Using VMware vCenter Orchestrator Plug-Ins

vCenter Orchestrator 5.1

This document supports the version of each product listed and supports all subsequent versions until the document is replaced by a new edition. To check for more recent editions of this document, see http://www.vmware.com/support/pubs.
You can find the most up-to-date technical documentation on the VMware Web site at:
http://www.vmware.com/support/
The VMware Web site also provides the latest product updates.
If you have comments about this documentation, submit your feedback to:
docfeedback@vmware.com
Contents

Using VMware vCenter Orchestrator Plug-Ins  7
Updated Information  9

1 Introduction to Orchestrator Plug-Ins  11
   Orchestrator Architecture  11
   Default Orchestrator Plug-Ins  12
   Access the Orchestrator API Explorer  14

2 Using the vCenter Server 5.1 Plug-In  17
   Configure the vCenter Server 5.1 Plug-In  17
   vCenter Server 5.1 Plug-In Scripting API  18
   Using the vCenter Server 5.1 Plug-In Inventory  18
   Access the vCenter Server 5.1 Plug-In Workflow Library  18
   vCenter Server 5.1 Plug-In Workflow Library  19
      Batch Workflows  19
      Cluster and Compute Resource Workflows  19
      Guest Operation Files Workflows  20
      Guest Operation Processes Workflows  21
      Custom Attributes Workflows  21
      Datacenter Workflows  21
      Datastore and Files Workflows  22
      Datacenter Folder Management Workflows  22
      Host Folder Management Workflows  22
      Virtual Machine Folder Management Workflows  23
      Basic Host Management Workflows  23
      Power Host Management Workflows  23
      Host Registration Management Workflows  24
      Networking Workflows  24
      Distributed Virtual Port Group Workflows  24
      Distributed Virtual Switch Workflows  25
      Standard Virtual Switch Workflows  25
      Resource Pool Workflows  26
      Storage Workflows  26
      Storage DRS Workflows  27
      Basic Virtual Machine Management Workflows  28
      Clone Workflows  29
      Linked Clone Workflows  29
      Linux Customization Clone Workflows  30
      Tools Clone Workflows  30
      Windows Customization Clone Workflows  31
      Device Management Workflows  31
Using VMware vCenter Orchestrator Plug-Ins

Move and Migrate Workflows 32
Other Workflows 32
Power Management Workflows 33
Snapshot Workflows 33
VMware Tools Workflows 34

3 Using the vCO Library Plug-In 35
vCO Library Plug-In Workflows 35

4 Using the SQL Plug-In 37
Configuring the SQL Plug-In 37
SQL Plug-In Configuration Workflows 37
Add a Database 38
Add Tables to a Database 38
Update a Database 39
SQL Plug-In Scripting API 39
SQL:ActiveRecord Type 39
SQL:Column Type 40
SQL:Database Type 40
SQL:Table Type 40
Connection Class 40
JDBCConnection Class 41
PreparedStatement Class 41
ResultSet Class 41
ResultSetMetaData Class 42
SQLActiveRecord Class 42
SQLColumn Class 43
SQLDatabase Class 43
SQLDatabaseManager Class 44
SQLTable Class 45
Running the SQL Sample Workflows 45
Generate a JDBC URL 45
Test a JDBC Connection 46
Create a Table by Using JDBC 46
Insert a Row into a JDBC Table 47
Select Rows from a JDBC Table 47
Delete an Entry from a JDBC Table 48
Delete All Entries from a JDBC Table 48
Drop a JDBC Table 49
Run a Complete JDBC Cycle 49
Using the SQL Plug-In Standard Workflows 50
SQL Plug-In Workflow Library 50
Generate CRUD Workflows for a Table 50

5 Using the SSH Plug-In 53
Configure the SSH Plug-In 53
SSH Plug-In Scripting API 54
SSH:File Type 54
9 Using the Enumeration Plug-In 79
   Enumeration Plug-In Scripting API 79
   Time Zone Codes 80

10 Using the Workflow Documentation Plug-In 83
   Workflow Documentation Plug-In Scripting API 83
   WorkflowDocumentationService Class 83
   Workflow Library for the Workflow Documentation Plug-In 83
   Generate Workflow Documentation 84

Index 85
Using VMware vCenter Orchestrator Plug-Ins

Using VMware vCenter Orchestrator Plug-Ins provides information and instructions about configuring and using the official set of plug-ins installed with VMware vCenter® 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.
**Updated Information**

*Using VMware vCenter Orchestrator Plug-Ins* is updated with each release of the product or when necessary. This table provides the update history of *Using VMware vCenter Orchestrator Plug-Ins.*

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000735 -01</td>
<td>- Updated Chapter 4, “Using the SQL Plug-In,” on page 37 with information about configuring the SQL plug-in, adding and updating databases.</td>
</tr>
<tr>
<td></td>
<td>- Updated the “SQL Plug-In Scripting API” on page 39 section to include all of the API types and classes.</td>
</tr>
<tr>
<td></td>
<td>- Included information about using the SQL plug-in standard workflows. For more details, see “Using the SQL Plug-In Standard Workflows,” on page 50.</td>
</tr>
<tr>
<td></td>
<td>- Minor updates.</td>
</tr>
<tr>
<td>000735 -00</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>
Introduction to Orchestrator Plug-Ins

Plug-ins allow you to use Orchestrator to access and control external technologies and applications. Exposing an external technology in an Orchestrator plug-in allows you to incorporate objects and functions in workflows that access the objects and functions of that external technology.

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

Orchestrator provides a set of standard plug-ins to allow you to incorporate such technologies as the VMware vCenter Server API and email capabilities into workflows. In addition, the Orchestrator open plug-in architecture allows 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 vCenter Orchestrator.

All default plug-ins are installed together with the Orchestrator server. Before you use some of the plug-ins, for example the vCenter Server plug-in, you should configure 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 trigger events in Orchestrator and in the plugged-in technology. Plug-ins provide an inventory of JavaScript objects that you can access on the Orchestrator Inventory tab. Each plug-in can provide one or more 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,” on page 11
- “Default Orchestrator Plug-Ins,” on page 12
- “Access the Orchestrator API Explorer,” on page 14

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, to allow you to orchestrate tasks in the different environments that the plug-ins expose.

Orchestrator also presents an open architecture to allow you to plug 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 a directory services server to manage user accounts, and to a database to store information from the workflows that it runs. You can access Orchestrator, the Orchestrator workflows, and the objects it exposes through the Orchestrator client interface, through a Web browser, or through Web services.
Default Orchestrator Plug-Ins

Orchestrator includes a collection of default plug-ins. Each plug-in exposes an external product API to the Orchestrator platform. Plug-ins provide inventory classes, extend the scripting engine with new object types, and publish notification events from the external system. Each plug-in can also provide a library of workflows that represents the typical use cases of the integrated product in an automated fashion.

You can see the list of available plug-ins from the Plug-ins tab in the Orchestrator configuration interface. Plug-ins that require configuration add separate tabs to the interface.

### Table 1-1. Plug-Ins Installed with Orchestrator by Default

<table>
<thead>
<tr>
<th>Plug-In</th>
<th>Purpose</th>
<th>Configuration</th>
<th>Input Types</th>
<th>Scripting Objects</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCO Library</td>
<td>Provides workflows that act as basic building blocks for customization and automation of client processes. The workflow library includes templates for lifecycle management, provisioning, disaster recovery, hot backup, and many other standard processes. Sources of library processes are accessible and can be copied and edited.</td>
<td>None</td>
<td>See Developing with VMware vCenter Orchestrator, Using the Orchestrator API.</td>
<td>Exposes nothing in the inventory.</td>
<td></td>
</tr>
</tbody>
</table>

Using VMware vCenter Orchestrator Plug-Ins

12 VMware, Inc.
<table>
<thead>
<tr>
<th>Plug-In</th>
<th>Purpose</th>
<th>Configuration</th>
<th>Input Types</th>
<th>Scripting Objects</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>Java Database Connectivity (JDBC) API 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.</td>
<td>None</td>
<td>ActiveRecord Column Database Table</td>
<td>Connection JDBCConnection PreparedStatement ResultSet ResultSetMetaData SQLActiveRecord SQLColumn SQLDatabase SQLDatabaseManager SQLTable</td>
<td>Exposes the databases you add in the inventory.</td>
</tr>
<tr>
<td>SSH</td>
<td>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. Optionally, the SSH plug-in can provide remote file system browsing directly in the vCO inventory.</td>
<td>See “Configure the SSH Plug-In,” on page 53.</td>
<td>File Folder RootFolder SshConnection</td>
<td>KeyPairManager SSHCommand SSHPublicKey SSHPublicKeyFile SSHPublicKeyFolder SSHPublicKeySession</td>
<td>Can expose objects in the inventory.</td>
</tr>
<tr>
<td>XML</td>
<td>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.</td>
<td>None</td>
<td>None</td>
<td>XMLDocument XMLHttpRequest XElement XMLManager XMLNameNodeMap XMLNode XMLNodeList</td>
<td>Exposes nothing in the inventory.</td>
</tr>
<tr>
<td>Mail</td>
<td>Uses Simple Mail Transfer Protocol (SMTP) to send email from workflows.</td>
<td>Set the default values for the EmailMessage object to use. See “Define the Default SMTP Connection,” on page 71.</td>
<td>None</td>
<td>EmailMessage</td>
<td>Exposes nothing in the inventory.</td>
</tr>
<tr>
<td>Net</td>
<td>Wraps the Jakarta Apache Commons Net Library. Provides implementations of Telnet, FTP, and POP3. The POP3 part allows reading email. In combination with the Mail plug-in, the Net plug-in provides full email sending and receiving capabilities in workflows.</td>
<td>None</td>
<td>None</td>
<td>FTPClient POP3Client POP3Message TelnetClient</td>
<td>Exposes nothing in the inventory.</td>
</tr>
<tr>
<td>Workflow documentation</td>
<td>Provides workflows that allow you to generate information in PDF format about a workflow or a workflow category.</td>
<td>None</td>
<td>None</td>
<td>workflowDocumentation workflowDocumentationService</td>
<td>Exposes nothing in the inventory.</td>
</tr>
</tbody>
</table>
Table 1-1. Plug-Ins Installed with Orchestrator by Default (Continued)

<table>
<thead>
<tr>
<th>Plug-In</th>
<th>Purpose</th>
<th>Configuration</th>
<th>Input Types</th>
<th>Scripting Objects</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration</td>
<td>Provides common enumerated types that can be used in workflows by other plug-ins.</td>
<td>None</td>
<td>JavaScript Enumerations</td>
<td>None</td>
<td>Exposes nothing in the inventory.</td>
</tr>
<tr>
<td>vCO WebOperator</td>
<td>A Web view that allows you to access the workflows in the Orchestrator library and interact with them across a network by using a Web browser.</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Exposes nothing in the inventory.</td>
</tr>
</tbody>
</table>

Plug-In Components

Each plug-in is a DAR file package. The DAR files are stored in Orchestrator_installation_path\app-server\server\vmo\plugins on the Orchestrator server system. The components of each plug-in, such as workflow categories and API modules, can use different naming conventions.

Table 1-2. Names of Plug-In Components

<table>
<thead>
<tr>
<th>Plug-In Name in the Configuration UI</th>
<th>DAR Name</th>
<th>Workflow Categories</th>
<th>API Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCenter Server 5.1</td>
<td>o11nplugin-vsphere51.dar</td>
<td>vCenter</td>
<td>VC</td>
</tr>
<tr>
<td>vCO Library</td>
<td>o11nplugin-library.dar</td>
<td>Locking</td>
<td>Orchestrator Troubleshooting</td>
</tr>
<tr>
<td>SQL</td>
<td>o11nplugin-database.dar</td>
<td>JDBC</td>
<td>SQL</td>
</tr>
<tr>
<td>SSH</td>
<td>o11nplugin-ssh.dar</td>
<td>SSH</td>
<td>SSH</td>
</tr>
<tr>
<td>XML</td>
<td>o11nplugin-xml.dar</td>
<td>XML</td>
<td>XML</td>
</tr>
<tr>
<td>Mail</td>
<td>o11nplugin-mail.dar</td>
<td>Mail</td>
<td>Mail</td>
</tr>
<tr>
<td>Net</td>
<td>o11nplugin-jakartacommonsnet.dar</td>
<td>None</td>
<td>Net</td>
</tr>
<tr>
<td>Workflow documentation</td>
<td>o11nplugin-wfdocs.dar</td>
<td>Workflow documentation</td>
<td>Workflow documentation</td>
</tr>
<tr>
<td>Common enumerated types</td>
<td>o11nplugin-enums.dar</td>
<td>None</td>
<td>Enums</td>
</tr>
<tr>
<td>vCO WebOperator</td>
<td>o11nplugin-weboperator.dar</td>
<td>None</td>
<td>N/A</td>
</tr>
</tbody>
</table>

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 and select Design or Run from the drop-down menu in the left upper corner.

