management workflows from **Library > vCenter > Host management > Basic** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Enter maintenance mode	Puts the host into maintenance mode. You can cancel the task.
Exit maintenance mode	Exits maintenance mode. You can cancel the task.
Move host to cluster	Moves an existing host to a cluster. The host must be part of the same data center, and if the host is part of a cluster, the host must be in maintenance mode.
Move host to folder	Moves a host into a folder as a standalone host. The host must be part of a `ClusterComputeResource` in the same data center and the host must be in maintenance mode.
Reload host	Forces vCenter Server to reload data from a host.

Host Registration Management Workflows

With the host registration management workflows, you can add a host to a cluster, disconnect, or reconnect a host from a cluster, and so on.

You can access the host management registration workflows from **Library > vCenter > Host management > Registration** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add host to cluster	Adds a host to the cluster. This workflow fails if it cannot authenticate the SSL certificate of the host.
Add standalone host	Registers a host as a standalone host.
Disconnect host	Disconnects a host from the vCenter Server instance.
Reconnect host	Reconnects a disconnected host by providing only the host information.
Reconnect host with all information	Reconnects a disconnected host by providing all information about the host.
Remove host	Removes a host and unregisters it from the vCenter Server instance. If the host is part of a cluster, you must put it in maintenance mode before attempting to remove it.

Networking Workflows

With networking workflows you can add a port group to distributed virtual switch, create a distributed virtual switch with a port group, and so on.

You can access the networking workflows from **Library > vCenter > Networking** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add port group to distributed virtual switch	Adds a new distributed virtual port group to a specified distributed virtual switch.
Attach host system to distributed virtual switch	Adds a host to a distributed virtual switch.
Create distributed virtual switch with port group	Creates a new distributed virtual switch with a distributed virtual port group.

Distributed Virtual Port Group Workflows

With the distributed virtual port group workflows, you can update or delete a port group, and reconfigure the port group.

You can access the distributed virtual port group workflows from **Library > vCenter > Networking > Distributed virtual port group** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Connect virtual machine NIC number to distributed virtual port group	Reconfigures the network connection of the specified virtual machine NIC number to connect to the specified distributed virtual port group. If no NIC number is specified, the number zero is used.
Delete distributed virtual port group	Deletes a specified distributed virtual port group.
Set teaming options	Provides an interface to manage the teaming options for a distributed virtual port group.
Update distributed virtual port group	Updates the configuration of a specified distributed virtual port group.

Distributed Virtual Switch Workflows

With distributed virtual switch workflows, you can create, update or delete a distributed virtual switch, and create, delete, or update a private VLAN.

You can access the distributed virtual switch workflows from **Library > vCenter > Networking > Distributed virtual switch** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Create distributed virtual switch	Creates a distributed virtual switch in the specified network folder with a name and uplink port names that you specify. You must specify at least one uplink port name.
Create private VLAN	Creates a VLAN on the specified distributed virtual switch.
Delete distributed virtual switch	Deletes a distributed virtual switch and all associated elements.
Delete private VLAN	Deletes a VLAN from a specified distributed virtual switch. If a secondary VLAN exists, you must first delete the secondary VLAN.
Update distributed virtual switch	Updates the properties of a distributed virtual switch.
Update private VLAN	Updates a VLAN on the specified distributed virtual switch.

Standard Virtual Switch Workflows

With the standard virtual switch workflows you can create, update, or delete a standard virtual switch, and create, delete, or update port groups in standard virtual switches.

You can access the standard virtual switch workflows from **Library > vCenter > Networking > Standard virtual switch** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add port group in standard virtual switch	Adds a port group in a standard virtual switch.
Create standard virtual switch	Creates a standard virtual switch.

Workflow Name	Description
Delete port group from standard virtual switch	Deletes a port group from a standard virtual switch.
Delete standard virtual switch	Deletes a standard virtual switch from a host network configuration.
Retrieve all standard virtual switches	Retrieves all standard virtual switches from a host.
Update port group in standard virtual switch	Updates the properties of a port group in a standard virtual switch.
Update standard virtual switch	Updates the properties of a standard virtual switch.
Update VNIC for port group in standard virtual switch	Updates a virtual NIC associated with a port group in a standard virtual switch.

Networking Virtual SAN Workflows

With Virtual SAN workflows, you can configure Virtual SAN network traffic.

You can access the networking workflows from **Library > vCenter > Networking > VSAN** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Set a cluster's VSAN traffic network	Sets a Virtual SAN traffic network of the cluster.
Set a host's VSAN traffic network	Sets a Virtual SAN traffic network of the host.

Resource Pool Workflows

With the resource pool workflows you can create, rename, reconfigure or delete a resource pool, and get resource pool information.

You can access the resource pool workflows from **Library > vCenter > Resource Pool** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Create resource pool	Creates a resource pool with the default CPU and memory allocation values. To create a resource pool in a cluster, the cluster must have VMware DRS enabled.
Create resource pool with specified values	Creates a resource pool with CPU and memory allocation values that you specify. To create a resource pool in a cluster, the cluster must have VMware DRS enabled.
Delete resource pool	Deletes a resource pool and waits for the task to complete.
Get resource pool information	Returns CPU and memory information about a given resource pool.
Reconfigure resource pool	Reconfigures CPU and memory allocation configuration for a given resource pool.
Rename resource pool	Renames a resource pool and waits for the task to complete

Storage Workflows

With the storage workflows, you can perform storage-related operations.

You can access the storage workflows from **Library > vCenter > Storage** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add datastore on iSCSI/FC/local SCSI	Creates a datastore on a Fibre Channel, iSCSI or local SCSI disk. Only disks that are not currently in use by an existing VMFS are applicable to new datastore creation. The new datastore allocates the maximum available space of the specified disk.
Add datastore on NFS	Adds a datastore on an NFS server.
Add iSCSI target	Adds iSCSI targets to a vCenter Server host. The targets can be of the type `Send` or `Static`.
Create VMFS for all available disks	Creates a VMFS volume for all available disks of a specified host.
Delete datastore	Deletes datastores from a vCenter Server host.
Delete iSCSI target	Deletes already configured iSCSI targets. The targets can be of type `Send` or `Static`.
Disable iSCSI adapter	Disables the software iSCSI adapter of a specified host.
Display all datastores and disks	Displays the existing datastores and available disks on a specified host.
Enable iSCSI adapter	Enables an iSCSI adapter.
List all storage adapters	Lists all storage adapters of a specified host.

Storage DRS Workflows

With the storage DRS workflows, you perform storage-related operations, such as creating and configuring a datastore cluster, removing a datastore from cluster, adding storage to a cluster, and others.

You can access the storage DRS workflows from **Library > vCenter > Storage > Storage DRS** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add datastore to cluster	Adds datastores to a datastore cluster. Datastores must be able to connect to all hosts to be included in the datastore cluster. Datastores must have the same connection type to reside within a datastore cluster.
Change Storage DRS per virtual machine configuration	Sets Storage DRS settings for each virtual machine.
Configure datastore cluster	Configures datastore cluster setting values for automation and runtime rules.
Create simple datastore cluster	Creates a simple datastore cluster with default configuration. The new datastore cluster contains no datastores.
Create Storage DRS scheduled task	Creates a scheduled task for reconfiguring a datastore cluster. Only automation and runtime rules can be set.
Create virtual machine anti-affinity rule	Creates an anti-affinity rule to indicate that all virtual disks of certain virtual machines must be kept on different datastores.
Create VMDK anti-affinity rule	Creates a VMDK anti-affinity rule for a virtual machine that indicates which of its virtual disks must be kept on different datastores. The rule applies to the virtual disks of the selected virtual machine.
Remove datastore cluster	Removes a datastore cluster. Removing a datastore cluster also removes all the settings and the alarms for the cluster from the vCenter Server system.
Remove datastore from cluster	Removes a datastore from a datastore cluster and puts the datastore in a datastore folder.

Workflow Name	Description
Remove Storage DRS scheduled task	Removes a scheduled Storage DRS task.
Remove virtual machine anti-affinity rule	Removes a virtual machine anti-affinity rule for a given datastore cluster.
Remove VMDK anti-affinity rule	Removes a VMDK anti-affinity rule for a given datastore cluster.

Storage VSAN Workflows

With the Virtual SAN workflows, you can manage non-SSD disks and disk groups in a Virtual SAN cluster.

You can access the networking workflows from **Library > vCenter > Storage > VSAN** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add disks to a disk group	Adds non-SSD disks to a Virtual SAN disk group.
Claim disks into disk groups	Claims disks for use by the Virtual SAN system and automatically creates disk groups and distributes the disks into existing disk groups.
Create a disk group	Creates a Virtual SAN disk group.
List hosts, disk groups and disks	Lists all hosts in a cluster, their disk groups and disks, used or eligible for use by the Virtual SAN system.
Remove disk groups	Removes Virtual SAN disk groups.
Remove disks from disk groups	Removes non-SSD disks from Virtual SAN disk groups.

Basic Virtual Machine Management Workflows

With the basic virtual machine management workflows, you can perform basic operations on virtual machines, for example, create, rename or delete a virtual machine, upgrade virtual hardware, and others.

You can access the basic virtual machine management workflows from **Library > vCenter > Virtual Machine management > Basic** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Create custom virtual machine	Creates a virtual machine with the specified configuration options and additional devices.
Create simple dvPortGroup virtual machine	Creates a simple virtual machine. The network used is a Distributed Virtual Port Group.
Create simple virtual machine	Creates a virtual machine with the most common devices and configuration options.
Delete virtual machine	Removes a virtual machine from the inventory and datastore.
Get virtual machines by name	Returns a list of virtual machines from all registered vCenter Server instances that match the provided expression.
Mark as template	Converts an existing virtual machine to a template, not allowing it to start. You can use templates to create virtual machines.

Workflow Name	Description
Mark as virtual machine	Converts an existing template to a virtual machine, allowing it to start.
Move virtual machine to folder	Moves a virtual machine to a specified virtual machine folder.
Move virtual machine to resource pool	Moves a virtual machine to a resource pool. If the target resource pool is not in the same cluster, you must use the migrate or relocate workflows.
Move virtual machines to folder	Moves several virtual machines to a specified virtual machine folder.
Move virtual machines to resource pool	Moves several virtual machines to a resource pool.
Register virtual machine	Registers a virtual machine. The virtual machine files must be placed in an existing datastore and must not be already registered.
Reload virtual machine	Forces vCenter Server to reload a virtual machine.
Rename virtual machine	Renames an existing virtual machine on the vCenter Server system or host and not on the datastore.
Set virtual machine performance	Changes performance settings such as shares, minimum and maximum values, shaping for network, and disk access of a virtual machine.
Unregister virtual machine	Removes an existing virtual machine from the inventory.
Upgrade virtual machine hardware (force if required)	Upgrades the virtual machine hardware to the latest revision that the host supports. This workflow forces the upgrade to continue, even if VMware Tools is out of date. If the VMware Tools is out of date, forcing the upgrade to continue reverts the guest network settings to the default settings. To avoid this situation, upgrade VMware Tools before running the workflow.
Upgrade virtual machine	Upgrades the virtual hardware to the latest revision that the host supports. An input parameter allows a forced upgrade even if VMware Tools is out of date.
Wait for task and answer virtual machine question	Waits for a vCenter Server task to complete or for the virtual machine to ask a question. If the virtual machine requires an answer, accepts user input and answers the question.

Clone Workflows

With clone workflows, you can clone virtual machines with or without customizing the virtual machine properties.

You can access the clone workflows from **Library > vCenter > Virtual Machine management > Clone** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Clone virtual machine from properties	Clones virtual machines by using properties as input parameters.
Clone virtual machine, no customization	Clones a virtual machine without changing anything except the virtual machine UUID.
Customize virtual machine from properties	Customizes a virtual machine by using properties as input parameters.

Linked Clone Workflows

With the linked clone workflows, you can perform linked clone operations such as restoring a virtual machine from a linked clone, creating a linked clone, or others.

You can access the linked clone workflows from **Library > vCenter > Virtual Machine management > Clone > Linked Clone** folder and its subfolders in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Restore virtual machine from linked clone	Removes a virtual machine from a linked clone setup.
Set up virtual machine for linked clone	Prepares a virtual machine to be link cloned.
Create a linked clone of a Linux machine with multiple NICs	Creates a linked clone of a Linux virtual machine, performs the guest operating system customization, and configures up to four virtual network cards.
Create a linked clone of a Linux machine with a single NIC	Creates a linked clone of a Linux virtual machine, performs the guest operating system customization, and configures one virtual network card.
Create a linked clone of a Windows machine with multiple NICs and credential	Creates a linked clone of a Windows virtual machine and performs the guest operating system customization. Configures up to four virtual network cards and a local administrator user account.
Create a linked clone of a Windows machine with a single NIC and credential	Creates a linked clone of a Windows virtual machine and performs the guest operating system customization. Configures one virtual network card and a local administrator user account.
Create a linked clone with no customization	Creates the specified number of linked clones of a virtual machine.

Linux Customization Clone Workflows

With Linux customization workflows, you can clone a Linux virtual machine and customize the guest operating system.

You can access the Linux customization clone workflows from **Library > vCenter > Virtual Machine management > Clone > Linux Customization** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Clone a Linux machine with multiple NICs	Clones a Linux virtual machine, performs the guest operating system customization, and configures up to four virtual network cards.
Clone a Linux machine with a single NIC	Clones a Linux virtual machine, performs the guest operating system customization, and configures one virtual network card.

Tools Clone Workflows

With the tools clone workflows, you can obtain customization information about the operating system of the virtual machine, information required to update a virtual device, and others.

You can access the tools clone workflows from **Library > vCenter > Virtual Machine management > Clone > Tools** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Get a virtual Ethernet card to change the network	Returns a new ethernet card to update a virtual device. Contains only the device key of the given virtual device and the new network.
Get Linux customization	Returns the Linux customization preparation.
Get multiple virtual Ethernet card device changes	Returns an array of `VirtualDeviceConfigSpec` objects for add and remove operations on `VirtualEthernetCard` objects.
Get NIC setting map	Returns the setting map for a virtual network card by using `VimAdapterMapping`.
Get Windows customization for Sysprep with credentials	Returns customization information about the Microsoft Sysprep process, with credentials. Workflows for cloning Windows virtual machines use this workflow.
Get Windows customization for Sysprep with `Unattended.txt`	Returns customization information about the Microsoft Sysprep process by using an `Unattended.txt` file. Workflows for cloning Windows virtual machines use this workflow.
Get Windows customization for Sysprep	Returns customization information about the Microsoft Sysprep process. Workflows for cloning Windows virtual machines use this workflow.

Windows Customization Clone Workflows

With the Windows customization clone workflows, you can clone Windows virtual machines and customize the guest operating system.

You can access the Windows customization clone workflows from **Library > vCenter > Virtual Machine management > Clone > Windows Customization** folder and its subfolder in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Customize a Windows machine with single NIC and credential	Performs guest operating system customization, configures one virtual network card and a local administrator user account on a Windows virtual machine.
Clone a thin provisioned Windows machine with single NIC and credential	Clones a Windows virtual machine performing the guest operating system customization. Configures one virtual network card and a local administrator user account. Sysprep tools must be available on vCenter Server.
Clone a Windows machine Sysprep with single NIC and credential	Clones a Windows virtual machine performing the guest operating system customization. Configures one virtual network card and a local administrator user account. Sysprep tools must be available on vCenter Server.
Clone a Windows machine with multiple NICs and credential	Clones a Windows virtual machine performing the guest operating system customization. Configures the local administrator user account and up to four virtual network cards. Sysprep tools must be available on the vCenter Server system.
Clone a Windows machine with single NIC	Clones a Windows virtual machine performing the guest operating system customization and configures one virtual network card. Sysprep tools must be available on the vCenter Server system.
Clone a Windows machine with single NIC and credential	Clones a Windows virtual machine performing the guest operating system customization. Configures one virtual network card and a local administrator user account. Sysprep tools must be available on the vCenter Server system.

Device Management Workflows

With the device management workflows, you can manage the devices that are connected to a virtual machine or to a host datastore.

You can access the device management workflows from **Library > vCenter > Virtual Machine management > Device Management** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add CD-ROM	Adds a virtual CD-ROM to a virtual machine. If the virtual machine has no IDE controller, the workflow creates one.
Add disk	Adds a virtual disk to a virtual machine.
Change RAM	Changes the amount of RAM of a virtual machine.
Convert disks to thin provisioning	Converts thick-provisioned disks of virtual machines to thin-provisioned disks.
Convert independent disks	Converts all independent virtual machine disks to normal disks by removing the independent flag from the disks.
Disconnect all detachable devices from a running virtual machine	Disconnects floppy disks, CD-ROM drives, parallel ports, and serial ports from a running virtual machine.
Mount CD-ROM	Mounts the CD-ROM of a virtual machine. If the virtual machine has no IDE controller or CD-ROM drive, the workflow creates them.
Mount floppy disk drive	Mounts a floppy disk drive FLP file from the ESXi datastore.

Move and Migrate Workflows

With the move and migrate workflows, you can migrate virtual machines.

You can access the move and migrate workflows from **Library > vCenter > Virtual Machine management > Move and Migrate** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Mass migrate virtual machines with storage vMotion	Uses Storage vMotion to migrate a single virtual machine, a selection of virtual machines, or all available virtual machines.
Mass migrate virtual machines with vMotion	Uses vMotion, Storage vMotion, or both vMotion and Storage vMotion to migrate a single virtual machine, a selection of virtual machines, or all available virtual machines.
Migrate virtual machine with vMotion	Migrates a virtual machine from one host to another by using the `MigrateVM_Task` operation from the vSphere API.
Move virtual machine to another vCenter Server system	Moves a list of virtual machines to another vCenter Server system.
Quick migrate multiple virtual machines	Suspends the virtual machines if they are powered on and migrates them to another host using the same storage.
Quick migrate virtual machine	Suspends the virtual machine if it is powered on and migrates it to another host using the same storage.
Relocate virtual machine disks	Relocates virtual machine disks to another host or datastore while the virtual machine is powered off by using the `RelocateVM_Task` operation from the vSphere API.

Other Workflows

With the workflows from the Others category, you can enable and disable Fault Tolerance (FT), extract virtual machine information, and find orphaned virtual machines.

You can access these workflows from **Library > vCenter > Virtual Machine management > Others** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Disable FT	Disables Fault Tolerance for a specified virtual machine.
Enable FT	Enables Fault Tolerance for a specified virtual machine.
Extract virtual machine information	Returns the virtual machine folder, host system, resource pool, compute resource, datastore, hard drive sizes, CPU and memory, network, and IP address for a given virtual machine. Might require VMware Tools.
Find orphaned virtual machines	Lists all virtual machines in an orphaned state in the Orchestrator inventory. Lists the VMDK and VMTX files for all datastores in the Orchestrator inventory that have no association with any virtual machines in the Orchestrator inventory. Sends the lists by email (optional).
Get Virtual Machine by Name and BIOS UUID	Searches virtual machines by name and then filters the result with particular universally unique identifier (UUID) in order to identify a unique virtual machine. **Note** This workflow is needed when DynamicOps calls vRealize Orchestrator workflows having input parameters of `VC:VirtualMachine` type in order to make the correspondence between a particular DynamicOps and vRealize Orchestrator virtual machine.
Get Virtual Machine by Name and UUID	Searches virtual machines by name and then filters the result with particular universally unique identifier (UUID) in order to identify a unique virtual machine. **Note** This workflow is needed when DynamicOps calls vRealize Orchestrator workflows having input parameters of `VC:VirtualMachine` type in order to make the correspondence between a particular DynamicOps and vRealize Orchestrator virtual machine.
Get Virtual Machine UUID	Searches virtual machines by name and then filters the result with particular universally unique identifier (UUID) in order to identify a unique virtual machine. **Note** This workflow is needed when DynamicOps calls vRealize Orchestrator workflows having input parameters of `VC:VirtualMachine` type in order to make the correspondence between a particular DynamicOps and vRealize Orchestrator virtual machine.

Power Management Workflows

With the power management workflows, you can power on and off virtual machines, reboot the guest operating system of a virtual machine, suspend a virtual machine, and others.

You can access the power management workflows from **Library > vCenter > Virtual Machine management > Power Management** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Power off virtual machine and wait	Powers off a virtual machine and waits for the process to complete.
Reboot guest OS	Reboots the guest operating system of the virtual machine. Does not reset nonpersistent virtual machines. VMware Tools must be running.
Reset virtual machine and wait	Resets a virtual machine and waits for the process to complete.

Workflow Name	Description
Resume virtual machine and wait	Resumes a suspended virtual machine and waits for the process to complete.
Set guest OS to standby mode	Sets the guest operating system to standby mode. VMware Tools must be running.
Shut down and delete virtual machine	Shuts down a virtual machine and deletes it from the inventory and disk.
Shut down guest OS and wait	Shuts down a guest operating system and waits for the process to complete.
Start virtual machine and wait	Starts a virtual machine and waits for VMware Tools to start.
Suspend virtual machine and wait	Suspends a virtual machine and waits for the process to complete.

Snapshot Workflows

With snapshot workflows, you can perform snapshot-related operations.

You can access the snapshot workflows from **Library > vCenter > Virtual Machine management > Snapshot** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Create a snapshot	Creates a snapshot.
Create snapshots of all virtual machines in a resource pool	Creates a snapshot of each virtual machine in a resource pool.
Remove all snapshots	Removes all existing snapshots without reverting to a previous snapshot.
Remove excess snapshots	Finds virtual machines with more than a given number of snapshots and optionally deletes the oldest snapshots. Sends the results by email.
Remove old snapshots	Gets all snapshots that are older than a given number of days and prompts the user to select which ones to delete.
Remove snapshots of a given size	Gets all snapshots that are larger than a given size and prompts the user to confirm deletion.
Revert to current snapshot	Reverts to the current snapshot.
Revert to snapshot and wait	Reverts to a specific snapshot. Does not delete the snapshot.

VMware Tools Workflows

With VMware Tools workflows, you can perform VMware Tools-related tasks on virtual machines.

You can access the VMware Tools workflows from **Library > vCenter > Virtual Machine management > VMware Tools** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Mount VMware Tools installer	Mounts the VMware Tools installer on the virtual CD-ROM.
Set console screen resolution	Sets the resolution of the console window. The virtual machine must be powered on.
Turn on time synchronization	Turns on time synchronization between the virtual machine and the ESXi server in VMware Tools.
Unmount VMware Tools installer	Unmounts the VMware Tools CD-ROM.

Workflow Name	Description
Update tools on Windows virtual machine without rebooting	Updates VMware Tools on a Windows virtual machine without performing a reboot.
Upgrade VMware Tools	Upgrades VMware Tools on a virtual machine.
Upgrade VMware Tools at next reboot	Upgrades VMware Tools on a virtual machine without performing an automatic reboot.

Using the vRealize Automation Plug-In

4

You use the vRealize Automation plug-in to run vRealize Orchestrator workflows from vRealize Automation.

The workflows provided with the plug-in help you deploy and manage resources from vRealize Automation. In addition to the provided workflows, you can create an run custom workflows.

This chapter includes the following topics:

Introduction to the VMware vRealize Orchestrator Plug-In for vRealize Automation

The VMware vRealize Orchestrator plug-in for vRealize Automation allows interaction between vRealize Orchestrator and vRealize Automation.

You can use the vRealize Automation plug-in to create and run workflows for the following vRealize Automation functions:

- XaaS custom resource and blueprint management
- Catalog item and resource management and requesting
- Entitlement configuration
- Approval policy configuration
- Work item interactions
- vSphere and vCloud Director virtual machine provisioning and post-provisioning actions
- Create, read, update, and delete (CRUD) operations on the vRealize Automation IaaS model

Role of vRealize Orchestrator with the vRealize Automation Plug-In

You use the Orchestrator client to run and create workflows and access the plug-in API. You can use either the embedded vRealize Orchestrator instance in your vRealize Automation installation, or an external vRealize Orchestrator server.

vRealize Orchestrator powers the vRealize Automation plug-in. vRealize Orchestrator is a development and process-automation platform that provides a library of extensible workflows to manage the VMware cloud stack and third party technologies.

vRealize Orchestrator allows integration with management and administration solutions through its open plug-in architecture.

Configuring the vRealize Automation Plug-In

You add vRealize Automation hosts and IaaS hosts to configure the plug-in.

Configuration Workflows

You can use the workflows in the **Configuration** workflow categories to manage vRealize Automation hosts.

vRealize Automation Hosts

You can access these workflows from the **Workflows** view of the Orchestrator client, in the **Configuration** subdirectory of the plug-in library.

Workflow Name	Description
Add a vRA host	Adds a vRealize Automation host to the plug-in inventory. For tenant management and administration tasks, you can use the **Inventory** view to run workflows on each tenant. To use the full function of the plug-in for a tenant, create a dedicated vRealize Automation host for each tenant.
Add a vRA host using component registry	Adds a vRealize Automation host to the plug-in inventory with a Per User Session connection. You must be logged in to the Orchestrator client with the credentials of the vRealize Automation system administrator. To use this function with an external vRealize Orchestrator server, you must register the Orchestrator server in the vRealize Automation component registry. **Note** To register an external vRealize Orchestrator server in the component registry, you must configure Orchestrator to use vRealize Automation as an authentication provider. For more information, see *Installing and Configuring VMware vRealize Orchestrator*.
Add the IaaS host of a vRA host	Adds the IaaS host of the selected vRealize Automation host to the plug-in inventory.
Remove a vRA host	Removes a vRealize Automation host from the plug-in inventory.

Workflow Name	Description
Update a vRA host	Updates a vRealize Automation host in the plug-in inventory.
Validate a vRA host	Validates the vRealize Automation host and the connection to it.

Note If your vRealize Orchestrator server is registered in the vRealize Automation component registry, a vRealize Automation host with the name Default is automatically added. The Default host is using Per User Session connection to the default tenant. The embedded Orchestrator server in the vRealize Automation installation is registered in the vRealize Automation component registry by default.

vRealize Automation IaaS Hosts

You can access these workflows from the **Workflows** view of the Orchestrator client, in the **Infrastructure Administration > Configuration** subdirectory of the plug-in library.

The embedded vRealize Orchestrator server in the vRealize Automation installation is registered in the vRealize Automation component registry by default.