2. Select Tools > API Explorer.
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.
Using the vCenter Server 5.1 Plug-In

You can use the vCenter Server 5.1 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 5.1 plug-in maps the vCenter Server 5.1 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 5.1 plug-in provides a library of standard workflows that automate vCenter Server 5.1 operations. For example, you can run workflows that create, clone, migrate, or delete virtual machines.

This chapter includes the following topics:
- “Configure the vCenter Server 5.1 Plug-In,” on page 17
- “vCenter Server 5.1 Plug-In Scripting API,” on page 18
- “Using the vCenter Server 5.1 Plug-In Inventory,” on page 18
- “Access the vCenter Server 5.1 Plug-In Workflow Library,” on page 18
- “vCenter Server 5.1 Plug-In Workflow Library,” on page 19

Configure the vCenter Server 5.1 Plug-In

You can configure Orchestrator to connect to your vCenter Server instances to run workflows over the objects in your vSphere infrastructure.

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 vCenter Server instance registered with the vSphere Web Client that uses the same vCenter Single Sign On instance with which you registered Orchestrator.

Prerequisites

Import the SSL certificates for each vCenter Server instance you define.

Procedure

1. Log in to the Orchestrator configuration interface as vmware.
2. Click vCenter Server, and click the New vCenter Server Host tab.
3. From the Available drop-down menu, select Enabled.
4. In the Host text box, type the IP address or the DNS name of the machine on which the vCenter Server instance you want to add is installed.
5. In the Port text box, retain the default value, 443.
6 (Optional) Select the Secure channel check box to establish a secure connection to your vCenter Server machine.

7 In the Path text box, retain the default value, /sdk.

This value is the location of the SDK that you use to connect to your vCenter Server instance.

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Share a unique session</td>
<td>Allows Orchestrator to create only one connection to vCenter Server. In the User name and Password text boxes, type 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 vCenter Web service, typically to operate Orchestrator system workflows.</td>
</tr>
<tr>
<td>Session per user</td>
<td>Creates a new session to vCenter Server. This 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 Sign On is enabled. The user that you select must be a valid user with privileges to manage vCenter Server extensions. You can leave the User name and Password text boxes empty, because the credentials are used only for connection check.</td>
</tr>
</tbody>
</table>

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 thus cannot collect the necessary data.

9 Click Apply changes.

The URL to the newly configured vCenter Server host is added to the list of defined hosts.

10 Repeat Step 2 through Step 9 for each vCenter Server instance.

**vCenter Server 5.1 Plug-In Scripting API**

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

For a list of available API objects, see http://www.vmware.com/support/orchestrator/doc/vco_vsphere51_api/index.html.

**Using the vCenter Server 5.1 Plug-In Inventory**

The vCenter Server plug-in exposes all objects of the connected vCenter Server hosts in the Inventory view. You can use the Inventory view 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.

**Access the vCenter Server 5.1 Plug-In Workflow Library**

You must use the Orchestrator client to access the elements from the vCenter Server 5.1 plug-in workflow library.

**Prerequisites**

- Configure the vCenter Server 5.1 plug-in in the Orchestrator configuration interface.
Log in to the Orchestrator client as a user who can run vCenter Server workflows.

Procedure
1. Log in to the Orchestrator client and 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 5.1 Plug-In Workflow Library
The vCenter Server 5.1 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.

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

- **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 of the workflows that have only one input parameter.
  - Fills the BatchAction configuration element with all of 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 cluster and compute resource workflows, you can create, rename or delete a cluster, and enable or disable high availability 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.

- **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 new 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.

Enable DRS on cluster
Enables DRS on a cluster.

Enable HA on cluster
Enables high availability on a cluster.

Remove virtual machine DRS group from cluster
Removes a DRS virtual machine group from a cluster.

Remove virtual machines from DRS group
Removes virtual machines from a cluster DRS group.

Rename cluster
Renames a cluster.

Guest Operation Files Workflows
With 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.

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.

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.

- **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.

- **Stop process in guest**
  Stops a process in a guest operating system.

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.

- **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.

- **Create datacenter**
  Creates a new datacenter in a datacenter folder.

- **Delete datacenter**
  Deletes a datacenter.

- **Reload datacenter**
  Forces vCenter Server to reload data from a datacenter.

- **Rename datacenter**
  Renames a datacenter and waits for the task to complete.

- **Rescan datacenter HBAs**
  Scans the hosts in a datacenter and initiates a rescan on the host bus adapters to discover new storage.
Datastore and Files Workflows

With 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.

- **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 Server instances 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.

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.

- **Create datacenter folder**
  - Creates a datacenter folder.

- **Delete datacenter folder**
  - Deletes a datacenter folder and waits for the task to complete.

- **Rename datacenter folder**
  - Renames a datacenter 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.

- **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.

- **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.

Basic Host Management Workflows

With basic host management workflows, you can put a host into maintenance mode, make a host exit maintenance mode, 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.

- **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 into a cluster. The host must be part of the same datacenter, 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 datacenter and the host must be in maintenance mode.
- **Reload host** Forces vCenter Server to reload data from a host.

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 does not receive an indication of success in the returned task, but rather loses the connection to the host if the operation succeeds.
- **Shut down host** Shuts down a host. If the Orchestrator client is connected directly to the host, it does not receive an indication of success in the returned task, but rather loses the connection to the host if the operation succeeds.
Host Registration Management Workflows

With 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.

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 vCenter Server.
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 vCenter Server. 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.

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 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.

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.

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 should 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 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.

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.

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's 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 VNIC associated with a port group in a standard virtual switch.

---

**Resource Pool Workflows**

With 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.

- **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 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.

- **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 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 so on.

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

**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 of 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.

**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.
### Basic Virtual Machine Management Workflows

With 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 so on.

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

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create custom virtual machine</td>
<td>Creates a virtual machine with the specified configuration options and additional devices.</td>
</tr>
<tr>
<td>Create simple dvPortGroup virtual machine</td>
<td>Creates a simple virtual machine. The network used is a Distributed Virtual Port Group.</td>
</tr>
<tr>
<td>Create simple virtual machine</td>
<td>Creates a virtual machine with the most common devices and configuration options.</td>
</tr>
<tr>
<td>Delete virtual machine</td>
<td>Removes a virtual machine from the inventory and datastore.</td>
</tr>
<tr>
<td>Get virtual machines by name</td>
<td>Returns a list of virtual machines from all registered vCenter Server hosts that match the provided expression.</td>
</tr>
<tr>
<td>Mark as template</td>
<td>Converts an existing virtual machine to a template, not allowing it to start. You can use templates to create virtual machines.</td>
</tr>
<tr>
<td>Mark as virtual machine</td>
<td>Converts an existing template to a virtual machine, allowing it to start.</td>
</tr>
<tr>
<td>Move virtual machine to folder</td>
<td>Moves a virtual machine to a specified virtual machine folder.</td>
</tr>
<tr>
<td>Move virtual machine to resource pool</td>
<td>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.</td>
</tr>
<tr>
<td>Move virtual machines to folder</td>
<td>Moves several virtual machines to a specified virtual machine folder.</td>
</tr>
<tr>
<td>Move virtual machines to resource pool</td>
<td>Moves several virtual machines to a resource pool.</td>
</tr>
<tr>
<td>Register virtual machine</td>
<td>Registers a virtual machine. The virtual machine files must be placed in an existing datastore and must not be already registered.</td>
</tr>
<tr>
<td>Reload virtual machine</td>
<td>Forces vCenter Server to reload a virtual machine.</td>
</tr>
<tr>
<td>Rename virtual machine</td>
<td>Renames an existing virtual machine on the vCenter Server system or host and not on the datastore.</td>
</tr>
<tr>
<td>Set virtual machine performance</td>
<td>Changes performance settings such as shares, minimum and maximum values, shaping for network, and disk access of a virtual machine.</td>
</tr>
<tr>
<td>Unregister virtual machine</td>
<td>Removes an existing virtual machine from the inventory.</td>
</tr>
<tr>
<td>Upgrade virtual machine hardware (force if required)</td>
<td>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.</td>
</tr>
</tbody>
</table>
**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.

- **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 linked clone workflows, you can perform linked clone operations such as restoring a virtual machine from a linked clone, creating a linked clone, and so on.

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.

- **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.

- **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 tools clone workflows you can obtain customization information about the operating system of the virtual machine, information needed to update a virtual device, and so on.

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

- **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 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.

- **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. Specifies virtual disk thin provisioning policy and configures one virtual network card and a local administrator user account. Sysprep tools must be available on the vCenter Server system.

- **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 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.

- **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 floppy disk drive
Mounts a floppy disk drive FLP file from the ESX datastore.

### Move and Migrate Workflows

With 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.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mass migrate virtual machines with storage vMotion</strong></td>
<td>Uses Storage vMotion to migrate a single virtual machine, a selection of virtual machines, or all available virtual machines.</td>
</tr>
<tr>
<td><strong>Mass migrate virtual machines with vMotion</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Migrate virtual machine with vMotion</strong></td>
<td>Migrates a virtual machine from one host to another by using the MigrateVM_Task operation from the vSphere API.</td>
</tr>
<tr>
<td><strong>Move virtual machine to another vCenter Server system</strong></td>
<td>Moves a list of virtual machines to another vCenter Server system.</td>
</tr>
<tr>
<td><strong>Quick migrate multiple virtual machines</strong></td>
<td>Suspends the virtual machines if they are powered on and migrates them to another host using the same storage.</td>
</tr>
<tr>
<td><strong>Quick migrate virtual machine</strong></td>
<td>Suspends the virtual machine if it is powered on and migrates it to another host using the same storage.</td>
</tr>
<tr>
<td><strong>Relocate virtual machine disks</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

### Other Workflows

With other workflows, 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.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Disable FT</strong></td>
<td>Disables Fault Tolerance for a specified virtual machine.</td>
</tr>
<tr>
<td><strong>Enable FT</strong></td>
<td>Enables Fault Tolerance for a specified virtual machine.</td>
</tr>
<tr>
<td><strong>Extract virtual machine information</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Find orphaned virtual machines</strong></td>
<td>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).</td>
</tr>
</tbody>
</table>
Power Management Workflows

With 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 so on.

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

- **Power off virtual machine and wait**: Powers off a virtual machine and waits for the process to complete.
- **Reboot guest OS**: Reboots the virtual machine’s guest operating system. 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.
- **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.

- **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.

- **Mount VMware tools installer**
  Mounts the VMware Tools installer on the virtual CD-ROM.

- **Set console screen resolution**
  Sets the console window's resolution. The virtual machine must be powered on.

- **Turn on time synchronization**
  Turns on time synchronization between the virtual machine and the ESX server in VMware Tools.

- **Unmount VMware tools installer**
  Unmounts the VMware Tools CD-ROM.

- **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 vCO Library Plug-In

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

vCO Library Plug-In Workflows

The vCO 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.

- 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.

- Create recurrent task: Creates a recurrent task an 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.

Start workflows in a series
Run 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
Run 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.

Troubleshooting Workflows

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

Export logs and application settings
Generates a ZIP archive of troubleshooting information that contains configuration files and server, configuration, wrapper, and installation log files. The output directory must exist and Orchestrator must have writing permissions.
Using the SQL Plug-In

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,” on page 37
- “SQL Plug-In Scripting API,” on page 39
- “Running the SQL Sample Workflows,” on page 45
- “Using the SQL Plug-In Standard Workflows,” on page 50

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.