Workflow Name	Description
Add an IaaS host	Adds a vRealize Automation IaaS host to the plug-in inventory. This workflow is functionally the same as Add the IaaS host of a vRA host, but does not require a vRealize Automation host.
Remove an IaaS host	Removes a vRealize Automation IaaS host from the plug-in inventory.
Update an IaaS host	Updates a vRealize Automation IaaS host in the plug-in inventory.
Validate an IaaS host	Validates the vRealize Automation IaaS host and the connection to it.

Add a vRealize Automation Host

You can run a workflow to add a vRealize Automation host and configure the host connection parameters.

Procedure

1 From the drop-down menu in the Orchestrator client, select **Run** or **Design**.

2 Click the **Workflows** view.

3 Expand **Library > vRealize Automation > Configuration**.

4 Right-click the **Add a vRA host** workflow and select **Start workflow**.

5 Enter a unique name for the host in the **Host Name** text box.

6 Enter the URL address of the host in the **Host URL** text box.

   For example: *https://hostname*.

7 (Required) Enter the name of the tenant in the **Tenant** text box.

   To use the full functionality of the plug-in for a tenant, create a dedicated vRealize Automation host for each tenant.

8 Select whether to install the SSL certificates automatically without user confirmation.

9 (Optional) To configure the length of time vRealize Orchestrator waits for a connection or response from vRealize Automation, enter timeout intervals in the **Connection timeout (seconds)** and **Operation timeout (seconds)** text boxes.

10 Select the type of connection to the host from the **Session mode** drop-down menu.

Option	Actions
Shared Session	Enter the credentials for a vRealize Automation user in the **Authentication username** and **Authentication password** text boxes.
Per User Session	Connect using the credentials of the user that is currently logged in. You must be logged in to the Orchestrator client with the credentials of the vRealize Automation system administrator. To use this option with an external vRealize Orchestrator server, you must register the Orchestrator server in the vRealize Automation component registry. **Note** To register an external vRealize Orchestrator server in the component registry, you must configure Orchestrator to use vRealize Automation as an authentication provider. For more information, see *Installing and Configuring VMware vRealize Orchestrator*.

11 Click **Submit**.

What to do next

Add a vRealize Automation Infrastructure Administration host.

Add an IaaS Host

You can run a workflow to add the IaaS host of a vRealize Automation host and configure the connection parameters.

Procedure

1 From the drop-down menu in the Orchestrator client, select **Run** or **Design**.

2 Click the **Workflows** view.

3 Expand **Library > vRealize Automation > Infrastructure Administration > Configuration**.

4 Right-click **Add an IaaS host** and select **Start workflow**.

5 Select the vRealize Automation host for which you want to configure an IaaS host from the **vCAC host** drop-down menu.

6 Enter a unique name for the host in the **Host Name** text box.

7 Enter the URL of the machine on which your Model Manager is installed.

For example: https://*model_manager_machine.com*.

8 To install the SSL certificates, select **Yes**.

9 To use a proxy to access your model manager machine, select **Yes**.

   If you select this option, you must provide the proxy host and the proxy port on the following page.

10 Click **Next**.

11 If you are configuring an explicit proxy, provide the proxy host URL and the port.

12 Click **Next**.

13 To configure your own timeout values, click **No**.

14 (Optional) To configure the length of time vRealize Orchestrator waits for a connection or response from vRealize Automation, enter timeout intervals in the **Connection timeout (seconds)** and **Operation timeout (seconds)** text boxes.

15 Click **Next**.

16 Select the host's authentication type.

Option	Description
SSO	Select this to use vCenter Single Sign-On.
NTLM	Select this to enable NT LAN Manager (NTLM) protocol-based authentication only if your Active Directory infrastructure relies on NTLM authentication. If you select this option, you must the additional NTLM credentials and authentication options.

17 If you selected NTLM, click **Next** and enter the name of the Workstation machine and the NetBIOS domain name.

18 Click **Submit**.

Using the vRealize Automation Plug-In Workflows

The vRealize Automation plug-in workflow library contains workflows that you can use for common tasks such as interacting with the catalog, managing infrastructure, and creating tenants and services.

You can use custom HTTP headers, such as the vRealize Automation specific headers Tasks and Identity, and apply them in the CRUD, provisioning, and post-provisioning workflows.

Remove Operation Restrictions

Some create, read, update, and delete operations are restricted beginning with version 7.0. If you used the operations in your workflows in previous versions, they will not work with 7.0 and later. You can update your workflows to the supported operations or you can re-enable the operations that you need.

To re-enable the operations, you must remove the operations that you want to enable from the `operations.properties` file. For a list of the operations in the file, see Restricted Operations.

Procedure

1 From the drop-down menu in vRealize Orchestrator, select **Design**.

2 Click the **Resources** view.

3 In the resource hierarchy, expand **Library > VCAC > Util**.

4 Create a backup and modify the `operations.properties` file.

   a Right-click `operations.properties` and select **Save to file**.

   b Save a copy as a backup.

   c Create a new copy and delete the operations that you want to re-enable.

   d Save the new file.

5 Replace the existing file in vRealize Orchestrator.

   a In vRealize Orchestrator, right-click the **Util** folder and click **Import resources**.

   b Browse to the new version of the `operations.properties` file and click **Open**.

   c Click **Replace once** to save your modified version.

6 Restart the vRealize Orchestrator server.

7 Select the `operations.properties` file and click the **Viewer** tab.

8 Verify that the operations that you are enabling are no longer in the file.

Results

The operations that you removed from the file now work in your older workflows.

What to do next

As you create new workflows, avoid using the restricted operations.

Restricted Operations

The contents of the `operations.properties` file contains the restricted operations. To re-enable the operation, you must remove it from the file.

The following text is the default version of the `operations.properties` file. To re-enable an operation, see Remove Operation Restrictions.

```
#Blueprints
operation.create=ManagementModelEntities.svc@VirtualMachineTemplates
operation.update=ManagementModelEntities.svc@VirtualMachineTemplates
operation.delete=ManagementModelEntities.svc@VirtualMachineTemplates
#Blueprint properties
operation.create=ManagementModelEntities.svc@VirtualMachineProperties
operation.read=ManagementModelEntities.svc@VirtualMachineProperties
operation.update=ManagementModelEntities.svc@VirtualMachineProperties
operation.delete=ManagementModelEntities.svc@VirtualMachineProperties
#Global profiles
operation.create=ManagementModelEntities.svc@GlobalProfiles
operation.read=ManagementModelEntities.svc@GlobalProfiles
operation.update=ManagementModelEntities.svc@GlobalProfiles
operation.delete=ManagementModelEntities.svc@GlobalProfiles
#Global profile properties
operation.create=ManagementModelEntities.svc@GlobalProfileProperties
operation.read=ManagementModelEntities.svc@GlobalProfileProperties
operation.update=ManagementModelEntities.svc@GlobalProfileProperties
operation.delete=ManagementModelEntities.svc@GlobalProfileProperties
#PropertySetXml
operation.create=ManagementModelEntities.svc@PropertySetXml
operation.read=ManagementModelEntities.svc@PropertySetXml
operation.update=ManagementModelEntities.svc@PropertySetXml
operation.delete=ManagementModelEntities.svc@PropertySetXml
#Property definitions
operation.create=ManagementModelEntities.svc@PropertyDefinitions
operation.read=ManagementModelEntities.svc@PropertyDefinitions
operation.update=ManagementModelEntities.svc@PropertyDefinitions
operation.delete=ManagementModelEntities.svc@PropertyDefinitions
#Property attributes
operation.create=ManagementModelEntities.svc@PropertyAttributes
operation.read=ManagementModelEntities.svc@PropertyAttributes
operation.update=ManagementModelEntities.svc@PropertyAttributes
operation.delete=ManagementModelEntities.svc@PropertyAttributes
#Property Attribute Types
operation.create=ManagementModelEntities.svc@PropertyAttributeTypes
operation.read=ManagementModelEntities.svc@PropertyAttributeTypes
operation.update=ManagementModelEntities.svc@PropertyAttributeTypes
operation.delete=ManagementModelEntities.svc@PropertyAttributeTypes
#Control layouts
operation.create=ManagementModelEntities.svc@ControlLayouts
operation.read=ManagementModelEntities.svc@ControlLayouts
operation.update=ManagementModelEntities.svc@ControlLayouts
operation.delete=ManagementModelEntities.svc@ControlLayouts
#Amazon Virtual Machine Templates
operation.create=AmazonWSModelEntities.svc@AmazonVirtualMachineTemplates
operation.read=AmazonWSModelEntities.svc@AmazonVirtualMachineTemplates
operation.update=AmazonWSModelEntities.svc@AmazonVirtualMachineTemplates
operation.delete=AmazonWSModelEntities.svc@AmazonVirtualMachineTemplates
#Openstack Virtual Machine Templates
operation.create=OpenStackModelEntities.svc@OpenstackVirtualMachineTemplates
operation.read=OpenStackModelEntities.svc@OpenstackVirtualMachineTemplates
operation.update=OpenStackModelEntities.svc@OpenstackVirtualMachineTemplates
operation.delete=OpenStackModelEntities.svc@OpenstackVirtualMachineTemplates
#Endpoint credentials
operation.create=ManagementModelEntities.svc@ConnectionCredentials
operation.update=ManagementModelEntities.svc@ConnectionCredentials
operation.delete=ManagementModelEntities.svc@ConnectionCredentials
#Management endpoints
operation.create=ManagementModelEntities.svc@ManagementEndpoints
operation.update=ManagementModelEntities.svc@ManagementEndpoints
operation.delete=ManagementModelEntities.svc@ManagementEndpoints
```

```
#Management endpoint properties
operation.create=ManagementModelEntities.svc@ManagementEndpointProperties
operation.read=ManagementModelEntities.svc@ManagementEndpointProperties
operation.update=ManagementModelEntities.svc@ManagementEndpointProperties
operation.delete=ManagementModelEntities.svc@ManagementEndpointProperties
```

Using the vRealize Automation Plug-In Inventory

You can use the **Inventory** view to run workflows on vRealize Automation objects.

To display the workflows that are available for an inventory object, navigate to **Tools > User preferences > Inventory** and select the **Use contextual menu in inventory** check box. After the option is enabled, when you right-click an object in the Orchestrator inventory, all available workflows for the object are displayed.

Using the vRealize Automation Plug-In Administration Workflows

You can use the administration workflows to manage vRealize Automation services, tenants, approval policies, entitlements, business groups, catalog items, and Advanced Services components.

Some of the workflows include an input parameter for the vRealize Automation host, `vCACCAFE:VCACHost`. How you configured the vRealize Automation host connection determines how the roles are applied when a user runs the workflows.

- If you configured the connection as a shared session, the user account for the shared session must have the roles required to run the workflows.
- If you configured the connection as session per user, the each user who runs the workflows must have the required roles, just as they would in the vRealize Automation user interface.

You can find these workflows on the **Workflows** view in the vRealize Orchestrator client, in the **Library > vRealize Automation > Administration** subdirectory.

You can use the workflows in the **Approval Policies** subdirectory to create and manage approval policies.

Table 4-1. Approval Policies

Workflow	Description
Activate an approval policy	Activates an approval policy. After you activate an approval policy, it becomes read-only.
Add an approval level	Adds an always required approval level to an approval. You must select specific users and groups for the approvers.
Copy an approval policy	Copies an approval policy.
Create an approval policy	Creates a draft approval policy with no levels or approvers. To create approval levels and designate approvers for your policy, run the Add an approval level workflow.

Table 4-1. Approval Policies (continued)

Workflow	Description
Deactivate an approval policy	Deactivates an approval policy. You can also delete all existing entitlements associated with the approval policy.
Delete an approval policy	Deletes an approval policy that is in draft state. Active approval policies are read-only.

You can use the workflows in the **Business Groups** subdirectory to create and manage business groups and business group custom properties.

Table 4-2. Business Groups

Workflow	Description
Add a custom property	Adds a custom property to a business group.
Create a business group	Creates a business group.
Delete a business group	Deletes a business group.
Delete a custom property	Removes a custom property from a business group.
Update a business group	Updates details for a business group, such as default machine prefix, active directory containers, and user roles.
Update a custom property	Updates a custom property for a business group.

The Administration subdirectory includes a **Business Groups (Deprecated)** subdirectory that works with versions before vRealize Automation 7.0. Use the workflows with the same name in the main folder.

You can use the workflows in the **Catalog Items** subdirectory to manage catalog items.

Table 4-3. Catalog Items

Workflow	Description
Activate a catalog item	Activates a catalog item. You must activate and assign a catalog item to a service before users can request it.
Assign a catalog item to a service	Assigns a catalog item to a service. You must activate and assign a catalog item to a service before users can request it.
Deactivate a catalog item	Deactivates a catalog item and removes it from the service catalog so that users cannot request it.

You can use the workflows in the **Composite Blueprint** subdirectory to manage composite blueprints create in the design canvas.

Table 4-4. Composite Blueprint

Workflow	Description
Delete a composite blueprint	Delete an unpublished blueprint form the Design blueprints list.
Import a composite blueprint	Import a composite blueprint from a YAML file.

Table 4-4. Composite Blueprint (continued)

Workflow	Description
Publish a composite blueprint	Publish a composite blueprint that is in a draft state.
Unpublish a composite blueprint	Unpublish a published composite blueprint.

The **Content** subdirectory workflows are deprecated. Use Cloud Client to perform the import and export actions. The Cloud Client download and documentation are available at the https://developercenter.vmware.com/tool/cloudclient.

Table 4-5. Content

Workflow	Description
Export content (deprecated)	Use Cloud Client to perform the import and export actions. The Cloud Client download and documentation are available at the https://developercenter.vmware.com/tool/cloudclient.
Import content (deprecated)	Use Cloud Client to perform the import and export actions. The Cloud Client download and documentation are available at the https://developercenter.vmware.com/tool/cloudclient.
Transfer content (deprecated)	Use Cloud Client to perform the import and export actions. The Cloud Client download and documentation are available at the https://developercenter.vmware.com/tool/cloudclient.
Validate content (deprecated)	Use Cloud Client to perform the import and export actions. The Cloud Client download and documentation are available at the https://developercenter.vmware.com/tool/cloudclient.

You can use the workflows in the **Entitlements** subdirectory to create and manage entitlements.

Table 4-6. Entitlements

Workflow	Description
Activate an entitlement	Activates an entitlement.
Assign catalog items to an entitlement	Assigns one or more catalog items to an entitlement. You can also use this workflow to assign an approval policy.
Assign immediate actions to an entitlement	Assigns one or more immediate actions to an entitlement. The immediate actions do not create requests.
Assign resource actions to an entitlement	Assigns one or more resource actions to an entitlement. You can also use this workflow to assign an approval policy.
Assign services to an entitlement	Assigns one or more services to an entitlement. You can also use this workflow to assign an approval policy.
Assign users and groups to an entitlement	Assigns one or more users or groups to an entitlement.
Create an entitlement (deprecated)	Creates an entitlement. Use Create and entitlement for subtenant.

Table 4-6. Entitlements (continued)

Workflow	Description
Create an entitlement for subtenant	Creates an entitlement.
Deactivate an entitlement	Deactivates an entitlement.
Unassign users and groups from an entitlement	Remove users and groups from the list of users for an entitlement.

You can use the workflows in the **Properties** subdirectory to manage property definitions and property groups. To avoid conflict with vRealize Automation properties, use a prefix such as a company or feature name followed by a dot for all custom property names.

Table 4-7. Property Definitions

Workflow	Description
Create property definition	Creates a custom property.
Delete property definition	Deletes a custom property.

Property groups are collections of property definitions.

Table 4-8. Property Groups

Workflow	Description
Add property to group	Adds a defined custom property to a group.
Create property group	Creates a property group to which you can add defined custom properties.
Delete property group	Deletes the property group.
Remove properties from group	Removes a defined custom property from a property group.
Update property group	Modifies the name or description of the property group.
Update property in group	Modifies the name, value, and behavior of the property in the property group.

You can use the workflows in the **Services** subdirectory to manage services.

Table 4-9. Services

Workflow	Description
Activate a service	Activates a service.
Assign catalog items to a service	Assigns one or more catalog items to a service.
Copy a service	Copies a service.
Create a service	Creates a service.
Deactivate a service	Deactivates a service.
Delete a service	Deletes a service.

You can use the workflows in the **Tenants** subdirectory to create and manage tenants.

The identify store workflows are deprecated. The replacement workflows work with the changes to vRealize Automation for the Directories Management API.

Table 4-10. Tenants

Workflow	Description
Add administrators	Adds one or more tenant administrators and infrastructure administrators to a tenant.
Add an identity store to a tenant	Adds an identity store to a tenant of a vRealize Automation host. You can run this workflow only if you are a system administrator configuring a tenant.
Add an identity store to a tenant (Deprecated)	Use the Add an identity store to a tenant workflow.
Add an identity store to a vCAC host	Adds an identity store to a tenant that is configured as a vRealize Automation host. You can run this workflow only if you are a tenant administrator configuring an identity store for your tenant.
Add an identity store to a vCAC host (Deprecated)	Use the Add an identity store to a vCAC host workflow.
Create a tenant	Creates a tenant. You must select the vRealize Automation host added with the system administrator credentials.
Delete an identity store from a tenant	Deletes an identity store from a tenant of a vRealize Automation host. You can run this workflow only if you are a system administrator configuring a tenant.
Delete an identity store from a vCAC host	Deletes an identity store from a tenant that is configured as a vRealize Automation host. You can run this workflow only if you are a tenant administrator configuring identity stores for your tenant.
Delete a tenant	Deletes a tenant.
Remove administrators	Removes one or more tenant administrators and infrastructure administrators from a tenant.
Update an identity store for a tenant	Updates an existing identity store for a tenant of a vRealize Automation host. You can run this workflow only if you are a system administrator configuring a tenant.
Update an identity store for a tenant (Deprecated)	Use the Update an identity store for a tenant workflow.
Update an identity store for a vCAC host	Updates an identity store for a tenant that is configured as a vRealize Automation host. You can run this workflow only if you are a tenant administrator configuring identity stores for your tenant.
Update an identity store for a vCAC host (Deprecated)	Use the Update an identity store for a vCAC host workflow.
Update a tenant	Updates the name, description, and contact email address of an existing tenant.

You can use the workflows in the **Workflow Subscriptions** subdirectory to manage the event workflow subscriptions.

Table 4-11. Workflow Subscriptions

Workflow	Description
Delete a workflow subscription	Delete an unpublished workflow subscription. This workflow applies to system and tenant workflow subscriptions.
Export system workflow subscription	Export a system workflow subscription and save it as a vRealize Orchestrator resource element in a JSON format. A system workflow subscription is a specialized workflow subscription that reacts to system events and to events in all the tenants.
Export tenant workflow subscription	Export a tenant workflow subscription and save it as a resource element in a JSON format. A specialized workflow subscription that runs tenant-specific workflows.
Import system workflow subscription	Import a system workflow subscription from a JSON file. System workflow subscriptions are triggered for system events and can be across tenants.
Import tenant workflow subscription	Import an exported workflow subscription from a JSON file. These workflow subscriptions are tenant-specific.
Publish a workflow subscription	Publish a workflow subscription that was in a draft or unpublished state. This workflow applies to system and tenant workflow subscriptions.
Register a system workflow subscription	Create a system workflow subscription, including timeout and priority values.
Register a tenant workflow subscription	Create a tenant-specific workflow subscription, including timeout and priority values.
Unpublish a workflow subscription	Unpublish a published workflow subscription. This workflow applies to system and tenant workflow subscriptions.
Update a workflow subscription	Change the name, description, vRealize Orchestrator workflow, subscription conditions, timeout value, status value, and priority value. You cannot update the event topic or blocking state.

You can use the workflows in the **XaaS Custom Resources** subdirectory to create and delete XaaS custom resources.

Table 4-12. XaaS Custom Resources

Workflow	Description
Create custom resource	Creates a custom resource.
Delete custom resource	Removes a custom resource.

You can use the workflows in the **XaaS Resource Actions** subdirectory to create and manage XaaS resource actions.

Table 4-13. XaaS Resource Actions

Workflow	Description
Clone resource action	Creates a copy of an existing resource action.
Create resource action	Creates a resource action.
Delete resource action	Deletes a resource action.
Publish resource action	Publishes a resource action.
Unpublish resource action	Unpublishes a resource action.

You can use the workflows in the **XaaS Resource Mappings** subdirectory to create and manage XaaS mappings to non-XaaS resources.

Table 4-14. XaaS Resource Mappings

Workflow	Description
Create a resource mapping	Map a catalog resource type to a vRealize Orchestrator type.
Delete a resource mapping	Delete a resource mapping.
Set a target criteria	Specify the conditions that determine the availability of the resource mapping.

You can use the workflows in the **XaaS Server Configuration** subdirectory to manage the target Orchestrator instance.

Table 4-15. XaaS Server Configuration

Workflow	Description
Update Orchestrator server configuration	Modify the server settings, including port, host, user name, and password.
Validate Orchestrator server configuration	Verifies that the vRealize Orchestrator settings are valid. The workflow returns a value of TRUE if the configuration is valid, and FALSE if the configuration is not valid.

You can use the workflows in the **XaaS Service Blueprints** subdirectory to create and manage XaaS blueprints.

Table 4-16. XaaS Blueprints

Workflow	Description
Clone a service blueprint	Creates a copy of a service blueprint.
Create a service blueprint	Creates a service blueprint.
Delete a service blueprint	Deletes a service blueprint.
Publish a service blueprint	Publishes a service blueprint.
Unpublish a service blueprint	Unpublishes a service blueprint.

Using the vRealize Automation Plug-In Infrastructure Administration Workflows

You can use the infrastructure administration workflows to run basic operations. You use the extensibility package to customize vRealize Automation with the ability to call vRealize Orchestrator workflows either as part of the provisioning process, or by using custom operation menus.

You can find the infrastructure administration workflows on the **Workflows** view of the Orchestrator client, in the **Infrastructure Administration** subdirectory of the plug-in library.

You can use the infrastructure administration workflows to provision virtual machines and run basic create, read, update, or delete operations.

Table 4-17. Infrastructure Administration

Workflow Name	Description
Await virtual machine state change	Awaits a state change for a set of virtual machines. If all virtual machines are in the success state, a trigger is called and the workflow ends successfully. If any of the specified virtual machines gets into the fail state, or does not exist, the workflow fails. You must enter the success and fail states selecting from the following options: ■ `Requested` ■ `AwaitingApproval` ■ `RegisterMachine` ■ `BuildingMachine` ■ `AddingDisks` ■ `MachineProvisioned` ■ `MachineActivated` ■ `InstallTools` (VMware only) ■ `On` ■ `Off` ■ `TurningOn` ■ `TurningOff` ■ `ShuttingDown` ■ `Suspending` ■ `Resetting` ■ `Rebooting` ■ `Expired` ■ `DeactivateMachine` ■ `UnprovisionMachine` ■ `Disposing` ■ `Finalized`
Create an IaaS model entity	Creates and persists an entity for a specified vRealize Automation model.
Delete an IaaS model entity	Deletes a specified vRealize Automation model entity.
Invoke a post-provisioning action (deprecated)	Use the Request a resource action workflow.

Table 4-17. Infrastructure Administration (continued)

Workflow Name	Description
Provision a virtual machine from a blueprint (removed in vRealize Automation 7.0)	Replaced by Request a catalog item or Request a catalog item with provisioning request.
Read an IaaS entity by custom filter	Reads a list of vRealize Automation entities by using a custom filter. If you do not specify a filter, all entities are returned as a result.
Read an IaaS entity by system query	Reads a list of vRealize Automation entities by using OData system filters. The system filters apply to the OData URI convention.
Read an IaaS model entity	Reads a vRealize Automation model entity by its ID.
Update an IaaS model entity	Updates a vRealize Automation model entity by its ID.

You use the workflows in the **Extensibility** subdirectory to customize vRealize Automation with the ability to call vRealize Orchestrator workflows either as part of the provisioning process, or by custom operation menus.

The subdirectory also includes workflows for managing IaaS credentials, endpoints, enterprise groups, machine prefixes, and other entities.

Table 4-18. Extensibility

Workflow Name	Description
Install vCO customization	Installs an Orchestrator customization, including customized state change workflows and menu operations workflows.
Uninstall vCO customization	Uninstalls an Orchestrator customization, including customized state change workflows and menu operations workflows.
Change reservation of an IaaS Virtual Machine	Changes the attributes, such as reservations and business groups, of a managed virtual machine.
Import an IaaS Virtual Machine (deprecated)	Use Cloud Client. The Cloud Client download and documentation are available at the https://developercenter.vmware.com/tool/cloudclient.
Import vCenter Virtual Machine (deprecated)	Use Cloud Client. The Cloud Client download and documentation are available at the https://developercenter.vmware.com/tool/cloudclient.
Unregister virtual machine (removed in vRealize Automation 7.0)	No replacement workflow is provided.
Assign a menu operation to a blueprint and its virtual machines (Deprecated)	Adds or updates a menu operation on virtual machines. Alternative non-deprecated workflows include Assign resource actions to an entitlement and Import a composite blueprint.
Assign a menu operation to virtual machines (Deprecated)	Updates a vRealize Automation model entity by its ID. Alternative non-deprecated workflows include Assign resource actions to an entitlement and Import a composite blueprint.

Table 4-18. Extensibility (continued)

Workflow Name	Description
Assign a state change workflow to a blueprint and its virtual machines (Deprecated)	Replaced by event broker subscriptions in vRealize Automation.
Customize a menu operation (removed in vRealize Automation 7.0)	No replacement workflow is provided.
Remove a menu operation from a blueprint and its virtual machines (removed in vRealize Automation 7.0)	No replacement workflow is provided.
Remove a state change workflow from a blueprint and its virtual machines	Removes a state change workflow from a blueprint and its virtual machines.

Create a vRealize Automation IaaS Model Entity

You can run a workflow to create a simple or complex vRealize Automation IaaS entity, such as a virtual machine reference to a user.

Procedure

1. From the drop-down menu in the Orchestrator client, select **Run** or **Design**.
2. Click the **Workflows** view.
3. Expand **Library > vRealize Automation > Infrastructure Administration**.
4. Right-click the **Create an IaaS model entity** workflow and select **Start workflow**.
5. Select a vRealize Automation host.
6. Enter the name of the model in the **Model name** text box.
7. Enter the name of the entity set, in the **Entity set name** text box.

   You use scripting or a REST API to set the Simple properties, Links to complex properties, and HTTP headers properties.
8. Click **Submit** to run the workflow.