<table>
<thead>
<tr>
<th>Workflow Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a database</td>
<td>Adds a database object to the Database plug-in inventory.</td>
</tr>
<tr>
<td>Add tables to a database</td>
<td>Adds database tables to a database in the Database plug-in inventory.</td>
</tr>
<tr>
<td>Remove a database</td>
<td>Removes a database object from the Database plug-in inventory.</td>
</tr>
<tr>
<td>Remove a table from a database</td>
<td>Removes a database table from a database in the Database plug-in inventory.</td>
</tr>
<tr>
<td>Update a database</td>
<td>Updates the configuration of a database object in the Database plug-in inventory.</td>
</tr>
<tr>
<td>Validate a database</td>
<td>Validates a database in the Database plug-in inventory.</td>
</tr>
</tbody>
</table>
Add a Database

You can run a workflow to add a database 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 from the Network tab in the Orchestrator configuration interface.

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, select 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.
   After the workflow runs successfully, the database appears in the Inventory view of the Orchestrator client with the name that you specify.
6. Select the type of the database.
7. In the Connection URL text box, type the address of the database.

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>jdbc:oracle:thin:@database_url:port_number:SID</td>
</tr>
<tr>
<td>Microsoft SQL</td>
<td>jdbc:jtds:sqlserver://database_url:port_number/database_name</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>jdbc:postgresql://database_url:port_number/database_name</td>
</tr>
<tr>
<td>MySQL</td>
<td>jdbc:mysql://database_url:port_number/database_name</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared Session</td>
<td>The plug-in uses shared credentials to connect to the database. You must provide the database credentials for the shared session.</td>
</tr>
<tr>
<td>Session Per User</td>
<td>The Orchestrator client retrieves credentials from the user who is logged in.</td>
</tr>
</tbody>
</table>

9. Click Submit to run the workflow.

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, select 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.

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's 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, select **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.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shared Session</strong></td>
<td>The plug-in uses shared credentials to connect to the database. You must provide the database credentials for the shared session.</td>
</tr>
<tr>
<td><strong>Session Per User</strong></td>
<td>The Orchestrator client retrieves credentials from the user who is logged in.</td>
</tr>
</tbody>
</table>

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

### SQL Plug-In Scripting API

The SQL scripting API contains classes, with their respective methods, that enable interaction between vCenter Orchestrator and SQL-based databases. You can use the API to create workflows that read data from and write data to SQL-based databases.

#### SQL:ActiveRecord Type

The SQL:ActiveRecord type uses the SQLActiveRecord class as its scripting object.

The SQL:ActiveRecord type contains no properties.
**SQL:Column Type**

The SQL:Column type uses the SQLColumn class as its scripting object.

The SQL:Column type contains the following properties.

- name
- typeName
- mandatory
- table

**SQL:Database Type**

The SQL:Database type uses the SQLDatabase class as its scripting object.

The SQL:Database type contains the following properties.

- name
- type
- connectionURL
- sessionMode
- username

**SQL:Table Type**

The SQL:Table type uses the SQLTable class as its scripting object.

The SQL:Table type contains the following properties.

- name
- database

**Connection Class**

The Connection class contains methods that control the connection to a database.

The Connection class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close():Object</td>
<td>Object</td>
<td>Releases the database and JDBC resources for a Connection object.</td>
</tr>
<tr>
<td>createStatement():Object</td>
<td>Object</td>
<td>Creates a Statement object for sending SQL statements to the database.</td>
</tr>
</tbody>
</table>
**JDBCConnection Class**

The JDBCConnection class contains a method that specifies the URL and credentials for a connection to a database.

The Connection class defines the following method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getConnection(string,string,string):Connection</td>
<td>Connection</td>
<td>The three strings represent URL, user name, and password.</td>
</tr>
</tbody>
</table>

**PreparedStatement Class**

The PreparedStatement class represents a precompiled SQL statement.

The PreparedStatement class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>execute():boolean</td>
<td>Boolean</td>
<td>Runs an SQL statement in the PreparedStatement object.</td>
</tr>
<tr>
<td>executeQuery():ResultSet</td>
<td>ResultSet</td>
<td>Runs an SQL query in the PreparedStatement object and returns the ResultSet object generated by the query.</td>
</tr>
<tr>
<td>executeUpdate():number</td>
<td>Number</td>
<td>Runs an SQL INSERT, UPDATE, or DELETE statement, or an SQL statement that returns nothing, in the PreparedStatement object.</td>
</tr>
<tr>
<td>setDate(number,object):Object</td>
<td>Object</td>
<td>Sets a designated parameter to the given Date value.</td>
</tr>
<tr>
<td>setString(number,object):Object</td>
<td>Object</td>
<td>Sets a designated parameter to the given String value.</td>
</tr>
<tr>
<td>setTimestamp(number,object):Object</td>
<td>Object</td>
<td>Sets a designated parameter to the given Timestamp value.</td>
</tr>
</tbody>
</table>

**ResultSet Class**

The ResultSet class is the result of a run of an executeQuery method.

The ResultSet class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>afterLast():Object</td>
<td>Object</td>
<td>Moves the cursor to the end of the ResultSet object, after the last row. This method has no effect if the result set contains no rows.</td>
</tr>
<tr>
<td>beforeFirst():Object</td>
<td>Object</td>
<td>Moves the cursor to the front of the ResultSet object, before the first row. This method has no effect if the result set contains no rows.</td>
</tr>
<tr>
<td>first():boolean</td>
<td>Boolean</td>
<td>Moves the cursor to the first row in the ResultSet object. Returns true if the cursor is on a valid row, and false if there are no rows in the result set.</td>
</tr>
<tr>
<td>Method</td>
<td>Returns</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getDate(string):Object</td>
<td>Object</td>
<td>Returns the Date value of a given column name.</td>
</tr>
<tr>
<td>getDateAt(number):Object</td>
<td>Object</td>
<td>Returns the Date value of a given column index.</td>
</tr>
<tr>
<td>getNumber(string):number</td>
<td>Number</td>
<td>Returns the Number value of a given column name.</td>
</tr>
<tr>
<td>getNumberAt(number):number</td>
<td>Number</td>
<td>Returns the Number value of a given column index.</td>
</tr>
<tr>
<td>getString(string):string</td>
<td>String</td>
<td>Returns the String value of a given column name.</td>
</tr>
<tr>
<td>getStringAt(number):string</td>
<td>String</td>
<td>Returns the String value of a given column index.</td>
</tr>
<tr>
<td>getTimestamp(string):Object</td>
<td>Object</td>
<td>Returns the Timestamp value of a given column name.</td>
</tr>
<tr>
<td>getTimestampAt(number):Object</td>
<td>Object</td>
<td>Returns the Timestamp value of a given column index.</td>
</tr>
<tr>
<td>last():boolean</td>
<td>Boolean</td>
<td>Moves the cursor to the last row in the ResultSet object. Returns \texttt{true} if the cursor is on a valid row, and \texttt{false} if there are no rows in the result set.</td>
</tr>
<tr>
<td>next():boolean</td>
<td>Boolean</td>
<td>Moves the cursor down one row. Returns \texttt{true} if the cursor is on a valid row, and \texttt{false} if there are no more rows.</td>
</tr>
<tr>
<td>previous():boolean</td>
<td>Boolean</td>
<td>Moves the cursor to the previous row in the ResultSet object. Returns \texttt{true} if the cursor is on a valid row, and \texttt{false} if it is off the result set.</td>
</tr>
<tr>
<td>wasNull():boolean</td>
<td>Boolean</td>
<td>Reports whether the last column read had a value of SQL NULL.</td>
</tr>
</tbody>
</table>

**ResultSetMetaData Class**

The ResultSetMetaData class contains the metadata of a ResultSet method.

The ResultSetMetaData class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getColumnCount():number</td>
<td>Number</td>
<td>Returns the number of columns in the selected ResultSet.</td>
</tr>
<tr>
<td>getColumnName(number):string</td>
<td>String</td>
<td>Returns the name of the column at position index (starting with one).</td>
</tr>
<tr>
<td>getTableName(number):string</td>
<td>String</td>
<td>Returns the table name to which the column belongs.</td>
</tr>
</tbody>
</table>

**SQLActiveRecord Class**

The SQLActiveRecord class contains a method that represents a table record.

The SQLActiveRecord class defines the following method.
Method | Returns | Description
--- | --- | ---
getProperty(String):Object | Object | Returns the value of the given property.

**SQLColumn Class**

The `SQLColumn` class represents a SQL table column.

The `SQLColumn` class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mandatory</td>
<td>Boolean</td>
<td>A mandatory column</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The column name</td>
</tr>
<tr>
<td>table</td>
<td>com.vmware.o11n.plugin.database.Table</td>
<td>The table to which the column belongs</td>
</tr>
<tr>
<td>type ID</td>
<td>Integer</td>
<td>The type ID</td>
</tr>
<tr>
<td>type name</td>
<td>String</td>
<td>The type name</td>
</tr>
</tbody>
</table>

**SQLDatabase Class**

The `SQLDatabase` class represents a SQL database connection.

The `SQLDatabase` class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectionURL</td>
<td>String</td>
<td>The database connection URL</td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>The database ID</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The database name</td>
</tr>
<tr>
<td>password</td>
<td>String</td>
<td>The database password</td>
</tr>
<tr>
<td>sessionMode</td>
<td>String</td>
<td>The user session mode</td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>The database type</td>
</tr>
<tr>
<td>username</td>
<td>String</td>
<td>The user name</td>
</tr>
</tbody>
</table>

The `SQLDatabase` class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>executeCustomQuery(String):int</td>
<td>Integer</td>
<td>Runs a non-read custom query on the database.</td>
</tr>
<tr>
<td>getAllTableNames():String[]</td>
<td>Array of Strings</td>
<td>Returns a list of all of the tables in the database.</td>
</tr>
<tr>
<td>getMappedTableNames():String[]</td>
<td>Array of Strings</td>
<td>Returns a list of all of the tables added to the database.</td>
</tr>
<tr>
<td>getTables():SQLTable[]</td>
<td>Array of SQL Tables</td>
<td>Returns a list of tables in the database.</td>
</tr>
<tr>
<td>getUnmappedTableNames():String[]</td>
<td>Array of Strings</td>
<td>Returns a list of tables that are not added to the database.</td>
</tr>
<tr>
<td>readCustomQuery(String):ResultSet</td>
<td>ResultSet</td>
<td>Runs a read custom query on the database and returns the results in a ResultSet.</td>
</tr>
</tbody>
</table>
**SQLDatabaseManager Class**

The **SQLDatabaseManager** class provides CRUD operations for database objects.

The **SQLDatabaseManager** class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addDatabase(SQLDatabase):SQLDatabase</td>
<td>Boolean</td>
<td>Adds a database object to the SQL plug-in inventory.</td>
</tr>
<tr>
<td>addTablesToDatabase(SQLDatabase, String[]):SQLTable[]</td>
<td>Array of SQL Table</td>
<td>Adds new tables to a database.</td>
</tr>
<tr>
<td>convertBooleanForPresentation(String):String</td>
<td>String</td>
<td>Converts Boolean values to Yes or No strings.</td>
</tr>
<tr>
<td>generateCreateWorkflow(SQLTable, ch.dunes.model.workflow.WorkflowCategory, Boolean, SQLColumn[]): ch.dunes.model.workflow.Workflow</td>
<td></td>
<td>Generates a workflow that creates database records.</td>
</tr>
<tr>
<td>generateDeleteWorkflow(SQLTable, ch.dunes.model.workflow.WorkflowCategory, Boolean): ch.dunes.model.workflow.Workflow</td>
<td></td>
<td>Generates a workflow that deletes table records.</td>
</tr>
<tr>
<td>generateReadWorkflow(SQLTable, ch.dunes.model.workflow.WorkflowCategory, Boolean): ch.dunes.model.workflow.Workflow</td>
<td></td>
<td>Generates a workflow that reads table records.</td>
</tr>
<tr>
<td>generateUpdateWorkflow(SQLTable, ch.dunes.model.workflow.WorkflowCategory, Boolean, SQLColumn[]): ch.dunes.model.workflow.Workflow</td>
<td></td>
<td>Generates a workflow that updates a table record.</td>
</tr>
<tr>
<td>getConnectionUrl(String, String): String</td>
<td>String</td>
<td>Returns a connection URL pattern for a particular database type.</td>
</tr>
<tr>
<td>getDatabase(String): SQLDatabase</td>
<td>SQLDatabase</td>
<td>Returns database objects with a specified name from the SQL plug-in inventory.</td>
</tr>
<tr>
<td>getDatabaseById(String): SQLDatabase</td>
<td>SQLDatabase</td>
<td>Returns database objects with a specified ID from the SQL plug-in inventory.</td>
</tr>
<tr>
<td>getDatabases(): SQLDatabase[]</td>
<td>Array of SQLDatabase</td>
<td>Returns a list of database objects in the SQL plug-in inventory.</td>
</tr>
<tr>
<td>getSupportedDatabaseTypes(): String[]</td>
<td>Array of strings</td>
<td>Returns a list of supported database types.</td>
</tr>
<tr>
<td>isValidNumberOrEmpty(Object): Boolean</td>
<td>Boolean</td>
<td>Returns true if the object you passed is a valid number or null. Otherwise returns false.</td>
</tr>
<tr>
<td>removeDatabase(SQLDatabase): SQLDatabase</td>
<td>SQLDatabase</td>
<td>Removes a database object from the SQL plug-in inventory.</td>
</tr>
<tr>
<td>removeTableFromDatabase(SQLTable): SQLTable</td>
<td>SQLTable</td>
<td>Removes a specified table from the database.</td>
</tr>
</tbody>
</table>
### SQLTable Class

The SQLTable class represents a database table. The SQLTable class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>com.vmware.o11n.plugin.databas.e.Database</td>
<td>The database to which the table belongs.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>The table name</td>
</tr>
</tbody>
</table>

The SQLTable class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>createRecord(Object, Boolean):SQLActiveRecord</td>
<td>SQLActiveRecord</td>
<td>Creates a new active record with primary fields.</td>
</tr>
<tr>
<td>deleteRecords(Object, Boolean):Integer</td>
<td>Integer</td>
<td>Deletes a record by certain search criteria.</td>
</tr>
<tr>
<td>findUniqueRecord(Object[]):SQLActiveRecord</td>
<td>SQLActiveRecord</td>
<td>Finds a record by certain search criteria.</td>
</tr>
<tr>
<td>readRecords(Object, Boolean):SQLActiveRecord[]</td>
<td>Array of SQLActiveRecord[]</td>
<td>Reads records by certain search criteria.</td>
</tr>
<tr>
<td>updateRecord(SQLActiveRecord, Object):Integer</td>
<td>Integer</td>
<td>Updates the record that matches the primary key fields.</td>
</tr>
</tbody>
</table>

### 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**

Log in to the Orchestrator client as a user who can run JDBC workflows.

**Procedure**

1. Click the Workflows view in the Orchestrator client.
2. In the workflows hierarchical list, open 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**

Log in to the Orchestrator client as a user who can run JDBC workflows.

**Procedure**

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

2 In the workflows hierarchical list, open **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**

Log in to the Orchestrator client as a user who can run JDBC workflows.

**Procedure**

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

2 In the workflows hierarchical list, open **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:
   ```sql
   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**

Log in to the Orchestrator client as a user who can run JDBC workflows.

**Procedure**

1 Click the Workflows view in the Orchestrator client.
2 In the workflows hierarchical list, open 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:
   ```sql
   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**

Log in to the Orchestrator client as a user who can run JDBC workflows.

**Procedure**

1 Click the Workflows view in the Orchestrator client.
2 In the workflows hierarchical list, open 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:
   ```sql
   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**

Log in to the Orchestrator client as a user who can run JDBC workflows.

**Procedure**

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

2 In the workflows hierarchical list, open **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:
   ```sql
   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**

Log in to the Orchestrator client as a user who can run JDBC workflows.

**Procedure**

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

2 In the workflows hierarchical list, open **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:
   
   ```sql
   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**

Log in to the Orchestrator client as a user who can run JDBC workflows.

**Procedure**

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

2 In the workflows hierarchical list, open **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:
   
   ```sql
   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**

Log in to the Orchestrator client as a user who can run JDBC workflows.

**Procedure**

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

2 In the workflows hierarchical list, open **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 as well as to run SQL operations.

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

- **Add a database**: Adds a database object to the plug-in's inventory.
- **Add tables to a database**: Adds database tables to a database in the plug-in's inventory.
- **Remove a database**: Removes a database object from the plug-in's inventory.
- **Remove a table from a database**: Removes a database table from a database in the plug-in's inventory.
- **Update a database**: Updates the configuration of a database object in the plug-in's inventory.
- **Validate a database**: Validates a database in the plug-in's inventory.

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

- **Execute a custom query on a database**: Executes 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**: Executes 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, select **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.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>The generated workflows overwrite existing workflows with the same name.</td>
</tr>
<tr>
<td>No</td>
<td>New workflows are not generated if workflows with the same name exist in the folder.</td>
</tr>
</tbody>
</table>

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.

   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

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,” on page 53
- “SSH Plug-In Scripting API,” on page 54
- “Running the SSH Plug-In Sample Workflows,” on page 58

Configure the SSH Plug-In

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

Procedure

1. Log in to the Orchestrator configuration interface as **vmware**.
2. Click **SSH**.
3. Click **New connection**.
4. In the **Host name** text box, type the host to access with SSH through Orchestrator.

   **NOTE**  No username and password are required because Orchestrator uses the credentials of the currently logged-in user to run SSH commands. You must reproduce the accounts you want to work on SSH on target hosts from the LDAP server.

5. Click **Apply changes**.

   The host is added to the list of SSH connections.
6. (Optional) Configure an entry path on the server.
   a. Click **New root folder**.
   b. Enter the new path and click **Apply changes**.

The SSH host is available in the **Inventory** view of the Orchestrator client.
SSH Plug-In Scripting API

The SSH scripting API contains classes, with their respective attributes, methods, and constructors, that allow Orchestrator to perform SSH operations from workflows. You can use the API to develop custom workflows that access a remote server through SSH.

**SSH:File Type**

The SSH:File type uses the SSHFile class as its scripting object.

The SSH:File type contains the following properties.

- path
- name
- hostname
- port

**SSH:Folder Type**

The SSH:Folder type uses the SSHFolder class as its scripting object.

The SSH:Folder type contains the following properties.

- path
- name
- hostname
- port

**SSH:RootFolder Type**

The SSH:RootFolder type uses the SSHFolder class as its scripting object.

The SSH:RootFile type contains the following property.

- name

**SSH:SshConnection Type**

The SSH:SshConnection type uses the FinderResult class as its scripting object.

The SSH:SshConnection type contains the following properties.

- name
- userName

**KeyPairManager Class**

The KeyPairManager class contains a set of functions to manage private and public SSH keys.

The KeyPairManager class defines the following methods.
### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>changePassphrase(Path,SecureString,SecureString):Object</td>
<td>Object</td>
<td>Changes the passphrase of a private key.</td>
</tr>
<tr>
<td>generateKeyPair(string,Path,SecureString,number,string):string</td>
<td>String</td>
<td>Generates a pair of a public and a private key. Returns the generated key fingerprint.</td>
</tr>
</tbody>
</table>

### SSHCommand Class

The SSHCommand class is the main class for running SSH commands from workflows.

The SSHCommand class defines the following constructors.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSHCommand(string,string,SecureString):SSHCommand</td>
<td>Creates a new SSHCommand.</td>
</tr>
<tr>
<td>SSHCommand(string,string,string,number):SSHCommand</td>
<td>Creates a new SSHCommand.</td>
</tr>
</tbody>
</table>

The SSHCommand class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd</td>
<td>String</td>
<td>Command to run</td>
</tr>
<tr>
<td>error</td>
<td>String</td>
<td>Command error, if any</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong>: The error message depends on your operating system. The operating system returns a string from the error messages buffer.</td>
</tr>
<tr>
<td>exitCode</td>
<td>Number</td>
<td>Exit code of the last command</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong>: The exit code depends on your operating system. Refer to your operating system’s documentation for descriptions.</td>
</tr>
<tr>
<td>output</td>
<td>String</td>
<td>Command output, if any</td>
</tr>
<tr>
<td>state</td>
<td>String</td>
<td>Run state</td>
</tr>
</tbody>
</table>

The SSHCommand class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disconnect():Object</td>
<td>Object</td>
<td>Disconnects the current session.</td>
</tr>
<tr>
<td>execute():Object</td>
<td>Object</td>
<td>Runs a single command and returns immediately. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>executeAndLog(string):Object</td>
<td>Object</td>
<td>Runs a single command and waits until it is completed.</td>
</tr>
<tr>
<td>executeCommand(string,boolean):string</td>
<td>String</td>
<td>Runs a single command, waits until it is completed, and returns the stdout result, if synchronous. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>findAll(string,string):string[]</td>
<td>Array of String</td>
<td>Searches recursively and returns files and directories that match a pattern. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>Method</td>
<td>Returns</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>findDir(string,string):string[]</td>
<td>Array of String</td>
<td>Searches recursively and returns directories that match a pattern. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>findFile(string,string):string[]</td>
<td>Array of String</td>
<td>Searches recursively and returns files that match a pattern. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>getFile(string,string):number</td>
<td>Number</td>
<td>Copies a file from a remote host to the Orchestrator server. Leaves an open session. You can disconnect the session manually. Returns 0 if successful, -1 if an error has occurred, and -2 if a fatal error has occurred.</td>
</tr>
<tr>
<td>listAll(string):string[]</td>
<td>Array of String</td>
<td>Lists files and directories in a path. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>listDir(string):string[]</td>
<td>Array of String</td>
<td>Lists directories in a path. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>listFile(string):string[]</td>
<td>Array of String</td>
<td>Lists files in a path. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>putFile(string,string):number</td>
<td>Number</td>
<td>Copies a file from the Orchestrator server to a remote host. The destination directory must exist. Returns 0 if successful, or -1 if an error has occurred.</td>
</tr>
</tbody>
</table>

**SSHFile Class**

The SSHFile class contains attributes for SSH access to files on a remote file system.

The SSHFile class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>String</td>
<td>File name</td>
</tr>
<tr>
<td>hostname</td>
<td>String</td>
<td>SSH host name</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>File name</td>
</tr>
<tr>
<td>path</td>
<td>String</td>
<td>Full file path</td>
</tr>
<tr>
<td>port</td>
<td>Number</td>
<td>SSH port</td>
</tr>
</tbody>
</table>

**SSHFolder Class**

The SSHFolder class contains attributes for SSH access to folders on a remote file system.

The SSHFolder class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>folderName</td>
<td>String</td>
<td>Folder name</td>
</tr>
<tr>
<td>hostname</td>
<td>String</td>
<td>SSH host name</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Folder name</td>
</tr>
</tbody>
</table>
### SSHSession Class

The SSHSession class is the main class for SSH session management.