Read a vRealize Automation IaaS Model Entity

You can run a workflow to read a vRealize Automation IaaS model entity.

Procedure

1. From the drop-down menu in the Orchestrator client, select **Run** or **Design**.
2. Click the **Workflows** view.
3. Expand **Library > vRealize Automation > Infrastructure Administration**.
4. Right-click **Read an IaaS model entity** and select **Start workflow**.
5. Select a vRealize Automation host.
6. Enter the name of the model in the **Model name** text box.

7 Enter the name of the entity set, in the **Entity set name** text box.

   You use scripting or a REST API to set the HTTP headers property.

8 Click **Submit** to run the workflow.

Using the vRealize Automation Plug-In Requests Workflows

You can use the requests workflows to request catalog items and resource actions, and to complete or cancel work items.

A work item requires user input or action. For example, a workflow interaction, approval action, or responding to a reclamation request.

You can access these workflows from the **Workflows** view of the vRealize Orchestrator client, in the **Requests** subdirectory of the plug-in library.

Workflow	Description
Cancel a work item	Cancels an active work item. You can use this workflow only if you are a system administrator.
Complete a work item	Finishes a work item based on provided user input.
Request a catalog item	Requests a catalog item for the user running the workflow. If you need a workflow to request a composite blueprint, use the Request a catalog with provisioning request workflow.
Request a catalog item on behalf of a user	Sends a request for a catalog item on behalf of a user. You can use this workflow only for catalog items entitled to both you and the user on behalf of whom you are sending the request.
Request a catalog with provisioning request	Requests a composite blueprint as a catalog item for the user running the workflow. If you are providing customized input to the request, you must customize the workflow. Use this workflow for composite blueprints.
Request a resource action	Requests a resource action for a catalog item owned by the user running the workflow.
Request a resource action on behalf of a user	Sends a request for a resource action on behalf of a user. You can use this workflow only for resource actions entitled to both you and the user on behalf of whom you are sending the request.

Workflow	Description

Request a resource action with a request template	Requests a resource action that includes complex parameters. The best practice is to duplicate the workflow and then customize it for the action. You can use the workflow to pass complex parameters or hidden parameters that you do not want to appear on the request form. One of the primary applications of this workflow is to customize the IaaS reconfigure virtual machine action.
	To create a reconfigure operation on a virtual machine, you must create a copy of the workflow and then modify the script. Configure the parameters that appear in vRealize Orchestrator and set the `Cafe.Shim.VirtualMachine.Reconfigure.Requestor` parameter. This parameter is used for logging and it must not be empty. See the following example.
	`var requestTemplate =`
	`vCACCAFERequestsHelper.getRequestForResourceAction(operation)`
	
	`var jsonData =`
	`vCACCAFERequestsHelper.getResourceActionRequestData(requestTemplate);`
	`var json = JSON.parse(jsonData);`
	`//Change cpu example`
	`json.cpu = 2;`
	
	`//This is a property needed for the Reconfigure IaaS operation:`
	`json["Cafe.Shim.VirtualMachine.Reconfigure.Requestor"] = 1;`
	`vCACCAFERequestsHelper.setResourceActionRequestData(requestTemplate,`
	`JSON.stringify(json));`
	
	`request =`
	`System.getModule("com.vmware.library.vcaccafe.request").requestResourceActionWith`
	`RequestTemplate(operation, requestTemplate);`

Wait for a catalog item request	Waits for a catalog item request to finish.
Wait for a resource action request	Waits for a resource action request to finish.
Wait for a work item	Waits for a work item to finish.

Using the vRealize Automation Plug-In Sample Workflows

You can use the sample workflows as examples, or as starting points for creating your own custom workflows.

You can find these workflows on the **Workflows** view of the vRealize Orchestrator client, in the **Sample** subdirectory of the plug-in library.

Workflow Name	Description
Create a permission	Provides a sample script that interacts with the authorization client and the permission service to create a permission in vRealize Automation.
Create a tenant	Creates a tenant with the same vRealize Automation host and Active Directory configuration as the default tenant. To run this workflow, select the vRealize Automation host that was added with your system administrator credentials. You can change the Active Directory settings before running the workflow.
List catalog items	Returns a list of catalog items for the selected tenant.

Workflow Name	Description
Print catalog item provisioning request as JSON	Retrieves the default request form for a catalog item and adds it to the console log in JSON format. You can be used the data to customize a provisioning request. You can use the information to modify the **Request a catalog item with a provisioning request** workflow.

Access the vRealize Automation Plug-In API

Orchestrator provides an API Explorer to allow you to search the vRealize Automation plug-in API and see the documentation for JavaScript objects that you can use in scripted elements.

For updated vRealize Automation API documentation, see https://www.vmware.com/support/pubs/vcac-pubs.html.

Procedure

1. Log in to the Orchestrator client as an administrator.
2. Select **Tools > API Explorer**.
3. Double-click the **vCAC** and **VCACCAFE** modules in the left pane to expand the hierarchical list of vRealize Automation plug-in API objects.

What to do next

You can copy code from API elements and paste it into scripting boxes. For more information about API scripting, see *Developing with VMware vRealize Orchestrator*.

For additional information about development best practices, see vRealize Orchestrator Documentation.

Example vRealize Automation Plug-In Scripts

You can cut, paste, and edit the JavaScript examples provided to develop your own custom scripts for automating vRealize Automation tasks.

CRUD Infrastructure Administration Tasks Example Scripts

You can cut, paste, and edit the JavaScript examples to write scripts for CRUD vRealize Automation tasks.

For more information about scripting in vRealize Orchestrator, see *Developing with VMware vRealize Orchestrator*.

Example: Create a vRealize Automation Model Entity

This example script performs the following actions:

1. Defines the model name and the entity set name.
2. Defines the properties of the host prefix.
3. Saves the host prefix entity.

4 Defines the properties of the provisioning group.

5 Defines the provisioning group as a link.

6 Saves the provisioning group entity, by linking it with the host name prefix.

Table 4-19. Input Variables

Variable	Type
host	vCAC:VcacHost

```
var modelName = 'ManagementModelEntities.svc';
var entitySetName = 'HostNamePrefixes';
var links = null;
var headers = null;
//Create properties for prefix entity
var prefixInputProperties = {
   MachinePrefix:'test-prefix',
   NextMachineNo:1,
   MachineNumberLength:3
};
//Save the prefix
var prefixEntity = vCACEntityManager
  .createModelEntity(host.id, modelName, entitySetName, prefixInputProperties, links, headers);
entitySetName = 'ProvisioningGroups';
//Create properties for the provisioning group entity
inputProperties = {
   GroupName:'TestGroupName',
   GroupDescription:'This group was generated with a vCO workflow',
   AdministratorEmail:'test@test.com',
   AdContainer:'AD',
   IsTestGroup:false,
   Flags:2,
   GroupType:1};
//Add a reference to the newly created prefix entity
links = {
 HostNamePrefix:prefixEntity
};
//Save the provisioning group
var entity = vCACEntityManager.createModelEntity(host.id, modelName, entitySetName, inputProperties, 
links, headers);
```

Example: Update a vRealize Automation Model Entity

This example script performs the following actions:

1 Gets the host ID from the provided entity.

2 Gets the model name from the provided entity.

3 Gets the entity set name from the provided entity.

4 Gets the entity ID from the provided entity.

5 Defines a set of properties that will be updated.

6 Starts the action responsible for updating the entity.

Table 4-20. Input Variables

Variable	Type
`entity`	`vCAC:Entity`
`updatedDescription`	String

```
var hostId = entity.hostId;
var modelName = entity.modelName;
var entitySetName = entity.entitySetName;
var entityIdString = entity.keyString;
var links = null;
var headers = null;
var updateProperties = new Properties();
updateProperties.put("UserNameDescription", updatedDescription);
//Update the user description
System.getModule("com.vmware.library.vcac")
  .updateVCACEntity(hostId, modelName, entitySetName, entityIdString, updateProperties, links,
headers);
```

Example: Read a vRealize Automation Model Entity

This example script performs the following actions:

1 Defines the model name and the entity set name.

2 Defines the blueprint ID with a property object.

3 Reads the entity.

Table 4-21. Input Variables

Variable	Type
`host`	`vCAC:VcacHost`
`blueprintID`	String

```
var modelName = 'ManagementModelEntities.svc';
var entitySetName = 'VirtualMachineTemplates';
var links = null;
var headers = null;
//Create properties for the prefix entity
var blueprintId = {
  VirtualMachineTemplateID:blueprintId,
};
//Read the blueprint
var entity = vCACEntityManager
  .readModelEntity(host.id, modelName, entitySetName, blueprintId, headers);
```

Example: Delete a vRealize Automation Model Entity

This example script performs the following actions:

1 Gets the host ID from the provided entity.

2 Gets the model name from the provided entity.

3 Gets the entity set name from the provided entity.

4 Gets the entity ID from the provided entity.

5 Starts the action responsible for deleting the entity.

Table 4-22. Input Variables

Variable	Type
entity	vCAC:Entity

```
var hostId = entity.hostId;
var modelName = entity.modelName;
var entitySetName = entity.entitySetName;
var entityKeyString = entity.keyString;
var headers = null;
//Delete the entity
System.getModule("com.vmware.library.vcac")
  .deleteVCACEntity(hostId, modelName, entitySetName, entityKeyString, headers);
```

Example: Read a vRealize Automation Entity by Custom Filter

This example script performs the following actions:

1 Defines the model name and the entity set name.

2 Defines the properties by which the entities are filtered.

3 Reads a list of entities.

Table 4-23. Input Variables

Variable	Type
host	vCAC:VcacHost
templateName	String

```
var modelName = 'ManagementModelEntities.svc';
var entitySetName = 'VirtualMachineTemplates';
var headers = null;
//Create properties for prefix entity
var properties = {
   VirtualMachineTemplateName:templateName,
};
//Read a list of entities
var entities = vCACEntityManager
  .readModelEntitiesByCustomFilter(host.id, modelName, entitySetName, properties, headers);
```

Example: Read a vRealize Automation Entity by System Query

This example script performs the following actions:

1 Defines the model name and the entity set name.

2 Defines the system queries by which the entities are filtered and selects the top ten results of all virtual machines, filtered by the machine state and component flag.

3 Reads a list of entities.

Table 4-24. Input Variables

Variable	Type
`host`	`vCAC:VcacHost`

```
var modelName = 'ManagementModelEntities.svc';
var entitySetName = 'VirtualMachines';
var filter = "VirtualMachineState eq 'Off' and IsComponent eq true";
var orderBy = 'VirtualMachineName asc';
var top = 10; {
var skip = 0;,
var headers = null;
var select = null;
var entities = vCACEntityManager
  readModelEntitiesBySystemQuery(host.id, modelName, entitySetName, filter, orderBy, select, top,
skip, headers);
```

Finding vRealize Automation Entities Example Scripts

You can cut, paste, and edit the JavaScript examples to write scripts for finding vRealize Automation entities by using the `vCACCAFEEntitiesFinder` scripting utility object.

For more information about scripting in vRealize Orchestrator, see *Developing with VMware vRealize Orchestrator*.

Example: Find Catalog Resources Filtered by Name

Table 4-25. Input Variables

Variable	Type
`host`	`vCACCAFE:VcacHost`

You can use one the following examples:

- This example script gets all catalog resources for the target host matching the query of *name_of_the_resource* by name and description.

  ```
  var items = vCACCAFEEntitiesFinder.findCatalogResources(host, "name_of_the_resource");
  ```

- This example script performs the following actions:

  a Gets the Consumer Resource service and invokes the `get` method passing as a `Pageable` parameter an instance of the `vCACCAFEPageOdataRequest` object.

b Creates the `vCACCAFEPageOdataRequest` object by providing an `OData` query as a single filter of the `name` attribute matching the *name_of_the_resource* string.

```
var service = host.createCatalogClient().getCatalogConsumerResourceService();

var filter = new Array();
filter[0] = vCACCAFEFilterParam.equal("name", vCACCAFEFilterParam.string("name_of_the_resource"));
var query = vCACCAFEOdataQuery.query().addFilter(filter);

var items = service.getResourcesList(new vCACCAFEPageOdataRequest(query));
```

Example: Find Catalog Resources Filtered by Owner

This example script performs the following actions:

1 Gets the Consumer Resource service and invokes the `get` method passing as a `Pageable` parameter an instance of the `vCACCAFEPageOdataRequest` object.

2 Creates the `vCACCAFEPageOdataRequest` object by providing an `OData` query as a single filter of the `owner/ref` attribute matching the *user@domain.com* string.

   The `owners/ref` attribute is a composition based on the internal structure and fields of the catalog resources. The `vCACCAFECatalogResource` entity has the `owners` attribute, which is a collection of `vCACCAFECatalogPrincipal` entities. The `vCACCAFECatalogPrincipal` entity has the `ref` property, which is a string representation of the principal id of the user.

```
var filter = new Array();
filter[0] = vCACCAFEFilterParam.substringOf("owners/ref",
vCACCAFEFilterParam.string("user@domain.com"));
var query = vCACCAFEOdataQuery.query().addFilter(filter);

var items = service.getResourcesList(new vCACCAFEPageOdataRequest(query));
```

Example: Find Catalog Resources Filtered by Name and Owner

This example script combines the `OData` queries from the previous two examples into a single one condition by using the `vCACCAFEFilterParam.and(array of conditions)` logic operator.

```
var conditions = new Array();
conditions[0] = vCACCAFEFilterParam.equal("name",
vCACCAFEFilterParam.string("name_of_the_resource_here"));
conditions[1] = vCACCAFEFilterParam.substringOf("owners/ref",
vCACCAFEFilterParam.string("user@domain.com"));

var filter = new Array();
filter[0] = vCACCAFEFilterParam.and(conditions);
var query = vCACCAFEOdataQuery.query().addFilter(filter);

var items = service.getResourcesList(new vCACCAFEPageOdataRequest(query));
```

You can define other conditions by using different logic operators such as `vCACCAFEFilterParam.group(array of parameters)`, `vCACCAFEFilterParam.not(parameter)`, `vCACCAFEFilterParam.startsWith(id, string)`, `vCACCAFEFilterParam.endsWith(id, string)`, `vCACCAFEFilterParam.greaterThan(id, number)`, `vCACCAFEFilterParam.lessThan(id, number)`, and so on.

Get a Resource Provisioned by vRealize Automation Example Script

You can cut, paste, and edit the JavaScript example to write scripts for retrieving the actual entities of vRealize Automation provisioned resources.

The `CatalogResource` type represents the provisioned resources in vRealize Automation. This type has an attribute of `ProviderBinding` type which represents the relation between the catalog resource and its provider with the following attributes:

- `bindingId` - represents the identifier of the entity which is unique for the provider
- `providerRef` - identifies the catalog provider which corresponds directly to a service registered in thevRealize Automation component registry

For more information about scripting in vRealize Orchestrator, see *Developing with VMware vRealize Orchestrator*.

Example: Get a Virtual Machine Provisioned as a vRealize Automation Catalog Resource

This example uses a vRealize Automation host and its IaaS host as input parameters and, for a provided resource id, returns the corresponding IaaS virtual machine. The scripting code takes only catalog resources of `Virtual Machine` type that are provisioned by the `iaas-service` provider.

Table 4-26. Input Variables

Variable	Type
vcacHost	vCACCAFE:VCACHost
iaasHost	vCAC:VCACHost

```
// Id of the catalog resource (or vCACCAFECatalogResource_instance.getId())
var resourceId = "c222629c-6f90-4458-8c92-8ece0ba06173";

var resource = vCACCAFEEntitiesFinder.getCatalogResource(vcacHost, resourceId);

var resourceType = resource.getResourceTypeRef().getLabel();
System.log("resource type: " + resourceType);

var providerBinding = resource.getProviderBinding();

var bindingId = providerBinding.getBindingId();
System.log("provider binding id: " + bindingId);

var provider = providerBinding.getProviderRef();
System.log("provider id: " + provider.getId());
System.log("provider name: " + provider.getLabel());
```

```
if ((resourceType == "Virtual Machine") && (provider.getLabel() == "iaas-service")) {
    System.log("It is an IaaS VM!");

    // IaaS virtual machine
    var vm = Server.findForType("vCAC:VirtualMachine", bindingId);
    System.log("IaaS VM id: " + vm.virtualMachineID);
    System.log("IaaS VM name: " + vm.displayName);

    // IaaS Entity
    var entity = System.getModule("com.vmware.library.vcac").getVirtualMachineEntityFromId(iaasHost,
bindingId);
    System.log("IaaS entity id: " + entity.keyString);
}
```

Common Tasks Example Scripts

You can cut, paste, and edit the JavaScript examples, or use them as samples to help you learn to develop your own scripts for common vRealize Automation tasks.

For more information about scripting in vRealize Orchestrator, see *Developing with VMware vRealize Orchestrator*.

Example: Create a vRealize Automation Advanced Service Blueprint

This example script performs the following actions:

1 Sets the vRealize Orchestrator workflow used to build the service blueprint.

2 Generates the content for the service blueprint based on the workflow.

3 Creates the service blueprint entity.

4 Publishes the service blueprint.

Table 4-27. Input Variables

Variable	**Type**
`host`	`vCACCAFE:VCACHost`

```
//ID of the workflow used to create the service blueprint
var workflowId = "44e42047-2fa0-4e4a-ba0c-12086540b28b";

var name = "MyBlueprint"
var description = "Blueprint description";
var workflowClient = host.createAdvancedDesignerClient().getAdvancedDesignerWorkflowService();

//Generate a service blueprint based on the workflow ID
var blueprint = workflowClient.generateServiceBlueprintByWorkflowId(workflowId);
blueprint.setTenant(host.tenant);
blueprint.setName(name);
blueprint.setDescription(description);


//Create the service blueprint
```

```
var blueprintService =
host.createAdvancedDesignerClient().getAdvancedDesignerServiceBlueprintService();
var uri = blueprintService.createServiceBlueprint(host.tenant , blueprint);

//Publish the service blueprint
var createdBlueprint = blueprintService.getServiceBlueprintByUri(uri);
blueprintService.updateServiceBlueprintStatus(host.tenant, createdBlueprint.getId(),
vCACCAFEDesignerPublishStatus.PUBLISHED);
```

Example: Create a vRealize Automation Approval Policy

This example script performs the following actions:

1 Gets the approval policy type.
2 Sets the user and group whose approval is required.
3 Sets the approval levels.
4 Defines the pre-provisioning approval phase.
5 Defines the post-provisioning approval phase.
6 Defines the approval policy specifications such as name, description, and type.
7 Creates the approval policy.
8 Publishes the approval policy. Once an approval policy is published, it becomes read-only.

Table 4-28. Input Variables

Variable	Type
host	vCACCAFE:VCACHost

```
// Get the type of approval policy by ID
var typeService = host.createApprovalClient().getApprovalApprovalPolicyTypeService();
var type = typeService.getApprovalPolicyType("com.vmware.cafe.catalog.request");

// Set the user and group required to complete the approval
var user = new vCACCAFEApprovalPrincipal();
user.setValue("user@domain.com");
user.setType(vCACCAFEApprovalPrincipalType.USER);

var group = new vCACCAFEApprovalPrincipal();
group.setValue("group@domain.com");
group.setType(vCACCAFEApprovalPrincipalType.GROUP);

// Set the level of the approval
var level = new vCACCAFEApprovalLevel();
level.setName("IT Approval Level");
level.setDescription("IT Approval Level description");
level.setApprovalMode(vCACCAFEApprovalMode.ALL);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(level, "getApprovers", user);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(level, "getApprovers", group);
level.setLevelNumber(1);
```

```
// Set pre-provisioning phase type and the phase of the approval
var phase1Type = new vCACCAFEApprovalPhaseType();
phase1Type.setId("com.vmware.cafe.catalog.request.pre");
phase1Type.setName("Pre-Provisioning type");
phase1Type.setDescription("Pre-Provisioning type description");
phase1Type.setPhaseOrder(1);

var phase1 = new vCACCAFEPhase();
phase1.setName("Pre-Provisioning");
phase1.setDescription("Pre provisioning phase");
phase1.setPhasetype(phase1Type);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(phase1, "getLevels", level);

// Set post-provisioning phase type and the phase of the approval
var phase2Type = new vCACCAFEApprovalPhaseType();
phase2Type.setId("com.vmware.cafe.catalog.request.post");
phase2Type.setName("Post-Provisioning type");
phase2Type.setDescription("Post-Provisioning type description");
phase2Type.setPhaseOrder(1);

var phase2 = new vCACCAFEPhase();
phase2.setName("Post-Provisioning");
phase2.setDescription("Post provisioning phase");
phase2.setPhasetype(phase2Type);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(phase2, "getLevels", level);

// Create the approval policy specifications
var spec = new vCACCAFEApprovalPolicy();
spec.setName("New Policy");
spec.setDescription("New Policy description");
spec.setPolicyType(type);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(spec, "getPhases", phase1);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(spec, "getPhases", phase2);

// Create the approval policy
var approvalPolicyService = host.createApprovalClient().getApprovalApprovalPolicyService();
var approvalPolicy = approvalPolicyService.createPolicy(spec);

// Publish the approval policy
approvalPolicy.setState(vCACCAFEApprovalPolicyState.PUBLISHED);
approvalPolicy = approvalPolicyService.update(approvalPolicy);
System.log("New approval policy id: " + approvalPolicy.getId());
```

Using the Configuration Plug-In 5

In addition to configuring Orchestrator by using Control Center, you can modify the Orchestrator server configuration settings by running workflows from the Configuration plug-in.

With the Configuration plug-in, you can configure and manage the Orchestrator server keystores and trusted certificates.

This chapter includes the following topics:

- Access the Configuration Plug-In Workflow Library
- Configuration Plug-In Workflow Library

Access the Configuration Plug-In Workflow Library

You must use the Orchestrator client to access the elements from the Configuration plug-in workflow library.

Procedure

1. From the drop-down menu in the Orchestrator client, select **Run** or **Design**.
2. Click the **Workflows** view.
3. Expand the hierarchical list to **Library > Configuration**.

What to do next

Review the workflow library.

Configuration Plug-In Workflow Library

The Configuration plug-in workflow library contains workflows that you can use to run automated processes related to the configuration of vRealize Orchestrator.

SSL Trust Manager Workflows

The SSL Trust Manager category contains workflows that you can use for deleting and importing SSL certificates.

You access these workflows from **Library > Configuration > SSL Trust Manager workflows** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Delete a trusted certificate	Deletes an SSL certificate from the server trust store.
Import certificate from URL	Imports an SSL certificate from a URL into the server trust store.
Import a certificate from URL using authenticated proxy server	Imports an SSL certificate from a URL that is reachable through an authenticated proxy server.
Import certificate from URL using proxy server	Imports an SSL certificate from a URL that is reachable through a proxy server.
Import certificate from URL with certificate alias	Imports an SSL certificate from a URL into the server trust store.
Import trusted certificate from a file	Imports an SSL certificate from a file into the server trust store.

Keystore Workflows

You access the Keystore configuration workflows from **Library > Configuration > Keystores** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add certificate	Adds a certificate to a keystore.
Add key	Adds a key.
Create a keystore	Creates a new keystore.
Delete a keystore	Deletes a keystore.
Delete certificate	Deletes a certificate from a keystore.
Delete entry	Deletes an entry.
Delete key	Deletes a key.

Using the Library Plug-In

6

You can use the Library plug-in workflows as templates for customization and automation of client processes, and to troubleshoot Orchestrator.

This chapter includes the following topics:

Library Plug-In Workflows

The Library plug-in provides workflows in the **Locking**, **Orchestrator**, and **Troubleshooting** workflow categories.

Locking Workflows

You access these workflows from **Library > Locking** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Display all locks	Shows all locks.
Locking test	A test workflow that creates a lock.
Locking test (x5)	A test workflow that creates five locks.
Release all locks	Releases all locks.

Orchestrator Task Workflows

You access these workflows from **Library > Orchestrator > Tasks** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Create recurrent task	Creates a recurrent task and returns the newly created task.
Create task	Schedules a workflow to run at a later time and date, as a task.

Orchestrator Workflows

You access these workflows from **Library > Orchestrator > Workflows** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Refresh stale workflow runs in waiting state	Processes all workflow runs that are in waiting state for the specified remote server and updates the workflow state according to the remote workflow run. You can use this workflow if there is data loss between the workflow runs, for example, when there is loss of connectivity between the Orchestrator servers.
Start workflows in a series	Runs a workflow multiple times in a series, one instance after the other. You provide workflow parameters in an array. You also provide a property list, with one property per workflow input, for each instance of the workflow that starts. The number of properties in the array define the number of workflow runs.
Start workflows in parallel	Runs a workflow multiple times, with different parameters. You provide workflow parameters in an array. You also provide a property list, with one property per workflow input, for each instance of the workflow that starts. The number of properties in the array define the number of workflow runs.

Tagging Workflows

You access these workflows from **Library > Tagging** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Find objects by tag	Finds objects by the tags assigned to them. You provide the names and values of the tags and the workflow returns a list of the objects to which these tags apply.
List workflow tags	Lists the tags assigned to the workflow you specified as an input parameter.
Tagging example	Demonstrates workflow tagging.
Tag workflow	Assigns a tag to a workflow. You must specify the workflow you want to tag and the tag name and value.
Untag workflow	Removes a tag from a workflow. You must specify the workflow you want to untag and the tag you want to remove from the specified workflow.

Using the SQL Plug-In

7

You can use the API that the SQL plug-in provides to implement connectivity to SQL databases and other tabular data sources, such as spreadsheets or flat files.

The SQL plug-in API which is based on JDBC, provides a call-level API for SQL-based database access. The SQL plug-in also provides sample workflows that demonstrate how to use the API in workflows.

This chapter includes the following topics:

Configuring the SQL Plug-In

You can use the workflows included in the SQL plug-in and run them from the Orchestrator client to configure the SQL plug-in and to add, update, or remove a database.

SQL Plug-In Configuration Workflows

The Configuration workflow category of the SQL plug-in contains workflows that allow you to manage databases and database tables.

You can access these workflows from **Library > SQL > Configuration** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add a database	Adds a database object to the Database plug-in inventory.
Add tables to a database	Adds database tables to a database in the Database plug-in inventory.
Remove a database	Removes a database object from the Database plug-in inventory.
Remove a table from a database	Removes a database table from a database in the Database plug-in inventory.
Update a database	Updates the configuration of a database object in the Database plug-in inventory.
Validate a database	Validates a database in the Database plug-in inventory.