The SSHSession class defines the following constructors.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSHSession(string, string): SSHSession</td>
<td>Creates a new SSHSession.</td>
</tr>
<tr>
<td>SSHSession(string, string, number): SSHSession</td>
<td>Creates a new SSHSession.</td>
</tr>
</tbody>
</table>

The SSHSession class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd</td>
<td>String</td>
<td>Command to run</td>
</tr>
<tr>
<td>error</td>
<td>String</td>
<td>Command error, if any. Note: The error message depends on your operating system. The operating system returns a string from the error messages buffer.</td>
</tr>
<tr>
<td>exitCode</td>
<td>Number</td>
<td>Exit code of the last command. Note: The exit code depends on your operating system. Refer to your operating system’s documentation for descriptions.</td>
</tr>
<tr>
<td>output</td>
<td>String</td>
<td>Command output, if any</td>
</tr>
<tr>
<td>pty</td>
<td>Boolean</td>
<td>Requesting a pseudo-terminal</td>
</tr>
<tr>
<td>state</td>
<td>String</td>
<td>Run state</td>
</tr>
<tr>
<td>terminal</td>
<td>String</td>
<td>Terminal type</td>
</tr>
</tbody>
</table>

The SSHSession class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addEnvironment(string, string): Object</td>
<td>Object</td>
<td>Fills a property list of environment variables that are set prior to opening a channel.</td>
</tr>
<tr>
<td>connectWithIdentity(Path, Secure String): Object</td>
<td>Object</td>
<td>Connects the session using public key authentication.</td>
</tr>
<tr>
<td>connectWithPassword(SecureString): Object</td>
<td>Object</td>
<td>Connects the session using simple password authentication.</td>
</tr>
<tr>
<td>connectWithPasswordOrIdentity (boolean, SecureString, Path): Object</td>
<td>Object</td>
<td>Connects the session using either password or public key authentication.</td>
</tr>
<tr>
<td>disconnect(): Object</td>
<td>Object</td>
<td>Disconnects the current session.</td>
</tr>
<tr>
<td>execute(): Object</td>
<td>Object</td>
<td>Runs a single command and returns immediately. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>Method</td>
<td>Returns</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>executeAndLog(string):Object</td>
<td>Object</td>
<td>Runs a single command and waits until it is completed.</td>
</tr>
<tr>
<td>executeCommand(string,boolean):</td>
<td>String</td>
<td>Runs a single command, waits until it is completed, and returns the stdout result, if synchronous. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>findAll(Path,string):string[]</td>
<td>Array of String</td>
<td>Searches recursively and returns files and directories that match a pattern. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>findDir(Path,string):string[]</td>
<td>Array of String</td>
<td>Searches recursively and returns directories that match a pattern. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>findFile(Path,string):string[]</td>
<td>Array of String</td>
<td>Searches recursively and returns files that match a pattern. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>getFile(Path,Path):number</td>
<td>Number</td>
<td>Copies a file from a remote host to the Orchestrator server. Leaves an open session. You can disconnect the session manually. Returns 0 if successful, or -1 if an error has occured.</td>
</tr>
<tr>
<td>listAll(Path):string[]</td>
<td>Array of String</td>
<td>Lists files and directories in a path. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>listDir(Path):string[]</td>
<td>Array of String</td>
<td>Lists directories in a path. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>listFile(Path):string[]</td>
<td>Array of String</td>
<td>Lists files in a path. Leaves an open session. You can disconnect the session manually.</td>
</tr>
<tr>
<td>putFile(Path,Path):number</td>
<td>Number</td>
<td>Copies a file from the Orchestrator server to a remote host. The destination directory must exist. Returns 0 if successful, or -1 if an error has occured.</td>
</tr>
</tbody>
</table>

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.

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
- Configure the SSH plug-in in the Orchestrator configuration interface.
- Log in to the Orchestrator client as a user who can run SSH workflows.

Procedure
1. Click the Workflows view in the Orchestrator client.
2. In the workflows hierarchical list, open Library > SSH to 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) Type a passphrase.
   
   **Note**: You can change the passphrase later.

   d. (Optional) Type 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 last.

Prerequisites
- Configure the SSH plug-in in the Orchestrator configuration interface.
- Log in to the Orchestrator client as a user who can run SSH workflows.

Procedure
1. Click the Workflows view in the Orchestrator client.
2. In the workflows hierarchical list, open Library > SSH to 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. Type the current passphrase.
   b. Type 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
- Configure the SSH plug-in in the Orchestrator configuration interface.
- Log in to the Orchestrator client as a user who can run SSH workflows.
Procedure

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

2. In the workflows hierarchical list, open **Library > SSH** to 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 host and authentication information.

   **Note** You must provide credentials that are registered on the SSH host.

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

   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 ESX host.

**Prerequisites**

- Configure the SSH plug-in in the Orchestrator configuration interface.
- Log in to the Orchestrator client as a user who can run SSH workflows.

**Procedure**

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

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

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

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

5. Type 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. Type a user name, and click **Next**.

8. Type a password if the authentication method requires a password. Otherwise, type the path to the private key and type 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**

- Configure the SSH plug-in in the Orchestrator configuration interface.
Log in to the Orchestrator client as a user who can run SSH workflows.

**Procedure**

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, open **Library > SSH** to 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**

- Configure the SSH plug-in in the Orchestrator configuration interface.
- Log in to the Orchestrator client as a user who can run SSH workflows.

**Procedure**

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, open **Library > SSH** to 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. Type the file information.
   a. Type the path to the file that you want to copy from the local Orchestrator server to the remote SSH host.
   b. Type 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

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 vCenter 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:

- “XML Plug-In Scripting API,” on page 63
- “Running the XML Plug-In Sample Workflows,” on page 67

XML Plug-In Scripting API

The XML scripting API contains classes, with their respective attributes and methods, that allow vCenter Orchestrator to manage XML documents. You can use the API to develop custom workflows.

**XMLDocument Class**

The XMLDocument class is the main XML class.

The XMLDocument class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appendChild(XMLNode):Object</td>
<td>Object</td>
<td>Adds a newChild node to the end of the list of child nodes of this node. If the newChild node exists in the tree, it is first removed.</td>
</tr>
<tr>
<td>cloneNode(boolean):Object</td>
<td>Object</td>
<td>Clones a node.</td>
</tr>
<tr>
<td>createCDATASection(Object):XMLNode</td>
<td>XMLNode</td>
<td>Creates a CDATA node.</td>
</tr>
<tr>
<td>createComment(String):XMLNode</td>
<td>XMLNode</td>
<td>Creates a comment node.</td>
</tr>
<tr>
<td>createElement(String):XMLElement</td>
<td>XMLElement</td>
<td>Creates an element with a given name.</td>
</tr>
<tr>
<td>createProcessingInstruction(String,String):XMLNode</td>
<td>XMLNode</td>
<td>Creates a ProcessingInstruction node with specified name and data strings.</td>
</tr>
<tr>
<td>Method</td>
<td>Returns</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>--------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>createTextNode(String):XMLNode</td>
<td>XMLNode</td>
<td>Creates a text node.</td>
</tr>
<tr>
<td>getChildNodes():Object</td>
<td>Object</td>
<td>Gets all child nodes.</td>
</tr>
<tr>
<td>getDocumentElement():XMLElement</td>
<td>XMLElement</td>
<td>Allows direct access to the child node that is at the root element of the document.</td>
</tr>
<tr>
<td>getElementsByTagName(String):Object</td>
<td>Object</td>
<td>Returns an XMLNodeList of all elements with a given tag name in the order in which they are encountered in a pre-ordered traversal of the XMLDocument tree.</td>
</tr>
<tr>
<td>insertBefore(XMLNode,XMLNode):Object</td>
<td>Object</td>
<td>Inserts a newChild node before the existing child node refChild. If refChild is null, the newChild node is inserted at the end of list of child nodes. If the newChild node exists in the tree, it is first removed.</td>
</tr>
<tr>
<td>normalize():Object</td>
<td>Object</td>
<td>Normalizes the document.</td>
</tr>
<tr>
<td>removeChild(XMLNode):Object</td>
<td>Object</td>
<td>Removes the child node indicated by oldChild from the list of child nodes and returns it.</td>
</tr>
<tr>
<td>replaceChild(XMLNode,XMLNode):Object</td>
<td>Object</td>
<td>Replaces the oldChild node with a newChild node in the list of child nodes and returns the oldChild node. If the newChild node exists in the tree, it is first removed.</td>
</tr>
</tbody>
</table>

**XMLElement Class**

The `XMLElement` class is the main element class.

The `XMLElement` class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagName</td>
<td>String</td>
<td>The tag name</td>
</tr>
<tr>
<td>textContent</td>
<td>String</td>
<td>The text content</td>
</tr>
</tbody>
</table>

The `XMLElement` class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appendChild(XMLNode):Object</td>
<td>Object</td>
<td>Adds a newChild node to the end of the list of child nodes of this node. If the newChild node exists in the tree, it is first removed.</td>
</tr>
<tr>
<td>cloneNode(boolean):Object</td>
<td>Object</td>
<td>Clones a node.</td>
</tr>
<tr>
<td>getAttributes():Object</td>
<td>Object</td>
<td>Contains the attributes of an element.</td>
</tr>
<tr>
<td>getDocumentElement():XMLElement</td>
<td>XMLNodeList</td>
<td>Gets all child nodes.</td>
</tr>
<tr>
<td>getElementsByTagName(String):Object</td>
<td>Object</td>
<td>Returns an XMLNodeList of all elements with a given tag name in the order in which they are encountered in a pre-ordered traversal of the current element tree.</td>
</tr>
<tr>
<td>Method</td>
<td>Returns</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>hasAttribute(String):boolean</td>
<td>Boolean</td>
<td>Returns true if the attribute exists.</td>
</tr>
<tr>
<td>insertBefore(XMLNode,XMLNode):Object</td>
<td>Object</td>
<td>Inserts a newChild node before the existing child node refChild. If refChild is null, the newChild node is inserted at the end of list of child nodes. If the newChild node exists in the tree, it is first removed.</td>
</tr>
<tr>
<td>normalize():Object</td>
<td>Object</td>
<td>Normalizes the node.</td>
</tr>
<tr>
<td>removeAttribute(String):Object</td>
<td>Object</td>
<td>Removes an attribute with a given name.</td>
</tr>
<tr>
<td>removeChild(XMLNode):Object</td>
<td>Object</td>
<td>Removes the child node indicated by oldChild from the list of child nodes and returns it.</td>
</tr>
<tr>
<td>replaceChild(XMLNode,XMLNode):Object</td>
<td>Object</td>
<td>Replaces the oldChild node with a newChild node in the list of child nodes and returns the oldChild node. If the newChild node exists in the tree, it is first removed.</td>
</tr>
<tr>
<td>setAttribute(String,String):Object</td>
<td>Object</td>
<td>Sets a new attribute.</td>
</tr>
</tbody>
</table>

**XMLManager Class**

The XMLManager class is the main class for creation of XML document parsers.

The XMLManager class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fromString(String):XMLDocument</td>
<td>XMLDocument</td>
<td>Gets a document for a given string content.</td>
</tr>
<tr>
<td>getDocumentContent(XMLDocument):string</td>
<td>String</td>
<td>Gets a document as a string.</td>
</tr>
<tr>
<td>loadDocument(String,boolean):XMLDocument</td>
<td>XMLDocument</td>
<td>Gets a document for a given path using the default character encoding.</td>
</tr>
<tr>
<td>loadDocumentWithEncoding (String,String,boolean):XMLDocument</td>
<td>XMLDocument</td>
<td>Gets a document for a given path using the specified character encoding.</td>
</tr>
<tr>
<td>saveDocument(XMLDocument,String,String):object</td>
<td>Object</td>
<td>Saves a document to a given path using the default character encoding.</td>
</tr>
<tr>
<td>saveDocumentWithEncoding (XMLDocument,String,String,String):object</td>
<td>Object</td>
<td>Saves a document to a given path using the specified character encoding.</td>
</tr>
</tbody>
</table>

**XMLNamedNodeMap Class**

The XMLNamedNodeMap class is the main class for node maps, usually used for attributes of an element.

The XMLNamedNodeMap class defines the following attribute.
The `XMLNamedNodeMap` class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getNamedItem(string):XMLNode</code></td>
<td><code>XMLNode</code></td>
<td>Retrieves a node specified by name.</td>
</tr>
<tr>
<td><code>item(number):XMLNode</code></td>
<td><code>XMLNode</code></td>
<td>Retrieves a child node at index.</td>
</tr>
</tbody>
</table>

### XMLNode Class

The `XMLNode` class is the main node class.

The `XMLNode` class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nodeName</code></td>
<td><code>String</code></td>
<td>The node name</td>
</tr>
<tr>
<td><code>nodeValue</code></td>
<td><code>String</code></td>
<td>The node value</td>
</tr>
<tr>
<td><code>parentNode</code></td>
<td><code>Object</code></td>
<td>The parent node (<code>XMLNode</code>)</td>
</tr>
</tbody>
</table>

The `XMLNode` class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appendChild(XMLNode):Object</code></td>
<td><code>Object</code></td>
<td>Adds a <code>newChild</code> node to the end of the list of child nodes of this node. If the <code>newChild</code> node exists in the tree, it is first removed.</td>
</tr>
<tr>
<td><code>cloneNode(boolean):Object</code></td>
<td><code>Object</code></td>
<td>Clones a node.</td>
</tr>
<tr>
<td><code>getChildNodes():XMLNodeList</code></td>
<td><code>XMLNodeList</code></td>
<td>Gets all child nodes.</td>
</tr>
<tr>
<td><code>insertBefore(XMLNode,XMLNode):Object</code></td>
<td><code>Object</code></td>
<td>Inserts a <code>newChild</code> node before the existing child node <code>refChild</code>. If <code>refChild</code> is null, the <code>newChild</code> node is inserted at the end of list of child nodes. If the <code>newChild</code> node exists in the tree, it is first removed.</td>
</tr>
<tr>
<td><code>normalize():Object</code></td>
<td><code>Object</code></td>
<td>Normalizes the node.</td>
</tr>
<tr>
<td><code>removeChild(XMLNode):Object</code></td>
<td><code>Object</code></td>
<td>Removes the child node indicated by <code>oldChild</code> from the list of child nodes and returns it.</td>
</tr>
<tr>
<td><code>insertBefore(XMLNode,XMLNode):Object</code></td>
<td><code>Object</code></td>
<td>Inserts a <code>newChild</code> node before the existing child node <code>refChild</code>. If <code>refChild</code> is null, the <code>newChild</code> node is inserted at the end of list of child nodes. If the <code>newChild</code> node exists in the tree, it is first removed.</td>
</tr>
<tr>
<td><code>replaceChild(XMLNode,XMLNode):Object</code></td>
<td><code>Object</code></td>
<td>Replaces the <code>oldChild</code> node with a <code>newChild</code> node in the list of child nodes and returns the <code>oldChild</code> node. If the <code>newChild</code> node exists in the tree, it is first removed.</td>
</tr>
</tbody>
</table>
**XMLNodeList Class**

The `XMLNodeList` class is the main class for node lists.

The `XMLNodeList` class defines the following attribute.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>Number</td>
<td>The length of a list</td>
</tr>
</tbody>
</table>

The `XMLNodeList` class defines the following method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>item(number):Object</code></td>
<td>Object</td>
<td>Retrieves a child node at index.</td>
</tr>
</tbody>
</table>

**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.

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 vCenter Orchestrator, Setting Server File System Access from Workflows and JavaScript*.

**Create a Simple XML Document**

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

**Prerequisites**

- Log in to the Orchestrator client as a user who can 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.

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

- Log in to the Orchestrator client as a user who can 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

- Log in to the Orchestrator client as a user who can 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.

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

- Log in to the Orchestrator client as a user who can 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.

The workflow creates a DTD, an XML, and a CSS file, appends the stylesheet, and stores the files in the specified
folder.
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,” on page 71
- “Mail Plug-In Scripting API,” on page 72
- “Email Scripting Examples,” on page 72
- “Using the Mail Plug-In Sample Workflows,” on page 73

### Define the Default SMTP Connection

The Mail plug-in is installed with Orchestrator Server and is used for email notifications. The only option available for this plug-in is to use default values for new mail messages. You can set the default email account.

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

**Procedure**

1. Log in to the Orchestrator configuration interface as vmware.
2. Click Mail.
3. Select the Define default values check box and fill in the required text boxes.

<table>
<thead>
<tr>
<th>Text box</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP host</td>
<td>Enter the IP address or domain name of your SMTP server.</td>
</tr>
<tr>
<td>SMTP port</td>
<td>Enter a port number to match your SMTP configuration. The default SMTP port is 25.</td>
</tr>
<tr>
<td>User name</td>
<td>Enter a valid email account. This is the email account Orchestrator uses to send emails.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the password associated with the user name.</td>
</tr>
<tr>
<td>From name and address</td>
<td>Enter the sender information to appear in all emails sent by Orchestrator.</td>
</tr>
</tbody>
</table>

4. Click Apply changes.
Mail Plug-In Scripting API

The Mail scripting API contains one class, with its respective attributes and methods, that allow vCenter Orchestrator workflows to send email messages. You can use the API to develop custom workflows.

EmailMessage Class

The EmailMessage class is the main class for creating email messages that you want to send from workflows. The EmailMessage class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bccAddress</td>
<td>String</td>
<td>Comma-separated list of blind carbon copy email address entries</td>
</tr>
<tr>
<td>ccAddress</td>
<td>String</td>
<td>Comma-separated list of carbon copy email address entries</td>
</tr>
<tr>
<td>fromAddress</td>
<td>String</td>
<td>Sender's email address</td>
</tr>
<tr>
<td>fromName</td>
<td>String</td>
<td>Sender's name</td>
</tr>
<tr>
<td>password</td>
<td>String</td>
<td>Password for authentication</td>
</tr>
<tr>
<td>smtpHost</td>
<td>String</td>
<td>SMTP host name or IP address</td>
</tr>
<tr>
<td>smtpPort</td>
<td>Number</td>
<td>SMTP port number (default is 25)</td>
</tr>
<tr>
<td>subject</td>
<td>String</td>
<td>Email subject</td>
</tr>
<tr>
<td>toAddress</td>
<td>String</td>
<td>Comma-separated list of email address entries</td>
</tr>
<tr>
<td>username</td>
<td>String</td>
<td>User name for authentication</td>
</tr>
</tbody>
</table>

The EmailMessage class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addMimePart(Object,String):void</td>
<td>Void</td>
<td>Adds a MIME part to the message.</td>
</tr>
<tr>
<td>sendMessage():void</td>
<td>Void</td>
<td>Sends an email message.</td>
</tr>
</tbody>
</table>

Email Scripting Examples

Workflow scripted elements can include scripting of common email-related tasks. You can cut, paste, and adapt these examples into your scripted elements.

When you run a mail workflow, it uses the default mail server configuration that you set in the Orchestrator configuration interface. You can override the default values by using input parameters, or by defining custom values in workflow scripted elements.

Obtain an Email Address

The following JavaScript example obtains the email address of the current owner of a running script.

```javascript
var emailAddress = Server.getRunningUser().emailAddress ;
```
Send an Email

The following JavaScript example sends an email to the defined recipient, through an SMTP server, with the defined content.

```javascript
var message = new EmailMessage();
message.smtpHost = "smtpHost";
message.subject = "my subject";
message.toAddress = "receiver@vmware.com";
message.fromAddress = "sender@vmware.com";
message.addMimePart("This is a simple message","text/html");
message.sendMessage();
```

Using the Mail Plug-In Sample Workflows

You can call the sample workflows of the Mail plug-in from custom workflows to implement email functionality. You can run an example workflow to test interaction with email.

Access the Mail Plug-In Sample Workflows

You must use the Orchestrator client to access the Mail plug-in sample workflows.

Prerequisites

- Configure the Mail plug-in in the Orchestrator configuration interface.
- Log in to the Orchestrator client as a user who can 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 the sample workflows.

Mail Plug-In Sample Workflows

You can integrate the sample Mail plug-in workflows in custom workflows.

<table>
<thead>
<tr>
<th>Workflow Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve messages</td>
<td>Retrieves the messages of a given email account by using the POP3 protocol.</td>
</tr>
<tr>
<td>Send interaction</td>
<td>Sends an email to answer a user interaction. The email body contains both the direct answer URL, and an interaction URL to process this request. If optional parameters are not specified, the workflow uses the default values set in the Orchestrator configuration interface.</td>
</tr>
<tr>
<td>Send notification</td>
<td>Sends an email with specified content to a given email address. If optional parameters are not specified, the workflow uses the default values set in the Orchestrator configuration interface.</td>
</tr>
<tr>
<td>Send notification to mailing list</td>
<td>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 in the Orchestrator configuration interface.</td>
</tr>
</tbody>
</table>
Test an Example Interaction with Email

You can run a workflow from the Orchestrator client to send an email to respond to a query, known as a user interaction.

The workflow uses the default mail server configuration that you set in the Orchestrator configuration interface. See “Define the Default SMTP Connection,” on page 71.

Prerequisites

- Configure the Mail plug-in in the Orchestrator configuration interface.
- Log in to the Orchestrator client as a user who can run Mail workflows.

Procedure

1. Click the Workflows view in the Orchestrator client.
2. In the workflows hierarchical list, open Library > Mail to navigate to the Example interaction with email workflow.
3. Right-click the Example interaction with email workflow and select Start workflow.
4. Provide the required information.
   a. Type a recipient address.
   b. Select an LDAP group of users who are authorized to answer the query.
5. Click Submit to run the workflow.

The workflow suspends its run and sends an email to the given address. The email body contains a link to the weboperator Web view. If weboperator is running, the user can answer the request for interaction directly in weboperator, allowing the workflow to finish its run.
Using the Net Plug-In

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

Net Plug-In Scripting API

The Net scripting API contains classes, with their respective attributes, methods, and constructors, that allow vCenter Orchestrator workflows to implement Telnet, POP3, and FTP functionality. You can use the API to develop custom workflows.

FTPClient Class

The FTPClient class is the main class for implementing FTP functionality into workflows.

The FTPClient class defines the FTPClient():FTPClient constructor.

The FTPClient class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectTimeout</td>
<td>number</td>
<td>Socket connection timeout</td>
</tr>
<tr>
<td>cwd</td>
<td>String</td>
<td>Current working directory</td>
</tr>
<tr>
<td>output</td>
<td>String</td>
<td>Command output if any</td>
</tr>
<tr>
<td>replyCode</td>
<td>Number</td>
<td>Last reply code</td>
</tr>
<tr>
<td>soLinger</td>
<td>Number</td>
<td>Socket SO Linger (null if disabled)</td>
</tr>
<tr>
<td>state</td>
<td>String</td>
<td>Run state</td>
</tr>
<tr>
<td>tcpNoDelay</td>
<td>Boolean</td>
<td>Socket TCP no delay</td>
</tr>
<tr>
<td>timeout</td>
<td>Number</td>
<td>Socket timeout</td>
</tr>
</tbody>
</table>

The FTPClient class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connect(String,number) : Object</td>
<td>Object</td>
<td>Connects the client to a host.</td>
</tr>
<tr>
<td>disconnect() : Object</td>
<td>Object</td>
<td>Logs out and disconnects the current session if open.</td>
</tr>
<tr>
<td>enableSSL(boolean) : Object</td>
<td>Object</td>
<td>Enables SSL.</td>
</tr>
<tr>
<td>executeCommand(String,String) : Number</td>
<td>number</td>
<td>Runs a command.</td>
</tr>
<tr>
<td>Method</td>
<td>Returns</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>getFile(String, String)</td>
<td>Number</td>
<td>Copies a file from a remote host to the Orchestrator server.</td>
</tr>
<tr>
<td>getStatus()</td>
<td>String</td>
<td>Provides the status information returned by the server.</td>
</tr>
<tr>
<td>getString(String)</td>
<td>String</td>
<td>Gets the contents of a remote file.</td>
</tr>
<tr>
<td>isAvailable()</td>
<td>boolean</td>
<td>Checks the socket to test whether it is available for use.</td>
</tr>
<tr>
<td>isConnected()</td>
<td>boolean</td>
<td>Provides information about whether the client is currently connected to a server.</td>
</tr>
<tr>
<td>listAll(String)</td>
<td>Array of String</td>
<td>Lists all files and directories in a path.</td>
</tr>
<tr>
<td>listDir(String)</td>
<td>Array of String</td>
<td>Lists directories in a path.</td>
</tr>
<tr>
<td>listFile(String)</td>
<td>Array of String</td>
<td>Lists files in a path.</td>
</tr>
<tr>
<td>login(String, String, String)</td>
<td>boolean</td>
<td>Logs in to an FTP server.</td>
</tr>
<tr>
<td>putFile(String, String)</td>
<td>Number</td>
<td>Copies a file from the Orchestrator server to a remote host.</td>
</tr>
<tr>
<td>putString(String, String)</td>
<td>Number</td>
<td>Adds a string of content to a remote file.</td>
</tr>
</tbody>
</table>

**POP3Client Class**

The POP3Client class is the main class for implementing POP3 functionality into workflows.

The POP3Client class defines the POP3Client():POP3Client constructor.