Add a Database

You can run a workflow to add a database to the Orchestrator server and configure the host connection parameters.

When you add a database that requires a secure connection, you must import the database SSL certificate. You can import the SSL certificate under the **Trusted Certificates** tab in Control Center.

Procedure

1. Log in to the Orchestrator client as an administrator.
2. Click the **Workflows** view in the Orchestrator client.
3. In the workflows hierarchical list, expand **Library > SQL > Configuration** and navigate to the **Add a database** workflow.
4. Right-click the **Add a database** workflow and select **Start workflow**.
5. In the **Name** text box, type the name of the database.
6. Select the type of the database.
7. In the **Connection URL** text box, type the address of the database.

Database Type	Syntax
Oracle	`jdbc:oracle:thin:@`*`database_url:port_number`*`:`*`SID`*
Microsoft SQL (with SQL authentication)	`jdbc:jtds:sqlserver://`*`database_url:port_number/database_name`*
Microsoft SQL (with Windows account authentication)	`jdbc:jtds:sqlserver://`*`database_url:port_number/database_name`*`;useNTLMv2=`*`true`*`;domain=`*`domain_name`*
PostgreSQL	`jdbc:postgresql://`*`database_url:port_number/database_name`*
MySQL	`jdbc:mysql://`*`database_url:port_number/database_name`*

8. Select the session mode that the plug-in uses to connect to the database.

Option	Description
Shared Session	The plug-in uses shared credentials to connect to the database. You must provide the database credentials for the shared session.
Session Per User	The Orchestrator client retrieves credentials from the user who is logged in. **Note** To use session per user mode, you must authenticate by using a user name only. You should not use *domain\user* or *user@domain* for authentication.

9. Click **Submit** to run the workflow.

Results

After the workflow runs successfully, the database and all tables that belong to it appear in the **Inventory** view.

Add Tables to a Database

You can run a workflow to add tables to a database that is in the Database plug-in inventory.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a database from the **Inventory** view.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > SQL > Configuration** and navigate to the **Add tables to a database** workflow.
3. Right-click the **Add tables to a database** workflow and select **Start workflow**.
4. Select a database to which to add tables.
5. Select the tables that you want to add.
6. Click **Submit** to run the workflow.

Results

After the workflow runs successfully, the added database tables appear in the **Inventory** view of the Orchestrator client.

Update a Database

You can run a workflow to update the configuration of a database that is in the plug-in inventory.

Procedure

1. Log in to the Orchestrator client as an administrator.
2. Click the **Workflows** view in the Orchestrator client.
3. In the **Workflows** hierarchical list, expand **Library > SQL > Configuration** and navigate to the **Update a database** workflow.
4. Right-click the **Update a database** workflow and select **Start workflow**.
5. Select a database that you want to update.
6. In the **Name** text box, type the new name of the database.

   The database appears in the **Inventory** view with the name that you specify.
7. Select the type of the database.

8 In the **Connection URL** text box, type the new address of the database.

9 Select the session mode that the plug-in uses to connect to the database.

Option	Description
Shared Session	The plug-in uses shared credentials to connect to the database. You must provide the database credentials for the shared session.
Session Per User	The Orchestrator client retrieves credentials from the user who is logged in. **Note** To use session per user mode, you must authenticate by using a user name only. You should not use *domain\user* or *user@domain* for authentication.

10 Click **Submit** to run the workflow.

Running the SQL Sample Workflows

You can run the SQL plug-in workflows to perform JDBC operations such as generating a JDBC URL, testing a JDBC connection, and managing rows in JDBC tables. You can also run the SQL plug-in workflows to manage databases and database tables, as well as to run SQL operations.

Generate a JDBC URL

You can run a workflow from the Orchestrator client to generate a JDBC connection URL.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run JDBC workflows.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > JDBC** to navigate to the JDBC URL generator workflow.

3 Right-click the JDBC URL generator workflow and select **Start workflow**.

4 Select the type of database for which to generate a URL.

   Note If you use a Microsoft database, you might have to click **Next** and to provide the database instance name and database user domain name.

5 Provide the required information to generate a database URL.

   a Type a database server name or IP address.

   b Type a database name.

   c (Optional) Type a database port number.

   If you do not specify a port number, the workflow uses a default port number.

   d Type a user name to access the database.
   e Type a password to access the database.

6 Click **Submit** to run the workflow.

Test a JDBC Connection

You can run a workflow from the Orchestrator client to test the connection to a database.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run JDBC workflows.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > JDBC > JDBC Examples** to navigate to the JDBC connection example workflow.

3 Right-click the JDBC connection example workflow and select **Start workflow**.

4 Provide the required information to test a database connection.

   a Type a user name to access the database.
   b Type the URL to test.
   c Type a password to access the database.

5 Click **Submit** to run the workflow.

Create a Table by Using JDBC

You can run a workflow from the Orchestrator client to create a database.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run JDBC workflows.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > JDBC > JDBC Examples** to navigate to the JDBC create table example workflow.

3 Right-click the JDBC create table example workflow and select **Start workflow**.

4 Provide the required information, and click **Next**.

   a Type a password to access the database.

   b Type a database connection URL.

   c Type a user name to access the database.

5 Type an SQL create statement.

   An example syntax is:

```
CREATE TABLE "table_name"
("column1" "data_type_for_column1",
"column2" "data_type_for_column2")
```

6 Click **Submit** to run the workflow.

Insert a Row into a JDBC Table

You can run a workflow from the Orchestrator client to test the insertion of a row into a JDBC table.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run JDBC workflows.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > JDBC > JDBC Examples** to navigate to the JDBC insert into table example workflow.

3 Right-click the JDBC insert into table example workflow and select **Start workflow**.

4 Provide the required information, and click **Next**.

   a Type a database connection URL.

   b Type a user name to access the database.

   c Type a password to access the database.

5 Type an SQL insert statement, and click **Next**.

   An example syntax is:

```
INSERT INTO "table_name" ("column1", "column2")
VALUES ("value1", "value2")
```

6 Type the values to insert into the row.

7 Click **Submit** to run the workflow.

Select Rows from a JDBC Table

You can run a workflow from the Orchestrator client to select rows from a JDBC table.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run JDBC workflows.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > JDBC > JDBC Examples** to navigate to the JDBC select from table example workflow.
3. Right-click the JDBC select from table example workflow and select **Start workflow**.
4. Provide the required information, and click **Next**.
   a. Type a database connection URL.
   b. Type a user name to access the database.
   c. Type a password to access the database.
5. Type an SQL select statement.

   An example syntax is:

   ```
   SELECT * FROM "table_name"
   ```
6. Click **Submit** to run the workflow.

Delete an Entry from a JDBC Table

You can run a workflow from the Orchestrator client to test the deletion of an entry from a JDBC table.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run JDBC workflows.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > JDBC > JDBC Examples** to navigate to the JDBC delete entry from table example workflow.
3. Right-click the JDBC delete entry from table example workflow and select **Start workflow**.

4 Provide the required information, and click **Next**.

   a Type the first name of the user entry to be deleted.

   b Type a user name to access the database.

   c Type a JDBC connection URL.

   d Type the last name of the user entry to be deleted.

   e Type a password to access the database.

5 Type an SQL delete statement.

   An example syntax is:

```
DELETE FROM "table_name" where ("column1" = ?, "column2" = ?)
```

6 Click **Submit** to run the workflow.

Delete All Entries from a JDBC Table

You can run a workflow from the Orchestrator client to delete all entries from a JDBC table.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run JDBC workflows.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > JDBC > JDBC Examples** to navigate to the JDBC delete all from table example workflow.

3 Right-click the JDBC delete all from table example workflow and select **Start workflow**.

4 Provide the required information, and click **Next**.

   a Type a database connection URL.

   b Type a user name to access the database.

   c Type a password to access the database.

5 Type an SQL delete statement.

   An example syntax is:

```
DELETE FROM "table_name"
```

6 Click **Submit** to run the workflow.

Drop a JDBC Table

You can run a workflow from the Orchestrator client to test the dropping of a JDBC table.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run JDBC workflows.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > JDBC > JDBC Examples** to navigate to the JDBC drop table example workflow.
3. Right-click the JDBC drop table example workflow and select **Start workflow**.
4. Provide the required information, and click **Next**.
   a. Type a password to access the database.
   b. Type a database connection URL.
   c. Type a user name to access the database.
5. Type an SQL drop statement.

   An example syntax is:

   ```
   DROP TABLE "table_name"
   ```
6. Click **Submit** to run the workflow.

Run a Complete JDBC Cycle

You can run a workflow from the Orchestrator client to test all JDBC example workflows in one full cycle.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run JDBC workflows.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > JDBC > JDBC Examples** to navigate to the Full JDBC cycle example workflow.
3. Right-click the Full JDBC cycle example workflow and select **Start workflow**.
4. Provide the required information, and click **Next**.
   a. Type a database connection URL.
   b. Type a user name to access the database.
   c. Type a password to access the database.
5. Type the values to be used as entries in the database.

6 Click **Submit** to run the workflow.

Using the SQL Plug-In Standard Workflows

You can use the SQL workflows to run SQL operations.

SQL Plug-In Workflow Library

You can run the SQL plug-in workflows to manage databases and database tables and to run SQL operations.

You can access the database configuration workflows from **Library > SQL > Configuration** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add a database	Adds a database object to the plug-in inventory.
Add tables to a database	Adds database tables to a database in the plug-in inventory.
Remove a database	Removes a database object from the plug-in inventory.
Remove a table from a database	Removes a database table from a database in the plug-in inventory.
Update a database	Updates the configuration of a database object in the plug-in inventory.
Validate a database	Validates a database in the plug-in inventory.

You can access the SQL operations workflows from **Library > SQL** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Execute a custom query on a database	Runs a custom query on a specified database and returns the number of affected rows. You can run the workflow to update, delete, insert, and write queries.
Generate CRUD workflows for a table	Generates Create, Read, Update, and Delete workflows for a particular table.
Read a custom query from a database	Runs a custom query on a specified database and returns the result in an array of properties. You can run the workflow to select and read queries.

Generate CRUD Workflows for a Table

You can run a workflow to generate Create, Read, Update, and Delete workflows for a particular table.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a database from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > SQL** and navigate to the **Generate CRUD workflows for a table** workflow.

3 Right-click the **Generate CRUD workflows for a table** workflow and select **Start workflow**.

4 Select a table for which to generate the workflows.

5 Select the workflow folder in which to generate the workflows.

6 Select whether to overwrite any existing workflows.

Option	Description
Yes	The generated workflows overwrite existing workflows with the same name.
No	New workflows are not generated if workflows with the same name exist in the folder.

7 (Optional) Select columns that should not be populated.

You cannot edit the selected columns with the generated CRUD workflows.

8 Click **Submit** to run the workflow.

Results

After the workflow runs successfully, the CRUD workflows appear in the selected workflow folder.

What to do next

You can run the generated workflows on the selected database table.

Using the SSH Plug-In 8

You can use the SSH plug-in workflows to run SSH commands on a remote host that supports SSH and transfer files between an Orchestrator server and a remote host through a secure connection.

This chapter includes the following topics:

Configure the SSH Plug-In

You can set up the SSH plug-in to ensure encrypted connections.

Procedure

1. Log in to the Orchestrator client as an administrator.
2. Click the **Workflows** view in the Orchestrator client.
3. In the workflows hierarchical list, expand **Library > SSH** and navigate to the Add SSH host workflow.
4. Right-click the Configure mail workflow and select **Start workflow**.
5. In the **Host name** text box, enter the name of the host that you want to access with SSH through Orchestrator.
6. Enter the target port. The default SSH port is 22.

   The host is added to the list of SSH connections.
7. (Optional) Configure an entry path on the server.

   a. Click **New root folder**.

   b. Enter the new path and click **Insert value**.
8. Enter the user name for a user who has the necessary permissions to run SSH commands.

9 Select the authentication type.

Option	Action
Yes	Enter a password to use password authentication.
No	Enter the path to the private key and the private key passphrase to use key authentication.

10 Click **Submit** to run the workflow.

Results

The SSH host is available in the **Inventory** view of the Orchestrator client.

Configuration Workflows

The Configuration category of the SSH plug-in contains workflows that let you manage the connections between Orchestrator and SSH hosts.

You can access these workflows from **Library > SSH > Configuration** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add a Root Folder to SSH Host	Adds a root folder to an existing connection to an SSH host.
Add SSH Host	Adds a new connection to an SSH host to the existing configuration.
Remove a Root Folder from SSH Host	Removes a root folder from an existing connection to an SSH host.
Remove SSH Host	Removes an existing connection to an SSH host from the existing configuration.
Update SSH Host	Updates an existing connection to an SSH host.

Running the SSH Plug-In Sample Workflows

You can run the SSH plug-in sample workflows from the Orchestrator client to test the connection between the Orchestrator server and the SSH host.

- Generate a Key Pair

  You can run a workflow from the Orchestrator client to generate a key pair. You can use the key pair to connect to an SSH host without a password.

- Change the Key Pair Passphrase

  You can run a workflow from the Orchestrator client to change the passphrase for the key pair that you generated most recently.

- Register an Orchestrator Public Key on an SSH Host

  You can use a public key instead of a password. To register an Orchestrator public key on an SSH host, you can run a workflow from the Orchestrator client.

- Run an SSH Command

  You can run a workflow from the Orchestrator client to run SSH commands on a remote SSH server.
- Copy a File from an SSH Host

  You can run a workflow on the Orchestrator client to copy files from an SSH host to the Orchestrator server.
- Copy a File to an SSH Host

  You can run a workflow from the Orchestrator client to copy files from the Orchestrator server to an SSH host.

Generate a Key Pair

You can run a workflow from the Orchestrator client to generate a key pair. You can use the key pair to connect to an SSH host without a password.

A key pair consists of a public key and a private key. Orchestrator can use the private key to connect to the public key on an SSH host. You can use a passphrase to improve security.

Caution All Orchestrator users with the right set of privileges can read, use, and overwrite your private key.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run SSH workflows.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > SSH** and navigate to the Generate key pair workflow.
3. Right-click the Generate key pair workflow and select **Start workflow**.
4. Provide the required information.
   a. Select the key type.
   b. Select the key size.
   c. (Optional) Enter a passphrase.

      Note You can change the passphrase later.
   d. (Optional) Enter a comment.
5. Click **Submit** to run the workflow.

   If a key pair exists, the new key pair overwrites it.

Change the Key Pair Passphrase

You can run a workflow from the Orchestrator client to change the passphrase for the key pair that you generated most recently.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run SSH workflows.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > SSH** and navigate to the Change key pair passphrase workflow.
3. Right-click the Change key pair passphrase workflow and select **Start workflow**.
4. Reset the key pair passphrase.
   a. Enter the current passphrase.
   b. Enter the new passphrase.
5. Click **Submit** to run the workflow.

Register an Orchestrator Public Key on an SSH Host

You can use a public key instead of a password. To register an Orchestrator public key on an SSH host, you can run a workflow from the Orchestrator client.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run SSH workflows.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > SSH** and navigate to the Register vCO public key on host workflow.
3. Right-click the Register vCO public key on host workflow and select **Start workflow**.
4. Provide the name of the SSH host, and the user name and password to log in to this host.

   Note You must provide credentials that are registered on the SSH host.
5. Click **Submit** to run the workflow.

Results

You can use public key authentication instead of password authentication when you connect to the SSH host as the registered user.

Run an SSH Command

You can run a workflow from the Orchestrator client to run SSH commands on a remote SSH server.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run SSH workflows.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > SSH** and navigate to the Run SSH command workflow.

3 Right-click the Run SSH command workflow and select **Start workflow**.

4 Enter an SSH host name or IP address, and click **Next**.

5 Enter an SSH command to run, and click **Next**.

   Note The default SSH command is `uptime`. It shows how long the server has been active and the user load for that period.

6 Select **Yes** to use password authentication, and click **Next**.

   Note The default option is to use key file authentication.

7 Enter a user name, and click **Next**.

8 Enter a password if the authentication method requires a password. Otherwise, enter the path to the private key and enter the passphrase for the private key.

9 Click **Submit** to run the workflow.

Copy a File from an SSH Host

You can run a workflow on the Orchestrator client to copy files from an SSH host to the Orchestrator server.

The SSH plug-in uses the Java JCraft library, which implements SFTP. The SCP get command workflow transfers files by using SFTP.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run SSH workflows.

Note Orchestrator must have explicit write permissions in order to write in folders.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > SSH** and navigate to the SCP get command workflow.

3 Right-click the SCP get command workflow and select **Start workflow**.

4 Provide the required information, and click **Next**.

   a Type an SSH host name or IP address.

   b Type the SSH authentication information.

5 Type the file information.

   a Type the path to the directory on the Orchestrator server into which to copy the file.

   b Type the path to the file to get from the remote SSH host.

6 Click **Submit** to run the workflow.

Copy a File to an SSH Host

You can run a workflow from the Orchestrator client to copy files from the Orchestrator server to an SSH host.

The SSH plug-in uses the Java JCraft library, which implements SFTP. The SCP put command workflow transfers files by using SFTP.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run SSH workflows.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > SSH** and navigate to the SCP put command workflow.

3 Right-click the SCP put command workflow and select **Start workflow**.

4 Provide the required information, and click **Next**.

   a Type an SSH host name or IP address.

   b Type the SSH authentication information.

5 Enter the file information.

   a Enter the path to the file that you want to copy from the local Orchestrator server to the remote SSH host.

   b Enter the path to the directory on the remote SSH host into which to copy the file.

6 Click **Submit** to run the workflow.

Using the XML Plug-In

9

You can use the XML plug-in to run workflows that create and modify XML documents.

The XML plug-in adds an implementation of a Document Object Model (DOM) XML parser to the Orchestrator JavaScript API. The XML plug-in also provides some sample workflows to demonstrate how you can create and modify XML documents from workflows.

Alternatively, you can use the ECMAScript for XML (E4X) implementation in the Orchestrator JavaScript API to process XML documents directly in JavaScript. For an E4X scripting example, see *Developing with VMware vRealize Orchestrator*.

For information about E4X, go to the Web site of the organization that maintains the ECMA-357 standard.

This chapter includes the following topics:

Running the XML Plug-In Sample Workflows

You can run the XML plug-in sample workflows from the Orchestrator client to create and modify XML documents for testing purposes.

Because the workflows can create, read, or modify files, you must have sufficient access rights to the working directory.

Orchestrator has read, write, and execute rights to a folder named `orchestrator`, at the root of the server system. Although workflows have permission to read, write, and execute in this folder, you must create the folder on the server system. If you use the Orchestrator Appliance, the folder is named `vco` and is located at `/var/run/vco`.

You can allow access to other folders by changing the settings for server file system access from workflows and JavaScript. See *Installing and Configuring VMware vRealize Orchestrator, Setting Server File System Access from Workflows and Actions*.

- Create a Simple XML Document

  You can run a workflow from the Orchestrator client to create a simple XML document for testing purposes.

- Find an Element in an XML Document

  You can run a workflow from the Orchestrator client to find an element in the XML created by the Create a simple XML document workflow.

- Modify an XML Document

  You can run a workflow from the Orchestrator client to modify the XML that the Create a simple XML document workflow creates.

- Create an Example Address Book from XML

  You can run a workflow from the Orchestrator client to create an address book for testing purposes.

Create a Simple XML Document

You can run a workflow from the Orchestrator client to create a simple XML document for testing purposes.

Prerequisites

- Verify that the user account you are logged in with has the necessary permissions to run XML workflows.
- Verify that you created the `c:/orchestrator` folder at the root of the Orchestrator server system or set access rights to another folder.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, open **Library > XML > Samples XML (Simple)** to navigate to the Create a simple XML document workflow.

3 Right-click the Create a simple XML document workflow and select **Start workflow**.

4 Type the filepath to the XML document to create.

  For example, `c:/orchestrator/`*`filename`*`.xml`.

5 Click **Submit** to run the workflow.

Results

The workflow creates an XML document that contains a list of users. The attributes for each entry are `user ID` and `name`.

Find an Element in an XML Document

You can run a workflow from the Orchestrator client to find an element in the XML created by the Create a simple XML document workflow.

Prerequisites

- Verify that the user account you are logged in with has the necessary permissions to run XML workflows.
- Verify that you created the `c:/orchestrator` folder at the root of the Orchestrator server system or set access rights to another folder.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, open **Library > XML > Samples XML (Simple)** to navigate to the Find element in document workflow.
3. Right-click the Find element in document workflow and select **Start workflow**.
4. Type the filepath to the XML document.

   For example, `c:/orchestrator/`*`filename`*`.xml`.
5. Click **Submit** to run the workflow.

   The workflow searches for an element and displays the result in the system log.

What to do next

To view the result, select the completed workflow run in the Orchestrator client and click **Logs** on the **Schema** tab.

Modify an XML Document

You can run a workflow from the Orchestrator client to modify the XML that the Create a simple XML document workflow creates.

Prerequisites

- Verify that the user account you are logged in with has the necessary permissions to run XML workflows.
- Verify that you created the `c:/orchestrator` folder at the root of the Orchestrator server system or set access rights to another folder.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, open **Library > XML > Samples XML (Simple)** to navigate to the Modify XML document workflow.
3. Right-click the Modify XML document workflow and select **Start workflow**.

4 Provide the input and output filepaths.

   a Type the filepath to the XML document to modify.

      For example, `c:/orchestrator/`*`filename`*`.xml`.

   b Type the filepath to the modified XML document.

      For example, `c:/orchestrator/`*`filename`*`.xml`.

   Note If you type the same filepath in both fields, the workflow overwrites the original file with the modified file. If you type an output filepath to a file that does not exist, the workflow creates a modified file.

5 Click **Submit** to run the workflow.

Results

The workflow searches for an element and modifies the entry where the element is found.

Create an Example Address Book from XML

You can run a workflow from the Orchestrator client to create an address book for testing purposes.

Prerequisites

- Verify that the user account you are logged in with has the necessary permissions to run XML workflows.
- Verify that you created the `c:/orchestrator` folder at the root of the Orchestrator server system or set access rights to another folder.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, open **Library > XML > Samples XML (Address Book)** to navigate to the Full address book test workflow.

3 Right-click the Full address book test workflow and select **Start workflow**.

4 Type the path to the address book folder.

   For example, `c:/orchestrator/`*`foldername`*.

   The workflow automatically creates the folder if it does not exist.

5 Click **Submit** to run the workflow.

Results

The workflow creates a DTD, an XML, and a CSS file, appends the stylesheet, and stores the files in the specified folder.

Using the Mail Plug-In

10

You can send email messages from workflows by using the Mail plug-in, which uses the Simple Mail Transfer Protocol (SMTP). For example, you can create a workflow to send an email to a given address if the workflow requires user interaction or when it completes its run.

This chapter includes the following topics:

Define the Default SMTP Connection

The Mail plug-in is installed together with the Orchestrator server and is used for sending and receiving email notifications. You can set the default email account that can authenticate against an SMTP server to send and receive email notifications.

Note Avoid load balancers when configuring mail in Orchestrator. You might receive `SMTP_HOST_UNREACHABLE` error.

Procedure

1. Log in to the Orchestrator client as an administrator.
2. Click the **Workflows** view in the Orchestrator client.
3. In the workflows hierarchical list, expand **Library > Mail** and navigate to the Configure mail workflow.
4. Right-click the Configure mail workflow and select **Start workflow**.
5. Enter the required information.

Text Box	Description
SMTP host	Enter the IP address or domain name of your SMTP server.
SMTP port	Enter a port number to match your SMTP configuration. The default SMTP port is 25.

Text Box	Description
User name	Enter a valid email account. This is the email account that Orchestrator uses to send emails.
Password	Enter the password associated with the user name.
From name and address	Enter the sender information to appear in all emails sent by Orchestrator.

6 Click **Submit** to run the workflow.

Using the Mail Plug-In Sample Workflows

You can call the sample workflows of the Mail plug-in from custom workflows to implement the email functionality to the custom workflows. You can run an example workflow to test the interaction between Orchestrator and your SMTP server.

- Access the Mail Plug-In Sample Workflows

  You can access the Mail plug-in sample workflows through the Orchestrator client.
- Mail Plug-In Sample Workflows

  You can enhance your custom workflows by integrating the sample Mail plug-in workflows.

Access the Mail Plug-In Sample Workflows

You can access the Mail plug-in sample workflows through the Orchestrator client.

Prerequisites

Verify that the user account you are logged in with has the necessary permissions to run Mail workflows.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 Expand the hierarchical list to **Library > Mail**.

What to do next

Review and run the sample workflows.

Mail Plug-In Sample Workflows

You can enhance your custom workflows by integrating the sample Mail plug-in workflows.

You can access the Mail workflows from **Library > Mail** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Configure mail	Defines the connection to the SMTP server, the SMTP authentication account, and the address and display name of the sender.
Retrieve messages	Retrieves the messages of a given email account by using the POP3 protocol.
Retrieve messages (via MailClient)	Retrieves the messages of a certain email account, without deleting them, by using the new scripting API provided by the `MailClient` class.
Send notification	Sends an email with specified content to a given email address. If optional parameters are not specified, the workflow uses the default values set through the Configure mail workflow.
Send notification to mailing list	Sends an email with specified content to a given email address list, CC list, and BCC list. If optional parameters are not specified, the workflow uses the default values set through the Configure mail workflow.

Using the Net Plug-In

11

You can use the Net plug-in to implement the Telnet, FTP, POP3, and IMAP protocols in workflows. The POP3 and IMAP implementations allow downloading and reading email. In combination with the Mail plug-in, the Net plug-in provides full email sending and receiving capabilities in workflows.

Using the Enumeration Plug-In

12

You can use the Enumeration plug-in to implement common enumerated types in workflows.

This chapter includes the following topics:

Time Zone Codes

You can use the time zone codes as possible values for the `Enums:MSTimeZone` enumeration.