The POP3Client class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectTimeout</td>
<td>Number</td>
<td>Socket connect timeout</td>
</tr>
<tr>
<td>output</td>
<td>String</td>
<td>Command output if any</td>
</tr>
<tr>
<td>soLinger</td>
<td>Number</td>
<td>Socket SO Linger (null if disabled)</td>
</tr>
<tr>
<td>state</td>
<td>String</td>
<td>Connection state</td>
</tr>
<tr>
<td>tcpNoDelay</td>
<td>Boolean</td>
<td>Socket TCP no delay</td>
</tr>
<tr>
<td>timeout</td>
<td>Number</td>
<td>Socket timeout</td>
</tr>
</tbody>
</table>

The POP3Client class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connect(String, number)</td>
<td>Object</td>
<td>Connects the client to a host.</td>
</tr>
<tr>
<td>disconnect()</td>
<td>Object</td>
<td>Disconnects the client from the server.</td>
</tr>
<tr>
<td>enableSSL(boolean)</td>
<td>Object</td>
<td>Enables SSL.</td>
</tr>
<tr>
<td>isAvailable()</td>
<td>Boolean</td>
<td>Makes checks on the socket to test if it is available for use.</td>
</tr>
<tr>
<td>isConnected()</td>
<td>Boolean</td>
<td>Provides information if the client is currently connected to a server.</td>
</tr>
</tbody>
</table>
Method | Returns | Description
---|---|---
listMessages():POP3Message[] | Array of POP3Message | Lists all messages. If there are no messages, the method returns a zero length array. If the operation fails, the method returns null.

login(String,String):boolean | Boolean | Logs in on a POP3 server with the given user name and password. Before attempting to log in, you must connect to the server with the connect() method.

loginWithSecret(String,String,String):boolean | Boolean | Logs in on a POP3 server with the given user name and authentication information.

logout():Object | Object | Logs out of a POP3 server. To fully disconnect from the server, you must call the disconnect() method.

noop():Object | Object | Sends a NOOP command to a POP3 server. This is useful for keeping the connection alive.

reset():Object | Object | Resets a POP3 session. This is useful for undoing any message deletions that might have been performed.

### POP3Message Class

The POP3Message class provides POP3 email functionality.

The POP3Message class defines the following attributes.

| Attribute | Returns | Description
---|---|---
body | String | Message body
from | String | Sender
id | Number | Message ID in current session
subject | String | Message subject
to | String | Recipient

The POP3Message class defines the following methods.

| Method | Returns | Description
---|---|---
deleteFromServer():Object | Object | Deletes a message from the POP3 server. The server marks the message for deletion but does not delete it immediately. If you decide to unmark the message, you must use call the reset() command from the POP3Client class. The server deletes all messages marked for deletion when you call the logout() command from the POP3Client class.

getHeader(String):string | String | Gets the header value.
TelnetClient Class

The TelnetClient class is the main class for implementing Telnet functionality into workflows.

The TelnetClient class defines the TelnetClient(String):TelnetClient constructor.

The TelnetClient class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>soLinger</td>
<td>Number</td>
<td>Socket SO Linger (null if disabled)</td>
</tr>
<tr>
<td>tcpNoDelay</td>
<td>Boolean</td>
<td>Socket TCP no delay</td>
</tr>
<tr>
<td>timeout</td>
<td>Number</td>
<td>Socket timeout</td>
</tr>
</tbody>
</table>

The TelnetClient class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connect(String,number):Object</td>
<td>Object</td>
<td>Connects the client to a host.</td>
</tr>
<tr>
<td>disconnect():Object</td>
<td>Object</td>
<td>Disconnects the client.</td>
</tr>
<tr>
<td>enableSSL(boolean):Object</td>
<td>Object</td>
<td>Enables SSL.</td>
</tr>
<tr>
<td>receiveAsBinary():Object[]</td>
<td>Array of Object</td>
<td>Gets a response as an array of numbers.</td>
</tr>
<tr>
<td>receiveAsString():string</td>
<td>String</td>
<td>Gets a response as a string.</td>
</tr>
<tr>
<td>sendBinary(Object[]):Object</td>
<td>Object</td>
<td>Sends binary information as an array of numbers.</td>
</tr>
<tr>
<td>sendString(String):Object</td>
<td>Object</td>
<td>Sends a string command.</td>
</tr>
<tr>
<td>waitForData(Number):boolean</td>
<td>Boolean</td>
<td>Waits for data to become available and returns true if data is available or false if the request times out.</td>
</tr>
</tbody>
</table>
Using the Enumeration Plug-In

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

This chapter includes the following topics:

- “Enumeration Plug-In Scripting API,” on page 79
- “Time Zone Codes,” on page 80

Enumeration Plug-In Scripting API

The Enums scripting API contains common enumerated types.

Table 9-1. Common Enumerated Types

<table>
<thead>
<tr>
<th>Enumeration Name</th>
<th>Possible Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enums:Backup</td>
<td><strong>All</strong>: Configuration file and all disks</td>
<td>Kind of backup</td>
</tr>
<tr>
<td></td>
<td><strong>Config</strong>: Configuration file only</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No backup at all</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System</strong>: Configuration file and first disk</td>
<td></td>
</tr>
<tr>
<td>Enums:Disk</td>
<td><strong>0</strong>: No disk</td>
<td>Possible disk sizes</td>
</tr>
<tr>
<td></td>
<td><strong>2</strong>: 2GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>4</strong>: 4GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>8</strong>: 8GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>12</strong>: 12GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>16</strong>: 16GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>20</strong>: 20GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>60</strong>: 60GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>80</strong>: 80GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>100</strong>: 100GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>120</strong>: 120GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>160</strong>: 160GB</td>
<td></td>
</tr>
<tr>
<td>Enums:Environment</td>
<td><strong>Validation</strong>: Validation environments</td>
<td>Type of deployment environment</td>
</tr>
<tr>
<td></td>
<td><strong>Test</strong>: Test environments</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Development</strong>: Development environments</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Production</strong>: Production environments</td>
<td></td>
</tr>
<tr>
<td>Enums:MSTimeZone</td>
<td>See “Time Zone Codes,” on page 80.</td>
<td>Time zones</td>
</tr>
</tbody>
</table>
### Table 9-1. Common Enumerated Types (Continued)

<table>
<thead>
<tr>
<th>Enumeration Name</th>
<th>Possible Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enums:Performance</td>
<td>Low: Low (20%)</td>
<td>Type of performance</td>
</tr>
<tr>
<td></td>
<td>Medium: Medium (60%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>High: High (80%)</td>
<td></td>
</tr>
<tr>
<td>Enums:RAM</td>
<td>128: 128MB</td>
<td>Possible RAM sizes</td>
</tr>
<tr>
<td></td>
<td>256: 256MB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>512: 512MB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1024: 1GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2048: 2GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3584: 3.5GB (ESX 2 maximum)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4096: 4GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6144: 6GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8192: 8GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10240: 10GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12288: 12GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14336: 14GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16384: 16GB</td>
<td></td>
</tr>
<tr>
<td>Enums:vCPU</td>
<td>1: Virtual machine with 1 virtual CPU</td>
<td>Number of virtual CPUs</td>
</tr>
<tr>
<td></td>
<td>2: Virtual machine with 2 virtual CPUs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3: Virtual machine with 3 virtual CPUs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4: Virtual machine with 4 virtual CPUs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5: Virtual machine with 5 virtual CPUs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6: Virtual machine with 6 virtual CPUs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7: Virtual machine with 7 virtual CPUs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8: Virtual machine with 8 virtual CPUs</td>
<td></td>
</tr>
</tbody>
</table>