Time Zone Code	Time Zone Name	Description
000	Dateline Standard Time	(GMT-12:00) International Date Line West
001	Samoa Standard Time	(GMT-11:00) Midway Island, Samoa
002	Hawaiian Standard Time	(GMT-10:00) Hawaii
003	Alaskan Standard Time	(GMT-09:00) Alaska
004	Pacific Standard Time	(GMT-08:00) Pacific Time (US and Canada); Tijuana
010	Mountain Standard Time	(GMT-07:00) Mountain Time (US and Canada)
013	Mexico Standard Time 2	(GMT-07:00) Chihuahua, La Paz, Mazatlan
015	U.S. Mountain Standard Time	(GMT-07:00) Arizona
020	Central Standard Time	(GMT-06:00) Central Time (US and Canada)
025	Canada Central Standard Time	(GMT-06:00) Saskatchewan
030	Mexico Standard Time	(GMT-06:00) Guadalajara, Mexico City, Monterrey
033	Central America Standard Time	(GMT-06:00) Central America
035	Eastern Standard Time	(GMT-05:00) Eastern Time (US and Canada)
040	U.S. Eastern Standard Time	(GMT-05:00) Indiana (East)

Time Zone Code	Time Zone Name	Description
045	S.A. Pacific Standard Time	(GMT-05:00) Bogota, Lima, Quito
050	Atlantic Standard Time	(GMT-04:00) Atlantic Time (Canada)
055	S.A. Western Standard Time	(GMT-04:00) Caracas, La Paz
056	Pacific S.A. Standard Time	(GMT-04:00) Santiago
060	Newfoundland and Labrador Standard Time	(GMT-03:30) Newfoundland and Labrador
065	E. South America Standard Time	(GMT-03:00) Brasilia
070	S.A. Eastern Standard Time	(GMT-03:00) Buenos Aires, Georgetown
073	Greenland Standard Time	(GMT-03:00) Greenland
075	Mid-Atlantic Standard Time	(GMT-02:00) Mid-Atlantic
080	Azores Standard Time	(GMT-01:00) Azores
083	Cape Verde Standard Time	(GMT-01:00) Cape Verde Islands
085	GMT Standard Time	(GMT) Greenwich Mean Time : Dublin, Edinburgh, Lisbon, London
090	Greenwich Standard Time	(GMT) Casablanca, Monrovia
095	Central Europe Standard Time	(GMT+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague
100	Central European Standard Time	(GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb
105	Romance Standard Time	(GMT+01:00) Brussels, Copenhagen, Madrid, Paris
110	W. Europe Standard Time	(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
113	W. Central Africa Standard Time	(GMT+01:00) West Central Africa
115	E. Europe Standard Time	(GMT+02:00) Bucharest
120	Egypt Standard Time	(GMT+02:00) Cairo
125	FLE Standard Time	(GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius
130	GTB Standard Time	(GMT+02:00) Athens, Istanbul, Minsk
135	Israel Standard Time	(GMT+02:00) Jerusalem
140	South Africa Standard Time	(GMT+02:00) Harare, Pretoria
145	Russian Standard Time	(GMT+03:00) Moscow, St. Petersburg, Volgograd
150	Arab Standard Time	(GMT+03:00) Kuwait, Riyadh
155	E. Africa Standard Time	(GMT+03:00) Nairobi
158	Arabic Standard Time	(GMT+03:00) Baghdad
160	Iran Standard Time	(GMT+03:30) Tehran

Time Zone Code	Time Zone Name	Description
165	Arabian Standard Time	(GMT+04:00) Abu Dhabi, Muscat
170	Caucasus Standard Time	(GMT+04:00) Baku, Tbilisi, Yerevan
175	Transitional Islamic State of Afghanistan Standard Time	(GMT+04:30) Kabul
180	Ekaterinburg Standard Time	(GMT+05:00) Ekaterinburg
185	West Asia Standard Time	(GMT+05:00) Islamabad, Karachi, Tashkent
190	India Standard Time	(GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi
193	Nepal Standard Time	(GMT+05:45) Kathmandu
195	Central Asia Standard Time	(GMT+06:00) Astana, Dhaka
200	Sri Lanka Standard Time	(GMT+06:00) Sri Jayawardenepura
201	N. Central Asia Standard Time	(GMT+06:00) Almaty, Novosibirsk
203	Myanmar Standard Time	(GMT+06:30) Yangon (Rangoon)
205	S.E. Asia Standard Time	(GMT+07:00) Bangkok, Hanoi, Jakarta
207	North Asia Standard Time	(GMT+07:00) Krasnoyarsk
210	China Standard Time	(GMT+08:00) Beijing, Chongqing, Hong Kong SAR, Urumqi
215	Singapore Standard Time	(GMT+08:00) Kuala Lumpur, Singapore
220	Taipei Standard Time	(GMT+08:00) Taipei
225	W. Australia Standard Time	(GMT+08:00) Perth
227	North Asia East Standard Time	(GMT+08:00) Irkutsk, Ulaan Bataar
230	Korea Standard Time	(GMT+09:00) Seoul
235	Tokyo Standard Time	(GMT+09:00) Osaka, Sapporo, Tokyo
240	Yakutsk Standard Time	(GMT+09:00) Yakutsk
245	A.U.S. Central Standard Time	(GMT+09:30) Darwin
250	Cen. Australia Standard Time	(GMT+09:30) Adelaide
255	A.U.S. Eastern Standard Time	(GMT+10:00) Canberra, Melbourne, Sydney
260	E. Australia Standard Time	(GMT+10:00) Brisbane
265	Tasmania Standard Time	(GMT+10:00) Hobart
270	Vladivostok Standard Time	(GMT+10:00) Vladivostok
275	West Pacific Standard Time	(GMT+10:00) Guam, Port Moresby
280	Central Pacific Standard Time	(GMT+11:00) Magadan, Solomon Islands, New Caledonia
285	Fiji Islands Standard Time	(GMT+12:00) Fiji Islands, Kamchatka, Marshall Islands

Time Zone Code	Time Zone Name	Description
290	New Zealand Standard Time	(GMT+12:00) Auckland, Wellington
300	Tonga Standard Time	(GMT+13:00) Nuku'alofa

Using the Workflow Documentation Plug-In

13

You can use the Workflow Documentation plug-in to generate PDF documentation about a specific workflow or workflow category.

This chapter includes the following topics:

Workflow Library for the Workflow Documentation Plug-In

With the Workflow Documentation plug-in workflows, you can generate PDF documentation about specific workflows or workflow categories.

You can access these workflows from **Library > Workflow documentation** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Get documentation for workflow	Generates information about a workflow that you select.
Get documentation for workflow category	Generates information about a workflow category that you select.

Generate Workflow Documentation

You can export documentation in PDF format about a workflow or a workflow folder that you select at any time.

The exported document contains detailed information about the selected workflow or the workflows in the folder. The information about each workflow includes name, version history of the workflow, attributes, parameter presentation, workflow schema, and workflow actions. In addition, the documentation also provides the source code for the used actions.

Procedure

1. From the drop-down menu in the Orchestrator client, select **Run** or **Design**.
2. Click the **Workflows** view.

3. Navigate to the workflow or workflow folder for which you want to generate documentation and right-click it.
4. Select **Generate documentation**.
5. Browse to locate the folder in which to save the PDF file, provide a file name, and click **Save**.

Results

The PDF file containing the information about the selected workflow, or the workflows in the folder, is saved on your system.

Using the HTTP-REST Plug-In

14

The HTTP-REST plug-in allows you to manage REST Web services by providing interaction between vRealize Orchestrator and REST hosts. You can define REST services and their operations as inventory objects by running configuration workflows, and perform REST operations on the defined objects.

The plug-in contains a set of standard workflows related to managing REST hosts and invoking REST operations. You can also generate custom workflows to automate tasks in a REST environment.

This chapter includes the following topics:

Configuring the HTTP-REST Plug-In

You must use the Orchestrator client to configure the HTTP-REST plug-in.

Configuration Workflows

The Configuration workflow category contains workflows that help you to manage REST hosts.

You can access these workflows from **Library > HTTP-REST > Configuration** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Add a REST host	Adds a REST host to the plug-in inventory.
Add a REST host by Swagger spec as a string	Adds a REST host based on a Swagger spec web resource provided as a string.
Add a REST host by Swagger spec from a URL	Adds a REST host based on a Swagger spec available at a specific URL.
Add a REST operation	Adds an operation to a REST host.
Add schema to a REST host	Adds an XSD schema to a REST host.

Workflow Name	Description
Clone a REST host	Creates a clone of a REST host.
Clone a REST operation	Creates a clone of a REST operation.
Reload plug-in configuration	Refreshes the list of REST hosts in the plug-in inventory.
Remove a REST host	Removes a REST host from the plug-in inventory.
Remove a REST operation	Removes an operation from a REST host.
Remove schemas form a REST host	Removes all associated XSD schemas from a REST host.
Update a REST host	Updates a REST host in the plug-in inventory.
Update a REST operation	Updates an operation on a REST host.

Configure Kerberos Authentication

You can use Kerberos authentication when you add and manage a PowerShell host.

With Kerberos authentication, domain users can run commands on remote PowerShell-enabled machines over WinRM.

Procedure

1. Enable Kerberos authentication on the WinRM service.

   a. Run the following command to check whether Kerberos authentication is allowed.

      ```
      c:\> winrm get winrm/config/service
      ```

   b. Run the following command to enable Kerberos authentication.

      ```
      c:\> winrm set winrm/config/service/auth @{Kerberos="true"}
      ```

2. Enable Kerberos authentication on the WinRM client.

   a. Run the following command to check whether Kerberos authentication is allowed.

      ```
      c:\> winrm get winrm/config/client
      ```

   b. Run the following command to enable Kerberos authentication.

      ```
      c:\> winrm set winrm/config/client/auth @{Kerberos="true"}
      ```

3. Run the following command to test the connection to the WinRM service.

   ```
   c:\> winrm identify -r:http://winrm_server:5985 -auth:Kerberos -u:user_name -
   p:password -encoding:utf-8
   ```

4 Create a `krb5.conf` file and save it to the following location.

Operating System	Path
Windows	`C:\Program Files\Common Files\VMware\VMware vCenter Server - Java Components\lib\security\`
Linux	`/usr/java/jre-vmware/lib/security/` for external vRealize Orchestrator. `/etc/krb5.conf` for vRealize Orchestrator that is built into vRealize Automation.

A `krb5.conf` file has the following structure:

```
[libdefaults]
default_realm = YOURDOMAIN.COM
udp_preference_limit = 1
[realms]
YOURDOMAIN.COM = {
kdc = kdc.yourdomain.com
default_domain = yourdomain.com
}
[domain_realm]
.yourdomain.com=YOURDOMAIN.COM
yourdomain.com=YOURDOMAIN.COM
```

The `krb5.conf` must contain specific configuration parameters with their values.

Kerberos configuration tags	Details
default_realm	The default Kerberos realm that a client uses to authenticate against an Active Directory server. **Note** Must be in uppercase letters.
kdc	The domain controller that acts as a Key Distribution Center (KDC) and issues Kerberos tickets.
default_domain	The default domain that is used to produce a fully qualified domain name. **Note** This tag is used for Kerberos 4 compatibility.

Note By default, the Java Kerberos configuration uses the UDP protocol. To use only the TCP protocol, you must specify the `udp_preference_limit` parameter with a value **1**.

Note The Kerberos authentication requires a Fully Qualified Domain Name (FQDN) host address.

Important When you add or modify the `krb5.conf` file, you must restart the Orchestrator server service.

Add a REST Host

You can run a workflow to add a REST host and configure the host connection parameters.

Procedure

1 Log in to the Orchestrator client as an administrator.
2 Click the **Workflows** view in the Orchestrator client.
3 In the workflows hierarchical list, expand **Library > HTTP-REST > Configuration** and navigate to the Add a REST host workflow.
4 Right-click the Add a REST host workflow and select **Start workflow**.
5 In the **Name** text box, enter the name of the host.
6 In the **URL** text box, enter the address of the host.

   Note The Kerberos authentication requires a Fully Qualified Domain Name (FQDN) host address.

7 In the **Connection timeout** text box, enter the number of seconds before a connection times out.
8 In the **Operation timeout** text box, enter the number of seconds before an operation times out.
9 Select **Yes** to accept the REST host certificate.

   The certificate is added to the Orchestrator server trust store.

10 Select the authentication type.

Option	Description
None	No authentication is required.
OAuth 1.0	Provide the required authentication parameters.
OAuth 2.0	Provide the authentication token.
Basic	Provides basic access authentication. Select the session mode. ■ If you select **Shared Session**, provide credentials for the shared session. ■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in.
Digest	Provides digest access authentication that uses encryption. Select the session mode. ■ If you select **Shared Session**, provide credentials for the shared session. ■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in.

Option	Description
NTLM	Provides NT LAN Manager (NTLM) access authentication within the Window Security Support Provider (SSPI) framework. Select the session mode. ■ If you select **Shared Session**, provide credentials for the shared session. ■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in. Provide the NTLM settings.
Kerberos	Provides Kerberos access authentication. Select the session mode. ■ If you select **Shared Session**, provide credentials for the shared session. ■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in.

11 To use a proxy, enter the address and the port of the proxy server.

a (Optional) Select the proxy authentication type.

Option	Description
None	No authentication is required.
Basic	Provides basic access authentication. Select the session mode. ■ If you select **Shared Session**, provide credentials for the shared session. ■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in.

12 Select whether you want the target hostname to match the name stored in the server certificate.

13 (Optional) Select a keystore entry to use to authenticate against the server. The keystore entry must be of the `PrivateKeyEntry` type.

14 Click **Submit** to run the workflow.

Results

After the workflow runs successfully, the REST host appears in the **Inventory** view.

What to do next

You can add operations and XSD schema to the REST host, and run workflows from the **Inventory** view.

Add a REST Operation

You can run a workflow to add an operation to a REST host from the plug-in inventory.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a REST host from the **Inventory** view.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > HTTP-REST > Configuration** and navigate to the Add a REST operation workflow.
3. Right-click the Add a REST operation workflow and select **Start workflow**.
4. Select the host to which you want to add the operation.
5. In the **Name** text box, type the name of the operation.
6. In the **Template URL** text box, type only the operation part of the URL.

   You can include placeholders for parameters that are provided when you run the operation.

   The following is an example URL syntax.

   ```
   /customer/{id}/orders?date={date}
   ```
7. Select the HTTP method that the operation uses.

   If you select **POST** or **PUT**, you can provide a Content-Type request header for the method.
8. Click **Submit** to run the workflow.

What to do next

You can run workflows on the operation from the **Inventory** view.

Add a Schema to a REST Host

You can run a workflow to add an XSD schema to a REST host from the plug-in inventory.

The XSD schema describes the XML documents that are used as input and output content from Web services. By associating such a schema with a host, you can specify the XML element that is required as an input when you are generating a workflow from a REST operation.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a REST host from the **Inventory** view.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > HTTP-REST > Configuration** to navigate to the Add a schema to a REST host workflow.

3 Right-click the Add a schema to a REST host workflow and select **Start workflow**.

4 Select the host to which you want to add the XSD schema.

5 Select whether to load the schema from URL.

Option	Action
Yes	Type the URL of the schema.
No	Provide the schema content.

6 Click **Submit** to run the workflow.

Generate a New Workflow from a REST Operation

You can create a custom workflow from a REST operation.

You can integrate custom-generated workflows into high-level workflows. For more information about workflow development, see the *vRealize OrchestratorDeveloper's Guide*.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a REST host from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > HTTP-REST** and navigate to the Generate a new workflow from a REST operation workflow.

3 Right-click the Generate a new workflow from a REST operation workflow and select **Start workflow**.

4 Select the REST operation from the list of available operations.

If the operation takes input and XSD schemas are added to its host, you can specify the request input type.

5 In the **Name** text box, type the name of the workflow to generate.

6 Select the workflow folder in which to generate the new workflow.

You can select any existing folder from the workflow library.

7 Click **Submit** to run the workflow.

Invoking a REST Operation

To make REST requests, you can either invoke a configured REST operation or invoke a REST operation dynamically by using a configured REST operation as a template and replacing any of the parameters at runtime.

There are several ways to invoke a REST operation.

- Configure REST hosts and associate REST operations with them by running the **Add a REST Host** and **Add a REST Operation** workflows. The registered REST hosts and REST operations are persistent and can be found in the **Inventory** and **Resources** views.
- Invoke a REST operation without previously configuring REST hosts and adding REST operations by running the **Invoke a dynamic REST operation** workflow from **Library > HTTP-REST Samples**. With this workflow, you can provide REST host base URL and operation parameters. The data is not persistent and is not available in the **Inventory** and **Resources** views.
- Configure REST hosts, associate REST operations with them, and use the configured REST hosts and REST operations as templates for further use, by running the **Invoke a REST host with dynamic params** and **Invoke a REST operation with dynamic params** workflows from **Library > HTTP-REST Samples**. You can replace some of the parameters of already configured REST hosts and REST operations when you run the workflows. The original REST hosts and REST operations are not affected.

Invoke a REST Operation

Call a REST operation directly

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a REST host from the **Inventory** view.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > HTTP-REST** and navigate to the **Invoke a REST operation** workflow.
3. Right-click the **Invoke a REST operation** workflow and select **Start workflow**.
4. Select the REST operation from the list of available operations.
5. Provide the input parameters and content that the operation requires.
6. Click **Submit** to run the workflow.

Using the SOAP Plug-In

15

The SOAP plug-in allows you to manage SOAP Web services by providing interaction between vRealize Orchestrator and SOAP hosts. You can define SOAP services as inventory objects by running configuration workflows, and perform SOAP operations on the defined objects.

The plug-in contains a set of standard workflows related to managing SOAP hosts and invoking SOAP operations. You can also generate custom workflows to automate tasks in a SOAP environment.

This chapter includes the following topics:

Configuring the SOAP Plug-In

You must use the Orchestrator client to configure the SOAP plug-in.

Configuration Workflows

The Configuration workflow category contains workflows that allow you to manage SOAP hosts.

You can access these workflows from **Library > SOAP > Configuration** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Add a SOAP host	Adds a SOAP host to the plug-in inventory.
Reload plug-in configuration	Refreshes the list of SOAP hosts in the plug-in inventory.
Remove a SOAP host	Removes a SOAP host from the plug-in inventory. **Caution** When you remove a host from the inventory, all workflows generated from it stops working.

Workflow Name	Description
Update a SOAP host	Updates a SOAP host in the plug-in inventory.
Update a SOAP host with an endpoint URL	Updates a SOAP host with a preferred endpoint address. The new endpoint address is used for sending and receiving SOAP messages, instead of the endpoint address defined within the WSDL.

Add a SOAP Host

You can run a workflow to add a SOAP host and configure the host connection parameters.

Procedure

1 Log in to the Orchestrator client as an administrator.

2 Click the **Workflows** view in the Orchestrator client.

3 In the workflows hierarchical list, expand **Library > SOAP > Configuration** and navigate to the **Add a SOAP** host workflow.

4 Right-click the **Add a SOAP** host workflow and select **Start workflow**.

5 In the **Name** text box, enter the name of the host.

6 Select whether to provide the WSDL content as text.

Option	Action
Yes	Copy the text in the **WSDL content** text box.
No	Enter the correct path in the **WSDL URI** text box.

7 In the **Connection timeout** text box, enter the number of seconds, within which Orchestrator must connect to the SOAP host, otherwise the connection times out.

8 In the **Request timeout** text box, specify the number of seconds, within which a SOAP request must succeed, before it times out.

9 Select whether to use a proxy.

Option	Action
Yes	Provide the proxy address and proxy port.
No	Continue to the next step.

10 Select the authentication type.

Option	Description
None	No authentication is required.
Basic	Provides basic access authentication. Select the session mode. ■ If you select **Shared Session**, provide credentials for the shared session. ■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in.
Digest	Provides digest access authentication that uses encryption. Select the session mode. ■ If you select **Shared Session**, provide credentials for the shared session. ■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in.
NTLM	Provides NT LAN Manager (NTLM) access authentication within the Window Security Support Provider (SSPI) framework. Select the session mode. ■ If you select **Shared Session**, provide credentials for the shared session. ■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in. Provide the NTLM settings.
Kerberos	Provides Kerberos access authentication. Select the session mode. ■ If you select **Shared Session**, provide credentials for the shared session. ■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in.

11 Click **Submit** to run the workflow.

Results

After the workflow runs successfully, the SOAP host appears in the **Inventory** view.

What to do next

You can explore the SOAP host objects and run workflows on them from the **Inventory** view.

Configure Kerberos Authentication

You can use Kerberos authentication when you add and manage a PowerShell host.

With Kerberos authentication, domain users can run commands on remote PowerShell-enabled machines over WinRM.

Procedure

1 Enable Kerberos authentication on the WinRM service.

   a Run the following command to check whether Kerberos authentication is allowed.

   ```
   c:\> winrm get winrm/config/service
   ```

   b Run the following command to enable Kerberos authentication.

   ```
   c:\> winrm set winrm/config/service/auth @{Kerberos="true"}
   ```

2 Enable Kerberos authentication on the WinRM client.

   a Run the following command to check whether Kerberos authentication is allowed.

   ```
   c:\> winrm get winrm/config/client
   ```

   b Run the following command to enable Kerberos authentication.

   ```
   c:\> winrm set winrm/config/client/auth @{Kerberos="true"}
   ```

3 Run the following command to test the connection to the WinRM service.

   ```
   c:\> winrm identify -r:http://winrm_server:5985 -auth:Kerberos -u:user_name -
   p:password -encoding:utf-8
   ```

4 Create a `krb5.conf` file and save it to the following location.

Operating System	Path
Windows	`C:\Program Files\Common Files\VMware\VMware vCenter Server - Java Components\lib\security\`
Linux	`/usr/java/jre-vmware/lib/security/` for external vRealize Orchestrator. `/etc/krb5.conf` for vRealize Orchestrator that is built into vRealize Automation.

A `krb5.conf` file has the following structure:

```
[libdefaults]
default_realm = YOURDOMAIN.COM
udp_preference_limit = 1
[realms]
YOURDOMAIN.COM = {
kdc = kdc.yourdomain.com
default_domain = yourdomain.com
}
[domain_realm]
.yourdomain.com=YOURDOMAIN.COM
yourdomain.com=YOURDOMAIN.COM
```

The `krb5.conf` must contain specific configuration parameters with their values.

Kerberos configuration tags	Details
default_realm	The default Kerberos realm that a client uses to authenticate against an Active Directory server. **Note** Must be in uppercase letters.
kdc	The domain controller that acts as a Key Distribution Center (KDC) and issues Kerberos tickets.
default_domain	The default domain that is used to produce a fully qualified domain name. **Note** This tag is used for Kerberos 4 compatibility.

Note By default, the Java Kerberos configuration uses the UDP protocol. To use only the TCP protocol, you must specify the `udp_preference_limit` parameter with a value **1**.

Note The Kerberos authentication requires a Fully Qualified Domain Name (FQDN) host address.

Important When you add or modify the `krb5.conf` file, you must restart the Orchestrator server service.

Generate a New Workflow from a SOAP Operation

You can create a custom workflow from a SOAP operation.

You can integrate custom-generated workflows into high-level workflows. For more information about workflow development, see the *vRealize OrchestratorDeveloper's Guide*.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a SOAP host from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > SOAP** to navigate to the Generate a new workflow from a SOAP operation workflow.

3 Right-click the Generate a new workflow from a SOAP operation workflow and select **Start workflow**.

4 Select the SOAP operation from the list of available operations.

5 In the **Name** text box, type the name of the workflow to generate.

6 Select the workflow folder in which to generate the new workflow.

   You can select any existing folder from the workflow library.

7 Click **Submit** to run the workflow.

What to do next

You can test the generated workflow.

Test a Custom-Generated Workflow

You can run a custom workflow generated from a SOAP operation to get the output parameters of the operation.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a SOAP host from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.
2 Navigate to the workflow location.
3 Right-click the custom workflow and select **Start workflow**.
4 Provide the input parameters that the SOAP operation requires.
5 Click **Submit** to run the workflow.
6 (Optional) In the **Logs** tab, review the list of available output parameters.

Invoke a SOAP Operation

You can call a SOAP operation directly, without generating a new workflow.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a SOAP host from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.
2 In the workflows hierarchical list, expand **Library > SOAP** and navigate to the Invoke a SOAP operation workflow.
3 Right-click the Invoke a SOAP operation workflow and select **Start workflow**.
4 Select the SOAP operation from the list of available operations.
5 Provide the input parameters that the SOAP operation requires.
6 Click **Submit** to run the workflow.
7 (Optional) In the **Logs** tab, review the list of available output parameters.

Using the AMQP Plug-In

16

The AMQP plug-in allows you to interact with Advanced Message Queuing Protocol (AMQP) servers also known as brokers. You can define AMQP brokers and queue subscriptions as inventory objects by running configuration workflows, and perform AMQP operations on defined objects.

The plug-in contains a set of standard workflows related to managing AMQP brokers and calling AMQP operations.

This chapter includes the following topics:

Configuring the AMQP Plug-In

You must use the Orchestrator client to configure the AMQP plug-in.

Configuration Workflows

The Configuration workflow category contains workflows that allow you to manage AMQP brokers.

You can access these workflows from **Library > AMQP > Configuration** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Add a broker	Adds an AMQP broker.
Remove a broker	Removes an AMQP broker.
Remove a subscription	Removes an AMQP message subscription.
Subscribe to queues	Creates a new subscription element.
Update a broker	Updates broker properties.
Validate a broker	Validate a broker by attempting to start a connection.

Add a Broker

You can run a workflow to add an AMQP broker.

Procedure

1 Log in to the Orchestrator client as an administrator.

2 Click the **Workflows** view in the Orchestrator client.

3 In the hierarchical list of workflows, expand **Library > AMQP > Configuration** and navigate to the Add a broker workflow.

4 Right-click the Add a broker workflow and select **Start workflow**.

5 Provide the information required for the Add a broker workflow.

Option	Action
Name	Type the name of the broker.
Host	Type the address of the host.
Port	Type the port of the AMQP broker service. The default port is 5672.
Virtual host	Type the address of the virtual host. The default value provided is /.
Use SSL	Select whether to use SSL certificates.
Accept all certificates	Select whether to accept all SSL certificates without validation.
User name	Type the user name for the broker.
Password	Type the password for the broker.

6 Click **Submit** to run the workflow.

Results

After the workflow runs successfully, the AMQP broker appears in the **Inventory** view.

What to do next

You can run a Validate a broker workflow. If an error occurs, use the Update a broker workflow to change the broker's properties before validating again.

Subscribe to Queues

You can run a workflow to create a new subscription element.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an AMQP broker from the **Inventory** view.
- Verify that the AMQP broker has all queues included in the subscription declared.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the hierarchical list of workflows, expand **Library > AMQP > Configuration** and navigate to the Subscribe to queues workflow.
3. Right-click the Subscribe to queues workflow and select **Start workflow**.
4. In the **Name** text box, type the name of the queue to display.
5. Select the broker to which you want to add the subscription.
6. Select all the queues for message subscription.
7. Click **Submit** to run the workflow.

Results

After the workflow runs successfully, a child of the broker appears in the **Inventory** view.

What to do next

You can create a policy.

Update a Broker

You can run a workflow to update the broker properties.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an AMQP broker from the **Inventory** view.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the hierarchical list of workflows, expand **Library > AMQP** and navigate to the Update a broker workflow.
3. Right-click the Update a broker workflow and select **Start workflow**.
4. Select the broker that you want to update.

   Current properties of the broker appear.
5. Edit the properties that you want.
6. Click **Submit** to run the workflow.

Using the AMQP Plug-In Standard Workflows

The AMQP workflow category contains workflows that allow you to run AMQP operations.

You can access these workflows from **Library > AMQP** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Bind	Creates a binding in a specified broker.
Declare a queue	Adds a queue to a specified broker.
Declare an exchange	Adds an exchange to a specified broker.
Delete a queue	Deletes a queue from a specified broker.
Delete an exchange	Deletes an exchange from a specified broker.
Receive a text message	Receives a text message from a specified broker.
Send a test message	Sends a text message using a specified broker.
Unbind	Unbinds binding in a specified broker.

Declare a Binding

You can run a workflow to create a binding in a specified broker.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an AMQP broker from the **Inventory** view.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the hierarchical list of workflows, expand **Library > AMQP** and navigate to the Bind workflow.
3. Right-click the Bind workflow and select **Start workflow**.
4. Select a broker in which you want to create a binding.
5. Provide information about the binding.

Option	Action
Queue name	Type the name of the queue.
Exchange name	Type the name of the exchange.
Routing key	Type the routing key.

6. Click **Submit** to run the workflow.

Declare a Queue

You can run a workflow to add a queue to a specified broker.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an AMQP broker from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the hierarchical list of workflows, expand **Library > AMQP** and navigate to the Declare a queue workflow.

3 Right-click the Declare a queue workflow and select **Start workflow**.

4 Select a broker to which you want to add the queue.

5 In the **Name** text box, type the name of the queue to display.

6 Select whether the queue is durable.

Option	Description
Yes	The queue is removed after a broker restart.
No	The queue remains after a broker restart.

7 Select whether an exclusive client is set for the specific queue.

Option	Description
Yes	Sets one client for this specific queue.
No	Sets more clients for this specific queue.

8 Select whether to automatically delete the queue with activated subscription.

Option	Description
Yes	Automatically deletes the queue when no more clients are connected to it. The queue remains until at least one client subscribes to it.
No	Does not automatically delete the queue.

9 Click **Submit** to run the workflow.

Declare an Exchange

You can run a workflow to add an exchange in a specified broker.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an AMQP broker from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the hierarchical list of workflows, expand **Library > AMQP** and navigate to the Declare an exchange workflow.

3 Right-click the Declare an exchange workflow and select **Start workflow**.

4 Select a broker to which you want to add the exchange.

5 In the **Name** text box, type a name for the exchange.

6 Select the exchange type.

Option	Description
direct	Makes a direct match between the routing key provided in the message and the routing criteria used when a queue is bound to this exchange.
fanout	Forwards any message sent to this exchange to all queues bound to it. Queues that are bound to this exchange contain no arguments.
headers	Queues are bound to this exchange with a table of arguments that can contain headers and values. A special argument named x-match determines the matching algorithm.
topic	Performs a wildcard match between the routing key and the routing pattern specified in the binding.

7 Select whether the exchange is durable.

Option	Description
Yes	The exchange remains after a broker restart.
No	The exchange is removed after a broker restart.

8 Select whether to automatically delete the exchange with activated subscription.

Option	Description
Yes	Automatically deletes the exchange when no more queues are bound to it. The exchange remains until at least one queue is bound to it.
No	Does not automatically delete the exchange.

9 Click **Submit** to run the workflow.

Send a Text Message

You can run a workflow to send a text message using a specified broker.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an AMQP broker from the **Inventory** view.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the hierarchical list of workflows, expand **Library > AMQP** and navigate to the Send a text message workflow.
3. Right-click the Send a text message workflow and select **Start workflow**.
4. Select a broker from which you want to send a message.
5. In the **Exchange name** text box, specify the name of the exchange.
6. In the **Routing key** text box, specify the routing key.
7. In the **Content** text box, type the message you want to send.
8. Click **Submit** to run the workflow.

Delete a Binding

You can run a workflow to delete a binding in a specified broker.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an AMQP broker from the **Inventory** view.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the hierarchical list of workflows, expand **Library > AMQP** and navigate to the Unbind workflow.
3. Right-click the Unbind workflow and select **Start workflow**.
4. Select a broker to remove the binding from.
5. Provide information about the binding.

Option	Action
Queue name	Specify the name of the queue.
Exchange name	Specify the name of the exchange.
Routing key	Specify the routing key.

6. Click **Submit** to run the workflow.

Using the SNMP Plug-In

17

The SNMP plug-in allows vRealize Orchestrator to connect and receive information from SNMP-enabled systems and devices. You can define SNMP devices as inventory objects by running workflows, and perform SNMP operations on the defined objects.

You can use the plug-in to connect to SNMP devices such as routers, switches, network printers, and UPS devices. The plug-in can also receive events from vCenter Server over the SNMP protocol.

The SNMP plug-in provides two methods of communication with the SNMP devices.

- Queries for the values of specific SNMP variables.
- Listening for events (SNMP traps) that are generated from the devices and pushed to the registered SNMP managers.

The plug-in contains a set of standard workflows related to managing SNMP devices, queries, the trap host, and performing SNMP operations. You can also create custom workflows to automate tasks in an SNMP environment.

This chapter includes the following topics:

Managing SNMP Devices

You can run workflows to register SNMP devices with Orchestrator, edit the settings for existing devices, and unregister devices.

Device Management Workflows

The Device Management workflow category contains workflows that allow you to manage SNMP devices.

You can access these workflows from **Library > SNMP > Device Management** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Edit an SNMP device	Edits the configuration of a registered SNMP device.
Register an SNMP device	Registers an SNMP-enabled device to the plug-in inventory.
Unregister an SNMP device	Unregisters an SNMP device from the plug-in inventory.

Register an SNMP Device

You can run a workflow to register an SNMP device and optionally configure advanced connection parameters.

Procedure

1 Log in to the Orchestrator client as an administrator.

2 Click the **Workflows** view in the Orchestrator client.

3 In the workflows hierarchical list, expand **Library > SNMP > Device Management** and navigate to the Register an SNMP device workflow.

4 Right-click the Register an SNMP device workflow and select **Start workflow**.

5 In the **Device address** text box, enter the IP address or DNS name of the SNMP device.

 Note To establish a more reliable connection, you should use an IP address.

6 (Optional) In the **Name** text box, enter a name for the device as you want it to appear in the **Inventory** view.

 If you leave the text box blank, the device address is used to generate a name automatically.

7 (Optional) To configure the advanced connection parameters, select **Yes**.

 a In the **Port** text box, specify the connection port.

 The default port is 161.

 b From the **Version** drop-down menu, select the SNMP version that you want to use and provide the credentials.

 The support for SNMPv3 is limited to the AuthPriv security level with MD5 authentication. The DES passphrase is the same as the MD5 password.

 Note The support for SNMPv3 is deprecated.

8 Click **Submit** to run the workflow.

Results

After the workflow runs successfully, the SNMP device appears in the **Inventory** view.

What to do next

You can add queries to the SNMP device and run workflows from the **Inventory** view.

Managing SNMP Queries

You can add queries to registered SNMP devices, run, copy, and edit existing queries, and remove queries from SNMP devices. You can use SNMP queries as building blocks in more complex workflows.

Query Management Workflows

The Query Management workflow category contains workflows that allow you to manage SNMP queries.

You can access these workflows from **Library > SNMP > Query Management** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Add a query to an SNMP device	Adds a query to an SNMP device.
Copy an SNMP query	Copies an SNMP query from one device to another.
Edit an SNMP query	Edits an existing SNMP query.
Remove a query from an SNMP device	Removes an SNMP query from a device.
Run an SNMP query	Runs a query against an SNMP device.

Add a Query to an SNMP Device

You can run a workflow to add a query to an SNMP device from the plug-in inventory.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an SNMP device from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > SNMP > Query Management** and navigate to the Add a query to an SNMP device workflow.

3 Right-click the Add a query to an SNMP device workflow and select **Start workflow**.

4 Select the device to which you want to add the query.

5 From the **Type** drop-down menu, select the query type.

6 In the **OID** text box, type the object identifier of the variable that you want to query.

   The following are example OID values.

   - `1.3.6.1.2.1.1.5.0`
   - `.1.3.6.1.2.1.1.5.0`
   - `iso.3.6.1.2.1.1.5.0`

   Note The plug-in supports only OID values that are numerical or that begin with `iso` and continue with numbers.

7 (Optional) In the **Name** text box, type a name for the query.

   If you leave the text box blank, the type and OID parameters are used to generate a name automatically.

8 Click **Submit** to run the workflow.

What to do next

You can run workflows on the query from the **Inventory** view.

Managing the SNMP Trap Host

vRealize Orchestrator can act as an SNMP listener. You can start and stop the SNMP trap host, and change the port on which Orchestrator listens for SNMP traps.

The SNMP plug-in supports SNMPv1 and SNMPv2c traps.

Note The support for SNMPv3 is deprecated.

Trap Host Management Workflows

The Trap Host Management workflow category contains workflows that allow you to manage the SNMP trap host.

You can access these workflows from **Library > SNMP > Trap Host Management** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Set the SNMP trap port	Sets the port on which Orchestrator listens for SNMP traps.
Start the trap host	Orchestrator starts listening for SNMP traps.
Stop the trap host	Orchestrator stops listening for SNMP traps.

Set the SNMP Trap Port

You can run a workflow to set the port on which Orchestrator listens for SNMP traps.

The default port for SNMP traps is 162. However, on Linux systems, you can open ports bellow 1024 only with superuser privileges.

Note To ensure better compatibility, the default port for listening to SNMP traps in the SNMP plug-in is set to 4000.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an SNMP device from the **Inventory** view.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > SNMP > Trap Host Management** and navigate to the Set the SNMP trap port workflow.
3. Right-click the Set the SNMP trap port workflow and select **Start workflow**.
4. In the **Port** text box, enter the port number on which Orchestrator should listen for SNMP traps.
5. Click **Submit** to run the workflow.

Results

The workflow stops the trap host, sets the new port, and starts the trap host again.

Receiving SNMP Traps

The SNMP plug-in can receive SNMP traps by running a workflow, which waits for a single trap message, or with a policy, which can handle traps continuously. The plug-in supports SNMPv1 and SNMPv2c traps.

Wait for a Trap on an SNMP Device

You can run a workflow that waits to receive an SNMP trap from a specified device.

This workflow features a trigger, which stops the run of the workflow and waits for an SNMP trap before continuing. When a trap is received, the workflow run resumes. You can use the workflow as part of more complex workflows, or as a sample that you can customize or extend for a specific need.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an SNMP device from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > SNMP** and navigate to the Wait for a trap on an SNMP device workflow.

3 Right-click the Wait for a trap on an SNMP device workflow and select **Start workflow**.

4 Select the device on which you want to wait for an SNMP trap.

5 (Optional) In the **OID** text box, type the object identifier of a specific trap.

Note If you leave the text box empty, the workflow run resumes after receiving any trap from the specified SNMP device.

6 Click **Submit** to run the workflow.

Set an SNMP Trap Policy

You can set a policy to continuously listen for traps from an SNMP device that is already registered in the plug-in inventory.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an SNMP device from the **Inventory** view.

Procedure

1 From the drop-down menu in the Orchestrator client, select **Administer**.

2 Click the **Policy Templates** view.

3 In the workflows hierarchical list, expand **Library > SNMP** and navigate to the SNMP Trap policy template.

4 Right-click the SNMP Trap policy template and select **Apply Policy**.

5 In the **Policy name** text box, enter a name for the policy that you want to create.

6 (Optional) In the **Policy description** text box, enter a description for the policy.

7 Select an SNMP device for which to set the policy.

8 Click **Submit** to create the policy.

The Orchestrator client switches to **Run** perspective.

9 On the **Policies** view, right-click the policy that you created and select **Start policy**.

Results

The trap policy starts to listen for SNMP traps.

What to do next

You can edit the SNMP Trap policy.

Configure an SNMP Trap Host Policy

With the SNMP Trap Host policy listens for SNMP traps from hosts that might not be added as registered SNMP devices.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an SNMP device from the **Inventory** view.

Procedure

1. From the drop-down menu in the Orchestrator client, select **Administer**.
2. Click the **Policy Templates** view.
3. In the workflows hierarchical list, expand **Library > SNMP** and navigate to the SNMP Trap Host policy template.
4. Right-click the SNMP Trap Host policy template and select **Apply Policy**.
5. In the **Policy name** text box, enter a name for the policy that you want to create.
6. (Optional) In the **Policy description** text box, enter a description for the policy.
7. Select `Trap Host (Online)` from the inventory tree.
8. Click **Submit** to create the policy.

   The Orchestrator client switches to **Run** perspective.
9. Right-click the policy and select **Edit**.
10. In the `Scripting` tab, expand **host > OnTrapAll**.
    a. Select a workflow or a script to associate with the policy.
11. Click **Save and close** to apply the edited settings.
12. On the **Policies** view, right-click the policy that you edited and select **Start policy**.

What to do next

You can edit the SNMP Trap Host policy.

Edit a Trap Policy

You can edit a trap policy to customize it for a specific use case. When you edit a trap policy, you can change its priority and startup settings, as well as customize the scripting and permissions associated with the policy.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to an SNMP device from the **Inventory** view.

Procedure

1. Click the **Policies** view in the Orchestrator client.
2. If the policy that you want to edit is running, right-click the policy and select **Stop policy**.
3. Right-click the policy and select **Edit**.
4. From the **General** tab, edit the startup settings, priority, and description of the policy.
5. (Optional) From the **Scripting** tab, you can associate a specific workflow or scripting code with the policy, for integration in a more complex scenario.

   You can set the policy to trigger a custom workflow when a trap is received.
6. (Optional) From the **Permissions** tab, you can modify the access rights.

   You can give permissions to a user or to a group to start the policy, without giving permissions to edit the policy.
7. Click **Save and close** to apply the edited settings.
8. On the **Policies** view, right-click the policy that you edited and select **Start policy**.

Generic SNMP Request Workflows

The SNMP workflow category contains workflows that allow you to perform basic SNMP requests without having to create a query.

You can access these workflows from **Library > SNMP** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Get bulk SNMP values	Runs a GET BULK query against an SNMP device.
Get next SNMP value	Runs a GET NEXT query against an SNMP device.
Get SNMP value	Runs a GET query against an SNMP device.
Send an SNMP trap	Sends an SNMP trap to a specified address.
Wait for a trap on all devices	Waits to receive an SNMP trap from all hosts trat send traps to Orchestrator.
Wait for a trap on an SNMP device	Waits to receive an SNMP trap from a specified device.

Using the Active Directory Plug-In 18

The Active Directory plug-in (VMware vRealize Orchestrator plug-in for Microsoft Active Directory) allows interaction between vRealize Orchestrator and Microsoft Active Directory. You can use the plug-in to run Orchestrator workflows that automate Active Directory processes.

The plug-in contains a set of standard workflows. You can also create custom workflows that implement the plug-in API to automate tasks in your Active Directory environment.

This chapter includes the following topics:

Configuring the Active Directory Plug-In

To connect to a Microsoft Active Directory instance by using the Active Directory plug-in, you must configure the connection parameters for the Microsoft Active Directory instance.

You can configure Active Directory by running the configuration workflows included in the plug-in.

Active DirectoryConfiguration Workflows

The Configuration workflow category of the Active Directory plug-in contains workflows that allow you to configure Active Directory servers and manage SSL certificates.

You can access these workflows from **Library > Microsoft > Active Directory > Configuration** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Add an Active Directory server	Configures a Microsoft Active Directory server.
Configure Active Directory plug-in options	Configures the search limitation options of the Active Directory plug-in.
Configure Active Directory server (Deprecated)	Creates or updates the default Active Directory server configuration. Use Update an Active Directory server.
Remove an Active Directory server	Removes an Active Directory server configuration.

Workflow Name	Description
Reset configuration (Deprecated)	Deletes the default Active Directory server configuration. Use Remove an Active Directory server.
Update an Active Directory server	Modifies an existing Active Directory server configuration.

Using the Active Directory Plug-In Workflow Library

The Active Directory plug-in workflow library contains workflows that allow you to run automated processes related to the management of Microsoft Active Directory objects.

The workflows are grouped into categories depending on object type. You can integrate standard workflows from the workflow library in custom workflows.

Using the Active Directory Plug-In Inventory

The Active Directory plug-in exposes all objects in the connected Microsoft Active Directory instance in the **Inventory** view. You can use the **Inventory** view to add authorization elements or to run workflows on Microsoft Active Directory objects.

To display the workflows that are available for an inventory object, navigate to **Tools > User preferences > Inventory** and select the **Use contextual menu in inventory** check box. After the option is enabled, when you right-click an object in the Orchestrator inventory, all available workflows for the object are displayed.

Access the Active Directory Plug-In Workflow Library

You must use the Orchestrator client to access the elements from the Active Directory plug-in workflow library.

Procedure

1. Log in to the Orchestrator client as an administrator.
2. Click the **Workflows** view.
3. In the hierarchical list, expand **Library > Microsoft > Active Directory** and expand the selection.

Active Directory Plug-In Workflows

The Active Directory plug-in contains a set of standard workflows that cover the most common LDAP functionality. You can use the workflows as building blocks for creating complex custom solutions. By combining standard workflows, you can automate multistep processes in the Active Directory enironment.

Computer Workflows

The Computer workflow category contains workflows related to Active Directory computer management.

You can access these workflows from **Library > Microsoft > Active Directory > Computer**.

Workflow Name	Description
Create a computer in a group	Creates an Active Directory computer in a group.
Create a computer in an organizational unit	Creates an Active Directory computer in an organizational unit.
Destroy a computer	Deletes a computer from an Active Directory instance.
Destroy a computer and delete its subtree	Deletes a computer from an Active Directory instance and all objects within the computer subtree.
Disable a computer	Disables a computer from an Active Directory instance.
Enable a computer	Enables a computer in an Active Directory instance.

Organizational Unit Workflows

The Organizational Unit workflow category contains workflows related to Active Directory organizational unit management.

You can access these workflows from **Library > Microsoft > Active Directory > Organizational Unit**.

Workflow Name	Description
Create an organizational unit	Creates an organizational unit in an existing organizational unit.
Destroy an organizational unit	Deletes an organizational unit from an Active Directory instance.
Destroy an organizational unit and delete its subtree	Deletes an organizational unit from an Active Directory instance and all objects within the organizational unit subtree.

User Workflows

The User workflow category contains workflows related to Active Directory user management.

You can access these workflows from **Library > Microsoft > Active Directory > User**.

Workflow Name	Description
Add a user to a user group	Adds one user as a member of a user group.
Change a user password	Changes the password for a user. SSL connection is required, and the password must meet the Active Directory restrictions.
Create a user in a group	Creates a user without specifying a password. The password must be changed at the next login. Domain policies must allow users to have empty passwords.
Create a user in an organizational unit	Creates a user in an organizational unit. If SSL connection is disabled, you cannot specify a password. Domain policies must allow users to have empty passwords.
Create a user with a password in a group	Creates a user and sets a password for the user. The password can be changed at the next login.
Create a user with a password in an organizational unit	Creates a user in an organizational unit and sets a password for the user. The password can be changed at the next login. If SSL connection is disabled, you cannot specify a password.
Destroy a user	Deletes a user from an Active Directory instance.

Workflow Name	Description
Disable a user	Disables a user from an Active Directory instance.
Enable a user	Enables a user in an Active Directory instance.
Remove a user from a user group	Removes a user from a user group.

User Group Workflows

The User Group workflow category contains workflows related to Active Directory user group management.

You can access these workflows from **Library > Microsoft > Active Directory > User Group**.

Workflow Name	Description
Add computers to group members	Adds one or more computers as members of a user group.
Add groups to group members	Adds one or more user groups as members of a user group.
Add users to group members	Adds one or more users as members of a user group.
Create a user group in a group	Creates a user group in an existing container (group).
Create a user group in a group and set attribute “Group name (pre-Windows 2000)”	Creates a user group in an existing container (organizational unit) and sets the `Group name (pre-Windows 2000)` attribute.
Create a user group in an organizational unit	Creates a user group in an existing container (organizational unit).
Destroy a user group	Deletes a user group from an Active Directory instance.
Remove computers from group members	Removes one or more computers from a user group.
Remove groups from group members	Removes one or more user groups from a user group.
Remove users from group members	Removes one or more users from a user group.

Using the Dynamic Types Plug-In 19

The Orchestrator Dynamic Types plug-in lets you define dynamic types, create objects of these types, and set relations between them.

The definition of a dynamic type contains the descriptions of its properties and a set of finder workflows and actions which can be used to find dynamic objects of this type. Runtime instances of dynamic types are called dynamic objects. You can run workflows on the dynamic objects you create and perform different operations on them.

Each dynamic type must be defined in a namespace. Namespaces are helper dynamic objects that let you group dynamic types in containers.

You can use the Dynamic Types plug-in together with the HTTP-REST plug-in to integrate third-party REST API services into the Orchestrator and expose third-party objects as Orchestrator types.

1 Define a new dynamic type and its properties by running the Define Namespace and Define Type workflows from the Dynamic Types plug-in. In result, you obtain a set of finder and inventory workflows for finding objects of the new dynamic type and their relations with other objects.

2 Modify the new finder and inventory workflows, so that they receive their input from the third-party REST API.

   a Create REST operations by using the Add a REST Operation workflow from the HTTP-REST plug-in and map these operations to the corresponding REST API methods.

   b Modify the finder and inventory workflows to invoke these REST operations and consume their outputs.

This chapter includes the following topics:

Dynamic Types Configuration Workflows

The workflows in the Configuration package of the Dynamic Types plug-in let you create dynamic types, export and import type definitions from an XSD file, and define relations between the dynamic types you created.

You can access these workflows from **Library > Dynamic Types > Configuration** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Define Namespace	Defines a new namespace.
Define Relation	Defines a new relation between types.
Define Type	Defines a new type within a given namespace.
Export Configuration As Package	Exports a Dynamic Types definition configuration as a file-based configuration. The exported package can be used for importing to other servers.
Import Configuration From Package	Imports a file-based configuration to the plug-in configuration.
Import Type Definitions From XSD	Imports type definitions from an XSD file.
Remove Namespace	Removes a namespace.
Remove Relation	Removes a relation.
Remove Type	Removes a type.
Update Namespace	Updates a namespace.
Update Type	Updates a type.

Using the PowerShell Plug-In

20

The PowerShell plug-in workflow library contains workflows that allow you to manage PowerShell hosts and run custom PowerShell operations.

You can use the **Inventory** view in the Orchestrator client to manage the available PowerShell resources. You can use the scripting API of the plug-in to develop custom workflows.

This chapter includes the following topics:

Introduction to the VMware vRealize Orchestrator PowerShell Plug-In

The PowerShell plug-in allows interaction between vRealize Orchestrator and Windows PowerShell.

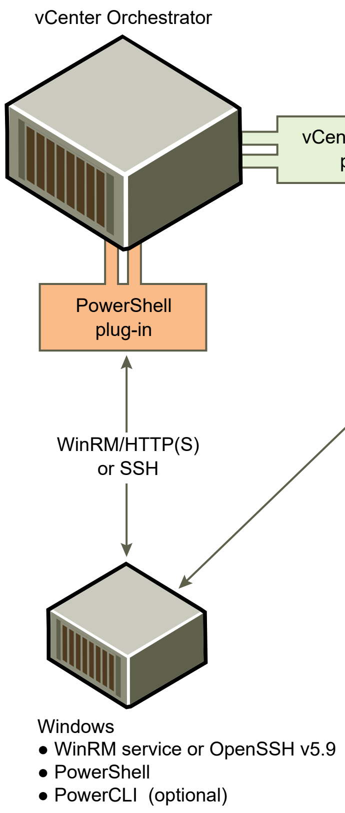
You use the plug-in to call PowerShell scripts and cmdlets from Orchestrator actions and workflows, and to work with the result. The plug-in contains a set of standard workflows. You can also create custom workflows that implement the plug-in API.

PowerShell Plug-In Components

The PowerShell plug-in relies on a number of components to function properly.

vRealize Orchestratorand Windows PowerShell provide the platform for the plug-in, and the plug-in provides interaction between those products. The PowerShell plug-in can also interact with other components, such as vCenter Server and vSphere PowerCLI.

Figure 20-1. Component Relations



The plug-in can communicate with Windows PowerShell through the OpenSSH and WinRM communication protocols. See Configuring WinRM.

Optionally, you can integrate the PowerShell plug-in with vSphere PowerCLI and vCenter Server. See PowerCLI Integration with the PowerShell Plug-In.

Note You can install all components on a local host. The usage, functionality, and communication protocol requirements of the PowerShell plug-in do not change if vRealize Orchestrator and Windows PowerShell are installed on the same machine.

Role of vRealize Orchestrator with the PowerShell Plug-In

You must use the Orchestrator configuration interface to install the PowerShell plug-in. You use the Orchestrator client to run and create workflows and access the plug-in API.

The PowerShell plug-in is powered by vRealize Orchestrator. Orchestrator is a development and process-automation platform that provides a library of extensible workflows to manage the VMware vCenter infrastructure and other technologies.

Orchestrator allows integration with management and administration solutions through its open plug-in architecture. PowerShell is one example of an administration solution that you can integrate with Orchestrator by using plug-ins.

Plug-In Interaction with Windows PowerShell

You can use the plug-in to run Orchestrator workflows that interact with Windows PowerShell hosts and perform tasks, such as invoking PowerShell scripts.

Windows PowerShell is a task-based command-line shell and scripting language designed for system administration.

Configuring WinRM

To establish a connection between the PowerShell plug-in and Windows PowerShell, you must configure WinRM to use one of the supported communication protocols.

The PowerShell plug-in supports Windows Remote Management (WinRM) 2.0 as a management protocol.

The following authentication methods are supported.

Authentication method	Details
Basic	Non-secure authentication mechanism that requires a user name and a password.
Kerberos	Secure authentication protocol that uses tickets to verify the identity of the client and the server.

Note The PowerShell plug-in does not support delegation of user credentials in WinRM and CredSSP is not a supported authentication method.

WinRM Through HTTP

The PowerShell plug-in supports communication with the WinRM host through the HTTP protocol. Although WinRM authenticates the communication, the data transfer is not encrypted and is sent as plain text on the network. You should use the HTTP protocol if IPSec is configured between the machines that communicate.

To use Basic authentication, you must set the `AllowUnencrypted` property to **`true`** in both the service and client WinRM configuration. For an example of HTTP configuration, see Configure WinRM to Use HTTP.

WinRM Through HTTPS

The PowerShell plug-in supports communication with the WinRM host through the HTTPS protocol. You can use the HTTPS protocol as a more secure communication method.

To use the HTTPS protocol, you must generate a certificate for server authentication and install the certificate on the WinRM host. For an example of HTTPS configuration, see Configure WinRM to Use HTTPS.

Configure WinRM to Use HTTP

You can configure the WinRM host to enable communication with the PowerShell plug-in through the HTTP protocol.

You must modify the WinRM configuration by running commands on the WinRM host machine. You can use the same machine as both the WinRM service and WinRM client.

Important If you skip any of the steps when configuring WinRM to use HTTP, the host might not be added, and you might get an error message in the logs such as

```
Caused by: org.dom4j.DocumentException: Error on line -1 of document : Premature end of file.
Nested exception: Premature end of file.
at org.dom4j.io.SAXReader.read(SAXReader.java:482)
at org.dom4j.DocumentHelper.parseText(DocumentHelper.java:278)
at
com.xebialabs.overthere.cifs.winrm.connector.JdkHttpConnector.sendMessage(JdkHttpConnector.java:117)
```

Procedure

1 Run the following command to set the default WinRM configuration values.

```
c:\> winrm quickconfig
```

2 (Optional) Run the following command to check whether a listener is running, and verify the default ports.

```
c:\> winrm e winrm/config/listener
```

The default ports are 5985 for HTTP, and 5986 for HTTPS.

3 Enable basic authentication on the WinRM service.

  a Run the following command to check whether basic authentication is allowed.

  ```
  c:\> winrm get winrm/config/service
  ```

  b Run the following command to enable basic authentication.

  ```
  c:\> winrm set winrm/config/service/auth @{Basic="true"}
  ```

4 Run the following command to allow transfer of unencrypted data on the WinRM service.

```
c:\> winrm set winrm/config/service @{AllowUnencrypted="true"}
```

5 If the channel binding token hardening level of the WinRM service is set to `strict`, change its value to `relaxed`.

```
c:\> winrm set winrm/config/service/auth @{CbtHardeningLevel="relaxed"}
```

6 Enable basic authentication on the WinRM client.

   a Run the following command to check whether basic authentication is allowed.

   ```
   c:\> winrm get winrm/config/client
   ```

   b Run the following command to enable basic authentication.

   ```
   c:\> winrm set winrm/config/client/auth @{Basic="true"}
   ```

7 Run the following command to allow transfer of unencrypted data on the WinRM client.

   ```
   c:\> winrm set winrm/config/client @{AllowUnencrypted="true"}
   ```

8 If the WinRM host machine is in an external domain, run the following command to specify the trusted hosts.

   ```
   c:\> winrm set winrm/config/client @{TrustedHosts="host1, host2, host3"}
   ```

9 Run the following command to test the connection to the WinRM service.

   ```
   c:\> winrm identify -r:http://winrm_server:5985 -auth:basic -u:user_name -
   p:password -encoding:utf-8
   ```

Configure WinRM to Use HTTPS

You can configure the WinRM host to enable communication with the PowerShell plug-in through the HTTPS protocol.

The WinRM host requires a certificate so that it can communicate through the HTTPS protocol. You can either obtain a certificate or generate one. For example, you can generate a self-signed certificate by using the Certificate Creation tool (`makecert.exe`) that is part of the .NET Framework SDK.

Prerequisites

- Configure WinRM to use the HTTP protocol. For more information, see Configure WinRM to Use HTTP.
- Verify that you can access the Microsoft Management Console (`mmc.exe`) on the WinRM host.

Procedure

1 Generate a self-signed certificate.

   The following command line contains example syntax for creating a certificate on the WinRM host by using `makecert.exe`.

   ```
   makecert.exe -r -pe -n "CN=host_name-3,O=organization_name" -e mm/dd/yyyy -eku
   1.3.6.1.5.5.7.3.1 -ss my -sr localMachine -sky exchange -sp "Microsoft RSA
   SChannel Cryptographic Provider" -sy 12 certificate_name.cer
   ```

2 Add the generated certificate by using the Microsoft Management Console.

   a Run `mmc.exe`.

   b Select **File > Add/Remove Snap-in**.

c From the list of available snap-ins, select **Certificates** and click **Add**.

d Select **Computer account** and click **Next**.

e Click **Finish**.

f Verify that the certificate is installed in **Console Root > Certificates (Local Computer) > Personal > Certificates** and **Console Root > Certificates (Local Computer) > Trusted Root Certification Authorities > Certificates**.

If the certificate is not installed in the Trusted Root Certification Authorities and Personal folders, you must install it manually.

3 Create an HTTPS listener by using the correct thumbprint and host name.

The following command line contains example syntax for creating an HTTPS listener.

```
winrm create winrm/config/Listener?Address=*+Transport=HTTPS
@{Hostname="host_name";CertificateThumbprint="certificate_thumbprint"}
```

Note Omit the spaces in the certificate thumbprint.

4 Test the connection.

The following command line contains example syntax for testing the connection.

```
winrs -r:https://host_name:port_number -u:user_name -p:password hostname
```

Configure Kerberos Authentication

You can use Kerberos authentication when you add and manage a PowerShell host.

With Kerberos authentication, domain users can run commands on remote PowerShell-enabled machines over WinRM.

Procedure

1 Enable Kerberos authentication on the WinRM service.

a Run the following command to check whether Kerberos authentication is allowed.

```
c:\> winrm get winrm/config/service
```

b Run the following command to enable Kerberos authentication.

```
c:\> winrm set winrm/config/service/auth @{Kerberos="true"}
```

2 Enable Kerberos authentication on the WinRM client.

a Run the following command to check whether Kerberos authentication is allowed.

```
c:\> winrm get winrm/config/client
```

b Run the following command to enable Kerberos authentication.

```
c:\> winrm set winrm/config/client/auth @{Kerberos="true"}
```

3 Run the following command to test the connection to the WinRM service.

```
c:\> winrm identify -r:http://winrm_server:5985 -auth:Kerberos -u:user_name -
p:password -encoding:utf-8
```

4 Create a `krb5.conf` file and save it to the following location.

Operating System	Path
Windows	`C:\Program Files\Common Files\VMware\VMware vCenter Server - Java Components\lib\security\`
Linux	`/usr/java/jre-vmware/lib/security/` for external vRealize Orchestrator. `/etc/krb5.conf` for vRealize Orchestrator that is built into vRealize Automation.

A `krb5.conf` file has the following structure:

```
[libdefaults]
default_realm = YOURDOMAIN.COM
udp_preference_limit = 1
[realms]
YOURDOMAIN.COM = {
kdc = kdc.yourdomain.com
default_domain = yourdomain.com
}
[domain_realm]
.yourdomain.com=YOURDOMAIN.COM
yourdomain.com=YOURDOMAIN.COM
```

The `krb5.conf` must contain specific configuration parameters with their values.

Kerberos configuration tags	Details
default_realm	The default Kerberos realm that a client uses to authenticate against an Active Directory server. **Note** Must be in uppercase letters.
kdc	The domain controller that acts as a Key Distribution Center (KDC) and issues Kerberos tickets.
default_domain	The default domain that is used to produce a fully qualified domain name. **Note** This tag is used for Kerberos 4 compatibility.

Note By default, the Java Kerberos configuration uses the UDP protocol. To use only the TCP protocol, you must specify the `udp_preference_limit` parameter with a value 1.

Note The Kerberos authentication requires a Fully Qualified Domain Name (FQDN) host address.

Important When you add or modify the `krb5.conf` file, you must restart the Orchestrator server service.

Configuring the PowerShell Plug-In

You must use the Orchestrator client to configure the PowerShell plug-in.

Configuration Workflows

The Configuration workflow category contains workflows that allow you to manage PowerShell hosts.

You can access these workflows from **Library > PowerShell > Configuration** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Add a PowerShell host	Adds a PowerShell host to the plug-in inventory.
Remove a PowerShell host	Removes a PowerShell host from the plug-in inventory.
Update a PowerShell host	Updates the specified PowerShell host in the plug-in inventory.
Validate a PowerShell host	Validates the configuration of the specified PowerShell host.

Add a PowerShell Host

You add a PowerShell host and configure the host connection parameters by running a workflow. You can set up a connection to a remote or a local PowerShell host.

Procedure

1. Log in to the Orchestrator client as an administrator.
2. Click the **Workflows** view in the Orchestrator client.
3. In the workflows hierarchical list, expand **Library > PowerShell > Configuration** and navigate to the Add a PowerShell host workflow.
4. Right-click the Add a PowerShell host workflow and select **Start workflow**.
5. In the **Name** text box, type the name of the host.
6. In the **Host / IP** text box, type the address of the host.

   Note The Kerberos authentication requires a Fully Qualified Domain Name (FQDN) host address.

7. (Optional) In the **Port** text box, type the port of the host.

   You use port 5985 for the HTTP or 5986 for the HTTPS protocol.

8 Select the PowerShell host type that the plug-in connects to.

   a Select a transport protocol.

   Note If you use the HTTPS transport protocol, the certificate of the remote PowerShell host is imported into the Orchestrator keystore.

   b Select the authentication type.

   Important If you want to use Kerberos authentication, you must enable it on the WinRM service.

9 Select the type of session mode that the plug-in uses to connect to the PowerShell host.

Option	Description
Shared Session	The plug-in uses shared credentials to connect to the remote host. You must provide the PowerShell host credentials for the shared session.
Session per User	The Orchestrator client retrieves credentials from the user who is logged in. You must log in with a ***user@domain*** format to Orchestrator to use the **Session per User** mode.

10 From the **Shell Code Page** drop-down menu, select the type of encoding that the PowerShell uses.

11 Click **Submit** to run the workflow.

Results

After the workflow runs successfully, the PowerShell host appears in the **Inventory** view.

Using the PowerShell Plug-In Inventory

The PowerShell plug-in exposes all objects in the connected PowerShell hosts in the **Inventory** view. You can use the **Inventory** view to add authorization elements or to run workflows on PowerShell objects.

Within the inventory of the plug-in, you can monitor PowerShell hosts and their snap-ins and cmdlets. Each remote host can contain snap-ins and each snap-in can contain cmdlets.

To display the workflows that are available for an inventory object, navigate to **Tools > User preferences > Inventory** and select the **Use contextual menu in inventory** check box. After the option is enabled, when you right-click an object in the Orchestrator inventory, all available workflows for the object are displayed.

Running PowerShell Scripts

You can run workflows to invoke an external or custom script on a PowerShell host.

Invoke a PowerShellScript

You can run an existing or custom PowerShell script on a host in the plug-in inventory.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a PowerShell host from the **Inventory** view.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > PowerShell** and navigate to the Invoke a PowerShell script workflow.
3. Right-click the Invoke a PowerShell script workflow and select **Start workflow**.
4. Select a PowerShell host on which to run the script.
5. In the **Script** text box, type or paste the PowerShell script that you want to run.
6. Click **Submit** to run the workflow.

Invoke an External Script

You can run an external PowerShell script on a host in the plug-in inventory.

External PowerShell scripts are contained in `.ps1` files. The `.ps1` file that you want to run must be stored on the PowerShell host.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a PowerShell host from the **Inventory** view.
- Verify that you have access to other `.ps1` files that the script might reference.

Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, expand **Library > PowerShell** and navigate to the Invoke an external script workflow.
3. Right-click the Invoke an external script workflow and select **Start workflow**.
4. Select a PowerShell host on which to run the script.
5. In the **Name** text box, type the filename of the external `.ps1` script that you want to run.

   Note If the `.ps1` file is not in the default folder, you must type the absolute file path. You can use system environment variables to specify script paths. For example, `$env:HOMEPATH\test1.ps1`.

6 In the **Arguments** text box, type the script arguments.

   The valid syntax is the same as used in the PowerShell console.

7 Click **Submit** to run the workflow.

Generating Actions

You can run workflows to generate actions based on a PowerShell script or a PowerShell cmdlet. You can use the generated actions as building blocks for custom workflows.

Generate an Action from a PowerShell Script

You can run a workflow to generate an action from a PowerShell script that you provide. You can optionally generate a sample workflow that can run the generated action.

You can customize the script of the action that you generate by using placeholders. For each placeholder, the workflow creates a corresponding action parameter of type `string` in the generated action. When you run the action, you can provide an actual value as the action parameter to replace the placeholder.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a PowerShell host from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > PowerShell > Generate** and navigate to the Generate an action from a PowerShell script workflow.

3 Right-click the Generate an action from a PowerShell script workflow and select **Start workflow**.

4 In the **Script** text box, type or paste the PowerShell script from which to generate the action.

   Note You can use **`{#ParamName#}`** as a placeholder for user input. If the placeholder is of type `string`, you must use double quotes to pass the value of the placeholder to the action.

   The following script is an example of how to link the generated action parameter to a script parameter.

```
param($name={#ParamName#})
echo $name;
```

5 In the **Name** text box, type a name for the action that you want to generate.

6 Select an existing module in which to generate the action.

7 Select whether to generate a workflow.

Option	Description
Yes	Generates a sample workflow that can run the generated action. You must select a folder in which to generate the workflow. **Note** The name of the generated workflow consists of the predefined string Invoke Script and the name of the generated action.
No	A sample workflow is not generated.

8 Click **Submit** to run the workflow.

What to do next

You can integrate the generated action in custom workflows.

Generate an Action for a PowerShell Cmdlet

You can run a workflow to generate an action for a PowerShell cmdlet and parameter set that you provide. With this action, you can use PowerShell functionality in Orchestrator. You can optionally generate a sample workflow that runs the generated action.

You can use a large set of data types with the PowerShell script engine. The data types that you can use include primitive types such as `Integer`, `Boolean`, `Char`, any type available from the .NET assembly, or user-defined types. When generating actions based on PowerShell cmdlet definitions, the input and output cmdlet parameters are represented by types that the Orchestrator platform supports. The PowerShell plug-in defines the type mappings. In general, primitive types are mapped to the corresponding Orchestrator types, and complex types are represented by the `PowerShellRemotePSObject` object.

Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that you have a connection to a PowerShell host from the **Inventory** view.

Procedure

1 Click the **Workflows** view in the Orchestrator client.

2 In the workflows hierarchical list, expand **Library > PowerShell > Generate** and navigate to the Generate an action for a PowerShell cmdlet workflow.

3 Right-click the Generate an action for a PowerShell cmdlet workflow and select **Start workflow**.

4 Select a PowerShell cmdlet to run when using the action that you generate.

5 Select a parameter set for the cmdlet.

The parameter set definition values appear in the **Parameter set definition** text box.

Note You cannot modify the parameter set definition values by editing the string in the **Parameter set definition** text box. You can review the string for information about the parameters that the parameter set contains.

6 In the **Name** text box, type a name for the action that you want to generate.

7 Select an existing module in which to generate the action.

8 Select whether to generate a workflow.

Option	Description
Yes	Generates a sample workflow that can run the generated action. You should select a folder in which to generate the workflow. **Note** The name of the generated workflow consists of the predefined string Execute Cmdlet and the name of the generated action.
No	A sample workflow is not generated.

9 Click **Submit** to run the workflow.

What to do next

You can integrate the generated action in custom workflows.

Passing Invocation Results Between Actions

The PowerShell plug-in supports passing of results as parameters from one PowerShell script invocation to another. To pass results correctly, both invocations must happen in the same session.

PowerCLI Integration with the PowerShell Plug-In

You can use functionality that is available in a third-party snap-in, such as VMware vSphere PowerCLI, with the PowerShell plug-in.

To use the third-party snap-in functionality, the snap-in must be available on the PowerShell host. To load the snap-in in the current session, you must also invoke the `AddPsSnapin` action. When using PowerCLI, you must set the name of the snap-in to `VMware.VimAutomation.Core`.

The PowerShell plug-in does not provide pre-generated actions for third-party snap-ins. You can generate actions for third-party snap-ins by running the Generate an action for a PowerShell cmdlet workflow. See Generate an Action for a PowerShell Cmdlet.

The `com.vmware.library.powershell.converter` package contains basic building blocks that allow conversion from a `VC:<SomeObjectType>` object, to the corresponding PowerCLI object. This feature allows workflows from the vCenter Server plug-in to interact with workflows from the PowerShell plug-in and to pass parameters between the two plug-ins.

Converter Workflows

You can use the sample workflows from the Converter workflow category to test the integration between the PowerShell plug-in and PowerCLI. To test the integration, PowerCLI must be installed on the PowerShell host.

The Converter sample workflows demonstrate the conversion functionality available in the plug-in.

Note The PowerShell plug-in does not support all types that are available in PowerCLI and the vCenter Server plug-in. Unsupported types return an exception.

You can access these workflows from **Library > PowerShell > Samples > Converter** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Convert PSObject to vCO object	Converts `PowerShellRemotePSObject` to `VC:<SomeObjectType>`.
Convert PSObject to vCO object to PSObject	Converts `PowerShellRemotePSObject` to `VC:<SomeObjectType>` and the reverse.
Convert vCO object to PSObject	Converts `VC:<SomeObjectType>` to `PowerShellRemotePSObject`.

Sample Workflows

The Samples workflow category contains workflows that allow you to test basic use cases.

You can access these workflows from **Library > PowerShell > Samples** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Invoke a script via API	Demonstrates how to call a PowerShell script through the available scripting API.
List directory content	Lists the contents of a directory on the PowerShell host file system.
Pipeline execution example	Demonstrates how you can run multiple cmdlets arranged into a pipe.
Toggle virtual machine state	Toggles the power state of a virtual machine.

Access the PowerShell Plug-In API

With the Orchestrator API Explorer you can search the PowerShell plug-in API and see the documentation for JavaScript objects that you can use in scripted elements.

Procedure

1 Log in to the Orchestrator client as an administrator.

2 Access the API Explorer from either the Orchestrator client or from the **Scripting** tabs of the workflow, policy, and action editors.

   - To access the API Explorer from the Orchestrator client, click **Tools > API Explorer** in the Orchestrator client toolbar.
   - To access the API Explorer from the **Scripting** tabs of the workflow, policy, and action editors, click **Search API** on the left.

3 To expand the hierarchical list of PowerShell plug-in API objects, double-click the **PowerShell** module in the left pane.

What to do next

You can copy code from API elements and paste it into scripting boxes. For more information about API scripting, see *Developing with VMware vRealize Orchestrator*.

Working with PowerShell Results

You can use objects from the PowerShell plug-in API to work with results that Windows PowerShell returns.

You can use the methods from the `PowerShellInvocationResult` class to retrieve information about a script that you run.

Method	Description
`getErrors()`	Returns a list of errors reported by the PowerShell engine during script invocation.
`getInvocationState()`	Status of the script. The possible values are `Completed` or `Failed`.
`getHostOutput()`	Output of the script as it appears on the PowerShell console.
`getResults()`	Objects returned by the PowerShell engine. The returned object is of type `PowershellRemotePSObject`.

`PowershellRemotePSObject` is a remote representation of objects returned by the PowerShell engine. `PowershellRemotePSObject` contains XML serialization of the result that can be accessed by calling the `getXml()` method.

The PowerShell plug-in also provides an object model that wraps the XML result and provides easier access to particular object properties. The `getRootObject()` method provides access to the object model. In general, the `getRootObject()` method maps the PowerShell types to types available in Orchestrator, by using the following rules.

- If the returned object is of a primitive PowerShell type, the object is mapped to the corresponding Orchestrator primitive type.
- If the returned object is of type `collection`, the object is represented as `ArrayList`.
- If the returned object is of type `dictionary`, the object is represented as `Hashtable`.
- If the returned object is of type `complex`, the object is represented as `PSObject`.

Examples of Scripts for Common PowerShell Tasks

You can cut, paste, and edit the JavaScript examples to write scripts for common PowerShell tasks.

For more information about scripting, see the *vRealize Orchestrator Developer's Guide*.

Example: Run a PowerShell Script Through the API

You can use JavaScript to run a PowerShell script through the plug-in API.

This example script performs the following actions.

- Opens a session to a PowerShell host.
- Provides a script to run.
- Checks invocation results.
- Closes the session.

```
var sess;
try {
     //Open session to PowerShell host
     var sess = host.openSession()
     //Set executed script
     var result = sess.invokeScript('dir')

     //Check for errors
     if (result.invocationState == 'Failed'){
          throw "PowerShellInvocationError: Errors found while executing script \n" +
result.getErrors();
     }
     //Show result
     System.log( result.getHostOutput() );
} catch (ex){
     System.error (ex)
} finally {
     if (sess) {
     //Close session
     host.closeSession( sess.getSessionId() );
     }
}
```

Example: Work with Result

You can use JavaScript to work with the result of a PowerShell script run.

This example script performs the following actions.

- Checks the invocation state.
- Extracts a value from the result.

- Checks the `RemotePSObject` type.

```
var sess = host.openSession()
sess.addCommandFromString("dir " + directory)
var invResult = sess.invokePipeline();
//Show result
System.log( invResult.getHostOutput() );

//Check for errors
if (invResult.invocationState == 'Failed'){
System.error(invResult.getErrors());
     } else {
     //Get PowerShellRemotePSObject
     var psObject = invResult.getResults();
     var directories = psObject.getRootObject();

     var isList = directories instanceof Array
     if ( isList ){
          for (idx in directories){
               var item = directories[idx];
               if ( item.instanceOf('System.IO.FileInfo') ){//Check type of object
                    System.log( item.getProperty('FullName') );//Extract value from result
               }
          }
     } else {
               System.log( directories.getProperty('FullName') );//Extract value from result
     }
}

host.closeSession( sess.getSessionId());
```

Example: Connect with Custom Credentials

You can use JavaScript to connect to a PowerShell host with custom credentials.

```
var sess;
try {
     sess = host.openSessionAs(userName, password);

     var invResult = sess.invokeScript('$env:username');

     //Check for errors
     if (invResult.invocationState == 'Failed'){
               System.error(invResult.getErrors());
     } else {
               //Show result
               System.log( invResult.getHostOutput() );
     }
} catch (ex){
     System.error (ex)
} finally {
     if (sess) {
                host.closeSession( sess.getSessionId());
```

```
        }
}
```

Troubleshooting

If you encounter problems when using the PowerShell plug-in, you can refer to a troubleshooting topic to understand the problem or solve it, if there is a workaround.

Enable Kerberos Event Logging

For troubleshooting purposes, you might want to enable Kerberos event logging on the Key Distribution Center (KDC) machine.

Prerequisites

Back up the Windows registry.

Procedure

1 Log in to the domain controller that acts as a Key Distribution Center (KDC).

2 Run the registry editor as an **administrator**.

3 In the registry window, expand `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Kerberos\Parameters`.

4 If a **`LogLevel`** registry key value does not exist, right-click to create it.

   a Right-click **Parameter**, select **New > DWORD (32-bit) Value**, and enter `LogLevel`.

   b Select **Parameter** and in the right pane, double-click `LogLevel` and enter **`1`** in the **Value data:** text box.

   The new setting becomes effective without a reboot on Windows Server 2003 and later.

Results

The Kerberos error event entries are recorded in the System Windows Event Log.

What to do next

To disable Kerberos event logging, delete the **`LogLevel`** registry key value or change its value data to **`0`**.

Servers Not Found in Kerberos Database

After you add servers with Kerberos authentication, the servers might not be found because they are not added correctly.

Problem

When you try to connect to a server, the server is not found in Kerberos database.

```
No valid credentials provided (Mechanism level: No valid
credentials provided (Mechanism level: Server not found in Kerberos
database (7)))
```

Cause

This error might be caused by several misconfigurations.

- The PowerShell host is not part of a domain.
- The host to realm mapping is not correct.
- The Service Principal Name of the PowerShell host is not built correctly.

Note Kerberos authentication does not work when the destination is an IP address.

Solution

When you add a PowerShell host using the Kerberos authentication, enter a DNS or NetBIOS destination.

Unable to Obtain a Kerberos Ticket

When you provide wrong credentials, the plug-in fails to obtain a Kerberos ticket.

Problem

You are unable to add a host to the plug-in inventory and the result is the following error message.

```
Pre-authentication information was invalid (24)
```

Cause

You have provided wrong credentials.

Solution

Provide the correct credentials.

Kerberos Authentication Fails Due to Different Time Settings

Inconsistent time settings in the environment that uses Kerberos configuration might lead to authentication failure.

Problem

Attempts to use Kerberos for initial authentication of a host or for resource access fail, and the following error message appears.

```
Clock Skew
```

Cause

If the system time on the computers in the environment differs with more than 5 minutes from the domain controller, or from one another, the Kerberos authentication fails.

Solution

Synchronize the system times in the environment.

Kerberos Authentication Session Mode Fails

When you use Kerberos authentication with Shared Session or Session per User, adding the PowerShell host might fail.

Problem

When you attempt to add a PowerShell host to the plug-in inventory using Shared Session or Session per User, the workflow fails with the following error.

`Null realm name (601) – default realm not specified (Dynamic Script Module name : addPowerShellHost#16)`

Cause

The default realm is not specified in the Kerberos configuration file `krb5.conf`, neither is provided as a part of the user name.

Solution

Provide a default realm in your Kerberos configuration file or include the realm in your user name when authenticating with Kerberos.

Unable to Reach a Key Distribution Center for a Realm

Any misspelling in the `krb5.conf` file might cause a failure when you add a host.

Problem

When you are adding a host, the Kerberos authentication is unable to reach a Key Distribution Center (KDC) for *yourrealm*.

```
Cannot get kdc for realm YOURREALM.COM
```

Cause

The `libdefaults` and `realms` sections in the `krb5.conf` file might be misspelled.

Solution

Verify that the `libdefaults` and `realms` sections in your `krb5.conf` file are spelled correctly.

Unable to Locate the Default Realm

Orchestrator workflows that require Kerberos authentication might fail if the Kerberos configuration file does not have the correct format or encoding.

Problem

Kerberos authentication cannot identify the default realm.

```
Cannot locate default realm
```

Cause

The Kerberos configuration file `krb5.conf` that you upload to the vRealize Orchestrator Appliance has been edited on a non-UNIX operating system. As a result, the format and the encoding might be incorrect.

Solution

In order for the Orchestrator appliance to read the `krb5.conf` file, the format of the file must be UNIX and the character encoding must be ANSI as UTF-8.

Using the Multi-Node Plug-In

21

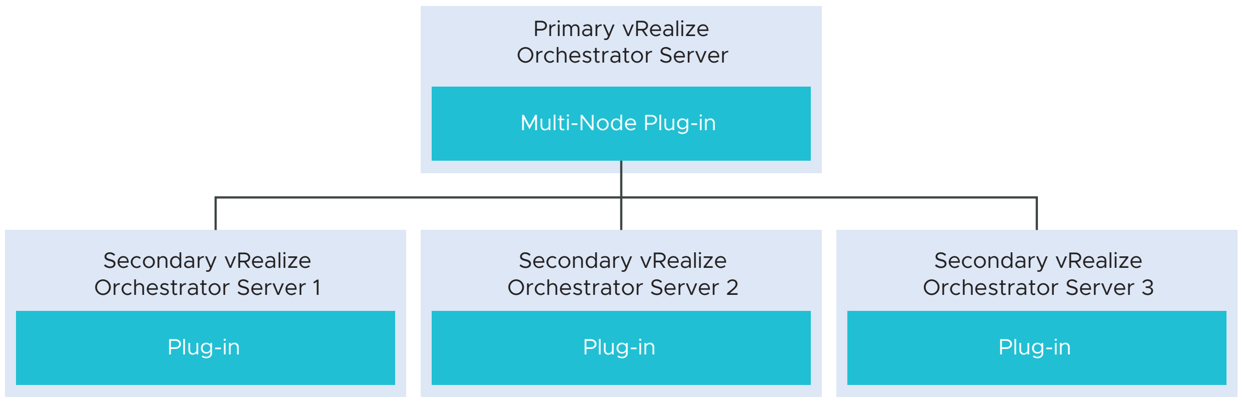
The Multi-Node plug-in workflow library contains workflows for hierarchical orchestration, management of Orchestrator instances, and scale-out of Orchestrator activities.

This chapter includes the following topics:

Introduction to the vRealize Orchestrator Multi-Node Plug-In

The Multi-Node plug-in creates a primary-secondary relation between vRealize Orchestrator servers, which extends in the areas of package management and workflow execution.

Figure 21-1. Multi-Node Plug-In Schema


Primary vRealize Orchestrator Server
Multi-Node Plug-in
Secondary vRealize Orchestrator Server 1
Plug-in
Secondary vRealize Orchestrator Server 2
Plug-in
Secondary vRealize Orchestrator Server 3
Plug-in

The plug-in contains a set of standard workflows for hierarchical orchestration, management of vRealize Orchestrator instances, and scale-out of vRealize Orchestrator activities.

Configuring the Multi-Node Plug-In

You must use the Orchestrator client to configure the Multi-Node plug-in.

Servers Configuration Workflows

The Servers Configuration workflow category contains workflows that allow you to configure the connected Orchestrator servers.

You can access these workflows from **Library > Orchestrator > Servers Configuration** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Add an Orchestrator server	Adds an Orchestrator server to the plug-in inventory.
Delete an Orchestrator server	Removes an Orchestrator server from the plug-in inventory and deletes all created proxies for this server.
Update an Orchestrator server	Updates an Orchestrator server from the plug-in inventory by changing its details.

Add an Orchestrator Server

You can run a workflow to establish a connection to a new vRealize Orchestrator server.

Prerequisites

Verify that the primary and secondary Orchestrator servers are the same version.

Procedure

1. Log in to the Orchestrator client as an administrator.
2. Click the **Workflows** view in the Orchestrator client.
3. In the hierarchical list of workflows, expand **Library > Orchestrator > Servers Configuration** and navigate to the Add an Orchestrator server workflow.
4. Right-click the Add an Orchestrator server workflow and select **Start workflow**.
5. Provide the new server details.
6. Select whether the connection is shared.

Option	Description
No	The credentials of the logged-in user are used to connect to the remote Orchestrator server.
Yes	All users can access the remote Orchestrator server using the same credentials. Provide the credentials for the shared connection.

7. Click **Submit** to run the workflow.

Using Proxy Workflows

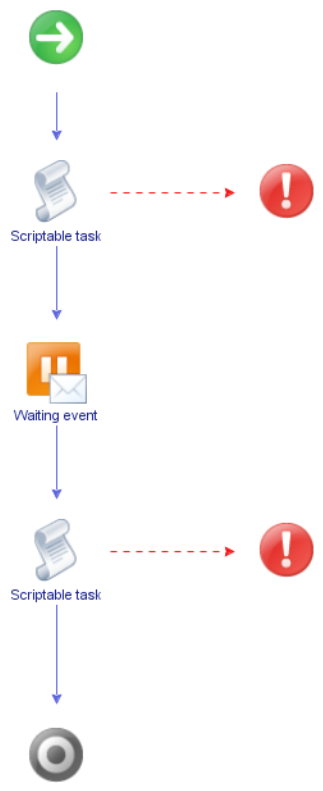
You can use proxy workflows to manage the interaction between the local Orchestrator server and workflows on a remote Orchestrator server.

You can use the Multi-Node plug-in to generate local workflows which interact with remote workflows. These local workflows are called proxy workflows. A proxy workflow takes the input parameters from the inventory of the Multi-Node plug-in. When you run the proxy workflow, it converts the parameters to the types required by the remote workflow. When the remote workflow finishes its run, the output parameters are converted back to the local representation on the primary Orchestrator server.

Synchronous Proxy Workflows

The synchronous type of proxy workflows preserve the API and the operation contract of the remote workflows.

The schema of all synchronous proxy workflows is the same, but contains different scripting.



The synchronous proxy workflow completes the run after the remote workflow completes and provides output parameters.

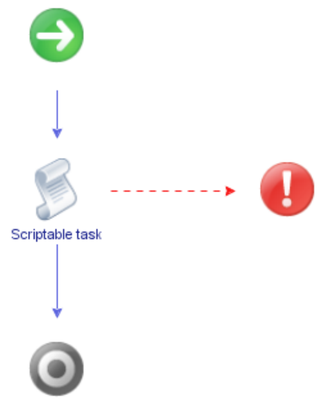
The local workflow consumes no server resources while waiting for the results of the remote workflow.

At the end of a successful run the output parameters of the proxy workflow contain a local representation of the remote workflow token. The output parameters can be used directly by other workflows on the local Orchestrator server when they are of simple type, such as, boolean, number, string, and similar.

Asynchronous Proxy Workflows

You can use asynchronous proxy workflows to optimize the run of remote workflows.

The schema of all asynchronous proxy workflows is the same, but contains different scripting.



An asynchronous proxy workflow returns immediately a result that is a local wrapper of the remote workflow token object. The proxy workflow uses this token to check the state of the run and to retrieve the output parameters when the remote workflow completes its run. The output parameters can be used directly by other workflows on the local Orchestrator server when they are of simple type, such as, boolean, number, string, and similar.

Remote Execution Workflows

The Remote Execution workflow category contains workflows that allow you to manage proxy workflows.

Remote Execution Standart Workflows

You can access the workflows to create proxy workflows from **Library > Orchestrator > Remote Execution** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Create a multi proxy action	Creates a multi-proxy action to run workflows on multiple servers.
Create a proxy workflow	Creates a proxy workflow that you can use to start a workflow on a remote Orchestrator server.
Create proxy workflows from a folder	Creates proxy workflows for all workflows in a folder on the remote Orchestrator server.

Server Proxies

You can access the workflows for managing server proxies from **Library > Orchestrator > Remote Execution > Server Proxies** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Create proxy workflows for an Orchestrator server	Creates proxy workflows on the local Orchestrator server by mirroring the remote server's structure.
Delete proxy workflows for an Orchestrator server	Removes the proxy workflows for the local Orchestrator server and deletes all generated workflows.
Refresh proxy workflows for an Orchestrator server	Regenerates all proxy workflows for the local Orchestrator server from the remote server.

Using the Multi-Node Plug-In Inventory

The Multi-Node plug-in mirrors all inventories of the connected vRealize Orchestrator servers in the **Inventory** view. You can use the **Inventory** view to add authorization elements or to run workflows on remote Orchestrator servers.

To display the workflows that are available for an inventory object, navigate to **Tools > User preferences > Inventory** and select the **Use contextual menu in inventory** check box. After the option is enabled, when you right-click an object in the Orchestrator inventory, all available workflows for the object are displayed.

The inventory for a single remote server consist of two major parts, system objects and plug-in objects. Both objects are wrappers of the remote objects into locally usable types:

System object

System objects are under a top-level group called **System**.They contain configurations, packages, workflows, actions, and related folders. Remote system objects have individual wrapper types.

Plug-in objects

Plug-in objects mirror the inventories of all plug-ins attached to the remote Orchestrator server. Remote plug-in objects are all wrapped into a single local type **VCO:RemotePluginObject**.

Remote Management Workflows

The Remote Management workflow category contains workflows that allow you to manage packages and workflows on remote Orchestrator instances.

Packages

You can access the workflows for managing remote packages from **Library > Orchestrator > Remote Management > Packages** on the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Delete a package	Deletes a package and its contents from a remote Orchestrator server.
Delete a package by name	Deletes a package and its contents by name on a remote Orchestrator server.
Deploy a package from a local server	Deploys a package from a local Orchestrator server to remote Orchestrator servers.
Deploy a package from a remote server	Deploys a package from one remote Orchestrator server to a list of remote Orchestrator servers.
Deploy packages from a local server	Deploys packages from a local Orchestrator server to remote Orchestrator servers.

Workflows

You can access the workflows for managing remote workflows from **Library > Orchestrator > Remote Management > Workflows** on the **Workflows** view in the Orchestrator client.

Workflow Name	
Delete a remote workflow	Deletes a workflow from a remote Orchestrator server.
Delete all finished workflow runs	Deletes all finished workflow runs from a remote workflow.
Deploy a workflow from a local server	Deploys a workflow from a local Orchestrator server to a list of remote Orchestrator servers.
Deploy a workflow from a remote server	Deploys a workflow from a remote Orchestrator server to a list of other remote Orchestrator servers.

Access the Multi-Node Plug-In API

Orchestrator provides an API Explorer to allow you to search the Multi-Node plug-in API and see the documentation for JavaScript objects that you can use in scripted elements.

Procedure

1 Log in to the Orchestrator client as an administrator.

2 Access the API Explorer from either the Orchestrator client or from the **Scripting** tabs of the workflow, policy, and action editors.

- To access the API Explorer from the Orchestrator client, click **Tools > API Explorer** in the Orchestrator client toolbar.
- To access the API Explorer from the **Scripting** tabs of the workflow, policy, and action editors, click **Search API** on the left.

3 To expand the hierarchical list of Multi-Node plug-in API objects, double-click the **VCO** module in the left pane.

What to do next

You can copy code from API elements and paste it into scripting boxes. For more information about API scripting, see *Developing with VMware vRealize Orchestrator*.

Multi-Node Plug-In Use Cases

The Multi-Node plug-in use cases include user scenarios such as importing a package from the local Orchestrator server to the remote servers, using multi proxy actions, as well as information about maintenance of remote and proxy workflows.

Create a Multi-Proxy Action

You can run the Create a multi-proxy action workflow to run a workflow on several servers.

You can create an action, so that you can run a workflow on a remote Orchestrator server at a later stage.

Procedure

1 Log in to the Orchestrator client as an administrator.

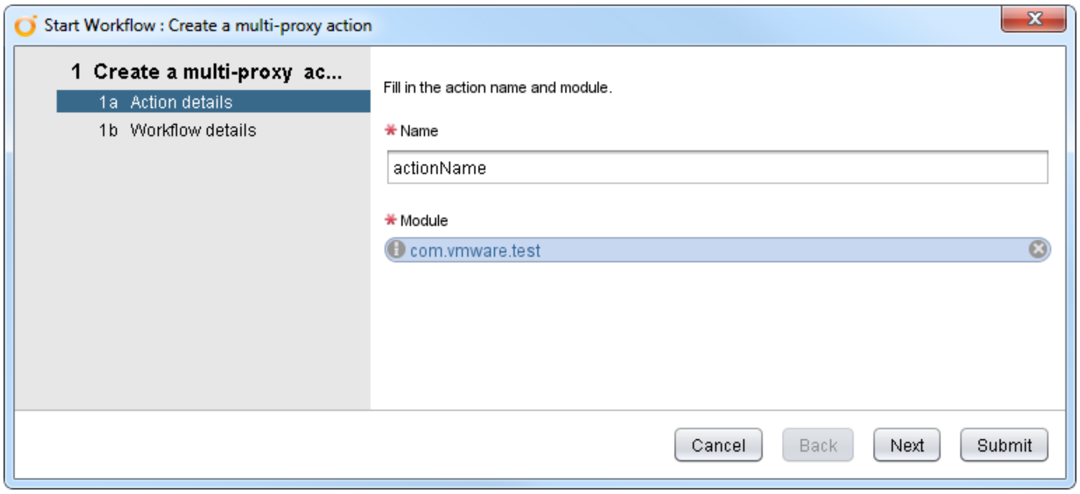
2 Click the **Workflows** view in the Orchestrator client.

3 In the hierarchical list of workflows, expand **Library > Orchestrator > Remote Execution** and navigate to the Create a multi-proxy action workflow.

4 Right-click the Create a multi-proxy action workflow and select **Start workflow**.

5 In the **Action name** text box, type the name of the action.

The action name must contain only alpha-numeric characters without spaces.


Start Workflow : Create a multi-proxy action
1 Create a multi-proxy ac...
1a Action details
1b Workflow details
Cancel
Back
Next
Submit

Fill in the action name and module.

* Name

actionName

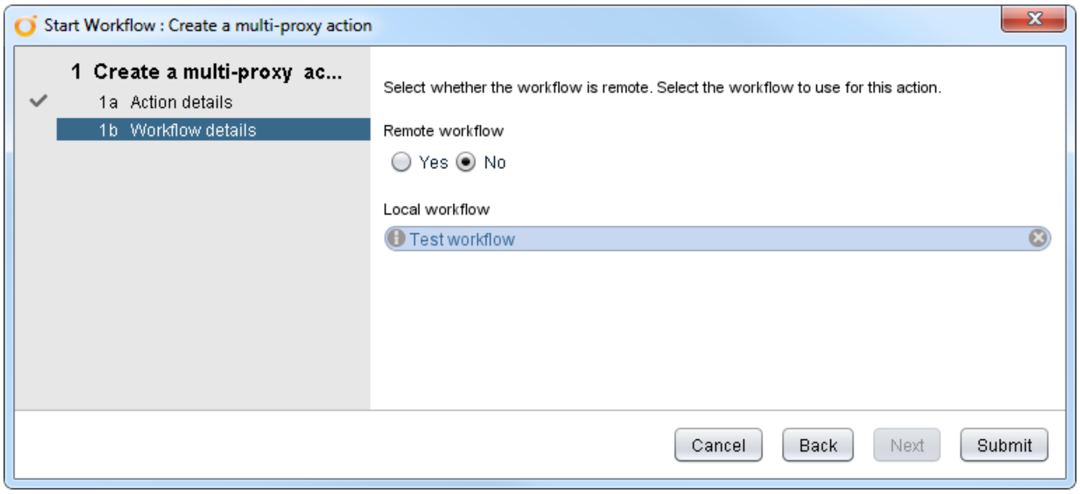
* Module

com.vmware.test

A new action is created even if another action with the same name exists.

6 Select a module in which to add the action.

7 Select whether the workflow is local or remote.



Select whether the workflow is remote. Select the workflow to use for this action.

Remote workflow: Yes [✓ No]

Local workflow: Test workflow

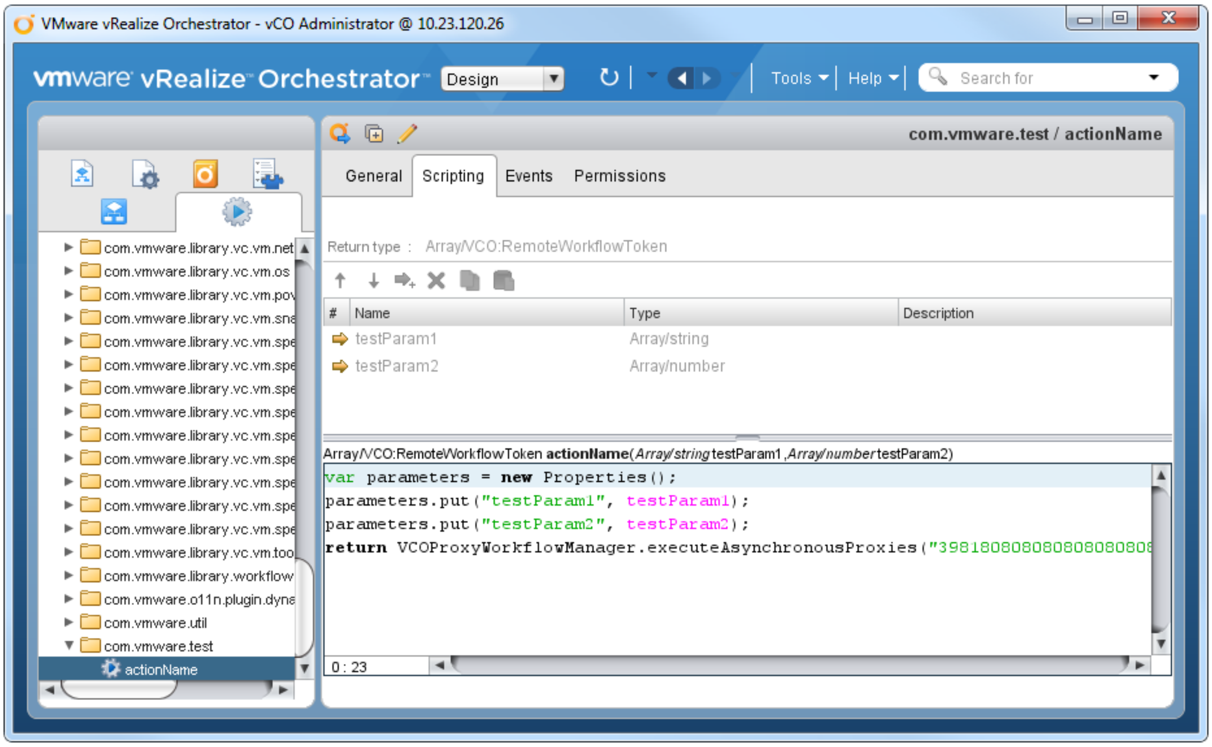
Start Workflow : Create a multi-proxy action
1 Create a multi-proxy ac...
1a Action details
1b Workflow details
Cancel
Back
Next
Submit

8 Select the workflow that you want to use for this action.

9 Click **Submit** to run the workflow.

Results

The generated action accepts the same parameters as the source workflow but promotes the parameters to an array in case of multi-selection of objects. The values in the array are indexed.



Return type : Array/VCO:RemoteWorkflowToken

#	Name	Type	Description
	testParam1	Array/string	
	testParam2	Array/number	

Array/VCO:RemoteWorkflowToken **actionName**(*Array/string* testParam1 ,*Array/number* testParam2)

```
var parameters = new Properties();
parameters.put("testParam1", testParam1);
parameters.put("testParam2", testParam2);
return VCOProxyWorkflowManager.executeAsynchronousProxies("39818080808080808080808
```

VMware vRealize Orchestrator - vCO Administrator @ 10.23.120.26
vmware vRealize Orchestrator
Design
Tools
Help
Search for
com.vmware.test / actionName
General
Scripting
Events
Permissions
com.vmware.library.vc.vm.net
com.vmware.library.vc.vm.os
com.vmware.library.vc.vm.pov
com.vmware.library.vc.vm.sna
com.vmware.library.vc.vm.spe
com.vmware.library.vc.vm.spe
com.vmware.library.vc.vm.spe
com.vmware.library.vc.vm.spe
com.vmware.library.vc.vm.spe
com.vmware.library.vc.vm.spe
com.vmware.library.vc.vm.spe
com.vmware.library.vc.vm.spe
com.vmware.library.vc.vm.spe
com.vmware.library.vc.vm.too
com.vmware.library.workflow
com.vmware.o11n.plugin.dyna
com.vmware.util
com.vmware.test
actionName
0 : 23

Maintenance of Remote and Proxy Workflows

If the remote and proxy workflows change, you might want to update the proxies or to delete them if you do not need them anymore. For maintenance purposes, the Multi-Node plug-in provides workflows that allow you to update or delete proxy and remote workflow information.

You can access the workflows for managing the proxy workflows from **Library > Orchestrator > Remote Execution > Server Proxies** in the **Workflows** view in the Orchestrator client.

Workflow Name	Description
Refresh proxy workflows for an Orchestrator server	Regenerates all proxy workflows for the local Orchestrator server from the remote server.
Delete proxy workflows for an Orchestrator server	Removes the proxy workflows for the local Orchestrator server and deletes all generated workflows.

You can access wokflows for further maintenance of the proxy workflows from **Library > Orchestrator > Remote Management > Workflows** in the **Workflows** view of the Orchestrator client.

Workflow Name	Description
Delete all finished workflow runs	Deletes all finished workflow runs from a remote workflow.
Delete a remote workflow	Deletes a workflow from a remote Orchestrator server.
Deploy a workflow from a local server	Deploys a workflow from a local Orchestrator server to a list of remote Orchestrator servers.

Deploy a Package from a Local Server

You can run a workflow to deploy a package from a local Orchestrator server to remote Orchestrator servers.

In this example, you can deploy a package from a local server to an array of remote servers.

Procedure

1. Log in to the Orchestrator client as an administrator.
2. Click the **Workflows** view in the Orchestrator client.
3. In the hierarchical list of workflows, expand **Library > Orchestrator > Remote Management** and navigate to the Deploy a package from a local server workflow.
4. Right-click the Deploy a package from a local server and select **Start workflow**.
5. Select the package to deploy from the local storage.
6. Select the remote servers to deploy the package to.

7 Select whether you want to overwrite the remote server packages.

Option	Description
Yes	The packages on the remote server are replaced, discarding the version of the packaged elements.
No	A version check of the server and the deploying packages is performed. The packages are deployed after a successful check.

8 Click **Submit** to run the workflow.

Results

After running the workflow, the status information is displayed in the log view and in the inventory of the plug-in.

Using the vCloud Suite API (vAPI) Plug-In

22

The vCloud Suite API plug-in provides the ability to consume API exposed by any vCloud Suite API provider. The vCloud Suite API provides a service-oriented architecture for accessing resources in the virtual environment by issuing requests to vCenter Server, through the vCloud Suite Endpoint.

The plug-in contains a set of standard workflows and example workflows. You can also create custom workflows that implement the plug-in to automate tasks in your virtual environment. For information about vCloud Suite API, see *VMware vCloud Suite SDKs Programming Guide*.

This chapter includes the following topics:

Configuring the vCloud Suite API Plug-In

You can configure vCloud Suite API by running the configuration workflows included in the plug-in.

Import a vCloud Suite API Metamodel

The vCloud Suite API plug-in discovers vCloud Suite API services dynamically by querying a vCloud Suite API provider metadata service. vCloud Suite API providers which are not exposing metadata service are not supported.

You must import a vCloud Suite API metamodel and add endpoints afterwards.

Procedure

1 Log in to the Orchestrator client as an administrator.

2 Click the **Workflows** view in the Orchestrator client.

3 In the workflows hierarchical list, expand **Library > VAPI** and navigate to the **Import vAPI metamodel** workflow.

4 Right-click the **Import vAPI metamodel** workflow and select **Start workflow**.

5 In the **vAPI endpoint URL** text box, type the URL of your vCloud Suite API endpoint.

6 Choose whether to use a secure protocol connection:

Option	Description
No	Import the vCloud Suite API metamodel, without using a secure protocol connection.
Yes	To import the vCloud Suite API metamodel with secure protocol connection: a Choose whether to ignore certificate warnings and accept the vCloud Suite endpoint automatically. b Provide the user credentials to authenticate with the vCloud Suite endpoint.

7 Click **Submit** to run the workflow.

What to do next

Add a vCloud Suite API Endpoint

Add a vCloud Suite API Endpoint

Add a vCloud Suite API endpoint.

Prerequisites

Import a vCloud Suite API metamodel.

Procedure

1 Log in to the Orchestrator client as an administrator.

2 Click the **Workflows** view in the Orchestrator client.

3 In the workflows hierarchical list, expand **Library > VAPI** and navigate to the **Add vAPI endpoint** workflow.

4 Right-click the **Add vAPI endpoint** workflow and select **Start workflow**.

5 In the **vAPI endpoint URL** text box, type the URL of your vCloud Suite API endpoint.

6 Choose whether to use a secure protocol connection:

Option	Description
No	Import the vCloud Suite API metamodel, without using a secure protocol connection.
Yes	To import the vCloud Suite API metamodel with secure protocol connection: a Choose whether to ignore certificate warnings and accept the vCloud Suite endpoint automatically. b Provide the user credentials to authenticate with the vCloud Suite endpoint.

7 Click **Submit** to run the workflow.

Access the vCloud Suite API Plug-In API

Orchestrator provides an API Explorer to allow you to search the vCloud Suite API plug-in API and see the documentation for JavaScript objects that you can use in scripted elements.

Procedure

1. Log in to the Orchestrator client as an administrator.
2. Access the API Explorer from either the Orchestrator client or from the **Scripting** tabs of the workflow, policy, and action editors.
   - To access the API Explorer from the Orchestrator client, click **Tools > API Explorer** in the Orchestrator client toolbar.
   - To access the API Explorer from the **Scripting** tabs of the workflow, policy, and action editors, click **Search API** on the left.
3. To expand the hierarchical list of vCloud Suite API plug-in API objects, double-click the **VAPI** module in the left pane.

What to do next

You can copy code from API elements and paste it into scripting boxes. For more information about API scripting, see *Developing with VMware vRealize Orchestrator*.