### Time Zone Codes

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

<table>
<thead>
<tr>
<th>Time Zone Code</th>
<th>Time Zone Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Dateline Standard Time</td>
<td>(GMT-12:00) International Date Line West</td>
</tr>
<tr>
<td>001</td>
<td>Samoa Standard Time</td>
<td>(GMT-11:00) Midway Island, Samoa</td>
</tr>
<tr>
<td>002</td>
<td>Hawaiian Standard Time</td>
<td>(GMT-10:00) Hawaii</td>
</tr>
<tr>
<td>003</td>
<td>Alaskan Standard Time</td>
<td>(GMT-09:00) Alaska</td>
</tr>
<tr>
<td>004</td>
<td>Pacific Standard Time</td>
<td>(GMT-08:00) Pacific Time (US and Canada); Tijuana</td>
</tr>
<tr>
<td>010</td>
<td>Mountain Standard Time</td>
<td>(GMT-07:00) Mountain Time (US and Canada)</td>
</tr>
<tr>
<td>013</td>
<td>Mexico Standard Time 2</td>
<td>(GMT-07:00) Chihuahua, La Paz, Mazatlan</td>
</tr>
<tr>
<td>015</td>
<td>U.S. Mountain Standard Time</td>
<td>(GMT-07:00) Arizona</td>
</tr>
<tr>
<td>020</td>
<td>Central Standard Time</td>
<td>(GMT-06:00) Central Time (US and Canada)</td>
</tr>
<tr>
<td>025</td>
<td>Canada Central Standard Time</td>
<td>(GMT-06:00) Saskatchewan</td>
</tr>
<tr>
<td>Time Zone Code</td>
<td>Time Zone Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>030</td>
<td>Mexico Standard Time</td>
<td>(GMT-06:00) Guadalajara, Mexico City, Monterrey</td>
</tr>
<tr>
<td>033</td>
<td>Central America Standard Time</td>
<td>(GMT-06:00) Central America</td>
</tr>
<tr>
<td>035</td>
<td>Eastern Standard Time</td>
<td>(GMT-05:00) Eastern Time (US and Canada)</td>
</tr>
<tr>
<td>040</td>
<td>U.S. Eastern Standard Time</td>
<td>(GMT-05:00) Indiana (East)</td>
</tr>
<tr>
<td>045</td>
<td>S.A. Pacific Standard Time</td>
<td>(GMT-05:00) Bogota, Lima, Quito</td>
</tr>
<tr>
<td>050</td>
<td>Atlantic Standard Time</td>
<td>(GMT-04:00) Atlantic Time (Canada)</td>
</tr>
<tr>
<td>055</td>
<td>S.A. Western Standard Time</td>
<td>(GMT-04:00) Caracas, La Paz</td>
</tr>
<tr>
<td>056</td>
<td>Pacific S.A. Standard Time</td>
<td>(GMT-04:00) Santiago</td>
</tr>
<tr>
<td>060</td>
<td>Newfoundland and Labrador Standard Time</td>
<td>(GMT-03:30) Newfoundland and Labrador</td>
</tr>
<tr>
<td>065</td>
<td>E. South America Standard Time</td>
<td>(GMT-03:00) Brasilia</td>
</tr>
<tr>
<td>070</td>
<td>S.A. Eastern Standard Time</td>
<td>(GMT-03:00) Buenos Aires, Georgetown</td>
</tr>
<tr>
<td>073</td>
<td>Greenland Standard Time</td>
<td>(GMT-03:00) Greenland</td>
</tr>
<tr>
<td>075</td>
<td>Mid-Atlantic Standard Time</td>
<td>(GMT-02:00) Mid-Atlantic</td>
</tr>
<tr>
<td>080</td>
<td>Azores Standard Time</td>
<td>(GMT-01:00) Azores</td>
</tr>
<tr>
<td>083</td>
<td>Cape Verde Standard Time</td>
<td>(GMT-01:00) Cape Verde Islands</td>
</tr>
<tr>
<td>090</td>
<td>Greenwich Standard Time</td>
<td>(GMT) Casablanca, Monrovia</td>
</tr>
<tr>
<td>095</td>
<td>Central Europe Standard Time</td>
<td>(GMT+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague</td>
</tr>
<tr>
<td>100</td>
<td>Central European Standard Time</td>
<td>(GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb</td>
</tr>
<tr>
<td>105</td>
<td>Romance Standard Time</td>
<td>(GMT+01:00) Brussels, Copenhagen, Madrid, Paris</td>
</tr>
<tr>
<td>110</td>
<td>W. Europe Standard Time</td>
<td>(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna</td>
</tr>
<tr>
<td>113</td>
<td>W. Central Africa Standard Time</td>
<td>(GMT+01:00) West Central Africa</td>
</tr>
<tr>
<td>115</td>
<td>E. Europe Standard Time</td>
<td>(GMT+02:00) Bucharest</td>
</tr>
<tr>
<td>120</td>
<td>Egypt Standard Time</td>
<td>(GMT+02:00) Cairo</td>
</tr>
<tr>
<td>125</td>
<td>FLE Standard Time</td>
<td>(GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius</td>
</tr>
<tr>
<td>130</td>
<td>GTB Standard Time</td>
<td>(GMT+02:00) Athens, Istanbul, Minsk</td>
</tr>
<tr>
<td>135</td>
<td>Israel Standard Time</td>
<td>(GMT+02:00) Jerusalem</td>
</tr>
<tr>
<td>140</td>
<td>South Africa Standard Time</td>
<td>(GMT+02:00) Harare, Pretoria</td>
</tr>
<tr>
<td>145</td>
<td>Russian Standard Time</td>
<td>(GMT+03:00) Moscow, St. Petersburg, Volgograd</td>
</tr>
<tr>
<td>150</td>
<td>Arab Standard Time</td>
<td>(GMT+03:00) Kuwait, Riyadh</td>
</tr>
<tr>
<td>155</td>
<td>E. Africa Standard Time</td>
<td>(GMT+03:00) Nairobi</td>
</tr>
<tr>
<td>158</td>
<td>Arabic Standard Time</td>
<td>(GMT+03:00) Baghdad</td>
</tr>
<tr>
<td>160</td>
<td>Iran Standard Time</td>
<td>(GMT+03:30) Tehran</td>
</tr>
<tr>
<td>Time Zone Code</td>
<td>Time Zone Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>165</td>
<td>Arabian Standard Time</td>
<td>(GMT+04:00) Abu Dhabi, Muscat</td>
</tr>
<tr>
<td>170</td>
<td>Caucasus Standard Time</td>
<td>(GMT+04:00) Baku, Tbilisi, Yerevan</td>
</tr>
<tr>
<td>175</td>
<td>Transitional Islamic State of Afghanistan Standard Time</td>
<td>(GMT+04:30) Kabul</td>
</tr>
<tr>
<td>180</td>
<td>Ekaterinburg Standard Time</td>
<td>(GMT+05:00) Ekaterinburg</td>
</tr>
<tr>
<td>185</td>
<td>West Asia Standard Time</td>
<td>(GMT+05:00) Islamabad, Karachi, Tashkent</td>
</tr>
<tr>
<td>190</td>
<td>India Standard Time</td>
<td>(GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi</td>
</tr>
<tr>
<td>193</td>
<td>Nepal Standard Time</td>
<td>(GMT+05:45) Kathmandu</td>
</tr>
<tr>
<td>195</td>
<td>Central Asia Standard Time</td>
<td>(GMT+06:00) Astana, Dhaka</td>
</tr>
<tr>
<td>200</td>
<td>Sri Lanka Standard Time</td>
<td>(GMT+06:00) Sri Jayawardenepura</td>
</tr>
<tr>
<td>201</td>
<td>N. Central Asia Standard Time</td>
<td>(GMT+06:00) Almaty, Novosibirsk</td>
</tr>
<tr>
<td>203</td>
<td>Myanmar Standard Time</td>
<td>(GMT+06:30) Yangon (Rangoon)</td>
</tr>
<tr>
<td>205</td>
<td>S.E. Asia Standard Time</td>
<td>(GMT+07:00) Bangkok, Hanoi, Jakarta</td>
</tr>
<tr>
<td>207</td>
<td>North Asia Standard Time</td>
<td>(GMT+07:00) Krasnoyarsk</td>
</tr>
<tr>
<td>210</td>
<td>China Standard Time</td>
<td>(GMT+08:00) Beijing, Chongqing, Hong Kong SAR, Urumqi</td>
</tr>
<tr>
<td>215</td>
<td>Singapore Standard Time</td>
<td>(GMT+08:00) Kuala Lumpur, Singapore</td>
</tr>
<tr>
<td>220</td>
<td>Taipei Standard Time</td>
<td>(GMT+08:00) Taipei</td>
</tr>
<tr>
<td>225</td>
<td>W. Australia Standard Time</td>
<td>(GMT+08:00) Perth</td>
</tr>
<tr>
<td>227</td>
<td>North Asia East Standard Time</td>
<td>(GMT+08:00) Irkutsk, Ulaan Batar</td>
</tr>
<tr>
<td>230</td>
<td>Korea Standard Time</td>
<td>(GMT+09:00) Seoul</td>
</tr>
<tr>
<td>235</td>
<td>Tokyo Standard Time</td>
<td>(GMT+09:00) Osaka, Sapporo, Tokyo</td>
</tr>
<tr>
<td>240</td>
<td>Yakutsk Standard Time</td>
<td>(GMT+09:00) Yakutsk</td>
</tr>
<tr>
<td>245</td>
<td>A.U.S. Central Standard Time</td>
<td>(GMT+09:30) Darwin</td>
</tr>
<tr>
<td>250</td>
<td>Cen. Australia Standard Time</td>
<td>(GMT+09:30) Adelaide</td>
</tr>
<tr>
<td>255</td>
<td>A.U.S. Eastern Standard Time</td>
<td>(GMT+10:00) Canberra, Melbourne, Sydney</td>
</tr>
<tr>
<td>260</td>
<td>E. Australia Standard Time</td>
<td>(GMT+10:00) Brisbane</td>
</tr>
<tr>
<td>265</td>
<td>Tasmania Standard Time</td>
<td>(GMT+10:00) Hobart</td>
</tr>
<tr>
<td>270</td>
<td>Vladivostok Standard Time</td>
<td>(GMT+10:00) Vladivostok</td>
</tr>
<tr>
<td>275</td>
<td>West Pacific Standard Time</td>
<td>(GMT+10:00) Guam, Port Moresby</td>
</tr>
<tr>
<td>280</td>
<td>Central Pacific Standard Time</td>
<td>(GMT+11:00) Magadan, Solomon Islands, New Caledonia</td>
</tr>
<tr>
<td>285</td>
<td>Fiji Islands Standard Time</td>
<td>(GMT+12:00) Fiji Islands, Kamchatka, Marshall Islands</td>
</tr>
<tr>
<td>290</td>
<td>New Zealand Standard Time</td>
<td>(GMT+12:00) Auckland, Wellington</td>
</tr>
<tr>
<td>300</td>
<td>Tonga Standard Time</td>
<td>(GMT+13:00) Nuku'alofa</td>
</tr>
</tbody>
</table>
Using the Workflow Documentation Plug-In

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 Documentation Plug-In Scripting API,” on page 83
- “Workflow Library for the Workflow Documentation Plug-In,” on page 83
- “Generate Workflow Documentation,” on page 84

Workflow Documentation Plug-In Scripting API

The Workflow Documentation scripting API contains one class, with its respective methods. You can use the API to develop custom workflows.

WorkflowDocumentationService Class

The WorkflowDocumentationService class is the main Workflow Documentation class.

The WorkflowDocumentationService class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getWorkflowCategoryDocumentation</td>
<td>MimeAttachment</td>
<td>Gets documentation about a workflow category.</td>
</tr>
<tr>
<td>getWorkflowDocumentation</td>
<td>MimeAttachment</td>
<td>Gets documentation about a workflow.</td>
</tr>
</tbody>
</table>

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.

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.

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

A  
add a database 50  
add tables to a database 50  
adding a database 38  
adding tables to a database 38  
API Explorer, accessing 14  
audience 7  

B  
basic host management workflows 23  
basic virtual machine workflows 28  
batch workflows 19  

C  
clone workflows 29  
cluster and compute resource workflows 19  
custom attributes workflows 21  

d  
database plug-in, workflow library 50  
Database plug-in  
  scripting API 39  
  usage 37  
Database plug-in API  
  connecting to a database 40  
  Connection class 40  
  database connection parameters 41  
  JDBCConnection class 41  
  PreparedStatement class 41  
  ResultSet class 41  
  ResultSetMetaData class 42  
  SQL:ActiveRecord 39  
  SQL:Column 40  
  SQL:Database 40  
  SQL:Table 40  
  SQLActiveRecord class 42  
  SQLColumn class 43  
  SQLDatabase class 43  
  SQLDatabaseManager class 44  
  SQLTable class 45  
Database plug-in sample workflows 45  
datacenter folder management folder 22  
datacenter workflows 21  
datastore and files workflows 22  
default plug-ins 12  
device management workflows 31  
distributed virtual port group workflows 24  
distributed virtual switch workflows 25  
documentation 84  

E  
Enumeration plug-in  
  Enums:MSTimeZone values 80  
  scripting API 79  
  usage 79  

generate, workflow documentation 84  
generating CRUD workflows 50  
guest operation files workflows 20  
guest operation processes workflows 21  

H  
host folder management workflows 22  
host management registration workflows 24  

I  
introduction 11  

J  
JDBC connection, testing 46  
JDBC dropping, testing 49  
JDBC entry deletion, testing 49  
JDBC full cycle, testing 49  
JDBC plug-in  
  scripting API 39  
  usage 37  
JDBC row insertion, testing 47  
JDBC row selection, testing 47  
JDBC sample workflows 45  
JDBC table creation, testing 46  
JDBC URL generator 45  

L  
linked clone workflows 29  
Linux customization clone workflows 30  
load balancing 71  

M  
Mail plug-in  
  sample workflows access 73
scripting API 72
usage 71
Mail plug-in API, EmailMessage class 72
Mail plug-in sample workflows 73
Mail plug-in workflows, example interaction with email 74
move and migrate workflows 32

N
Net plug-in
scripting API 75
usage 75
Net plug-in API
FTPClient class 75
POP3Client class 76
POP3Message class 77
TelnetClient class 78
networking workflows 24

O
Orchestrator architecture 11
overview of
Enumeration plug-in 12
Mail plug-in 12
Net plug-in 12
overview of 12
SQL plug-in 12
SSH plug-in 12
vCenter Server 5.1 plug-in 12
vCO Library plug-in 12
vCO Weboperator plug-in 12
XML plug-in 12

P
PDF 84
plug-ins configuration
Mail plug-in 71
SSH plug-in 53
vCenter Server plug-in 17
power host management workflows 23
power management workflows 33
precompiled SQL statement 41

R
remove a database 50
remove a table from a database 50
resource pool workflows 26

S
scripting, email examples 72
scripting API, Workflow Documentation plug-in 83
SFTP 60, 61
SMTP connection 71
snapshot workflows 33
SQL operations 50
SQL plug-in
PreparedStatement class 41
scripting API 39
SQLActiveRecord class 42
usage 37
SQL plug-in API
Connection class 40
JDBCConnection class 41
ResultSet class 41
ResultSetMetaData class 42
SQL:ActiveRecord 39
SQL:ColumnType 40
SQL:Database 40
SQLColumn class 43
SQLDatabase class 43
SQLDatabaseManager class 44
SQLTable class 45
SQL plug-in configuration 37
SSH commands, running 60
SSH file copy
SCP get command 60
SCP put command 61
SSH plug-in
scripting API 54
usage 53
SSH plug-in API
KeyPairManager class 54
managing private and public SSH keys 54
SSH:File type 54
SSH:Folder type 54
SSH:RootFolder type 54
SSH:SshConnection type 54
SSHCommand class 55
SSHFile class 56
SSHFolder class 56
SSHSession class 57
SSH plug-in sample workflows 58
SSH workflows
changing a key pair passphrase 59
generating a key pair 58
registering a public key 59
standard virtual switch workflows 25
storage DRS workflows 27
storage workflows 26

T
tools clone workflows 30

U
update a database 50
updated information 9
updating a database 39

V
vCenter Server 5.1 plug-in
Inventory 18
library accessing 18
scripting API 18
usage 17
workflow library 19
workflow library access 18
workflow library accessing 18
vCO Library plug-in
usage 35
workflows 35
virtual machine folder management workflows 23
VMware Tools workflows 34

W
windows customization clone workflows 31
workflow documentation 84
workflow documentation plug-in 83
Workflow Documentation plug-in scripting API 83
workflow library 83
Workflow Documentation plug-in API, SSHFolder class 83
workflows
basic host management 23
basic virtual machine 28
batch 19
clone 29
custom attributes 21
database configuration 37
datacenter 21
datacenter folder management 22
datasstore and files 22
device management 31
distributed virtual port group 24
distributed virtual switch 25
guest operation files 20
guest operation processes 21
host folder management 22
host management registration 24
linked clone 29
Linux customization clone 30
move and migrate 32
networking 24
other 32
power host management 23
power management 33
resource pool 26
snapshot 33
standard virtual switch 25
standard workflows 50
storage 26
Storage DRS 27
tools clone 30
virtual machine folder management 23
VMware Tools 34
windows customization clone 31

X
XML
DOM 63
E4X 63
XML plug-in
scripting API 63
usage 63
XML plug-in API
XMLDocument class 63
XMLElement class 64
XMLManager class 65
XMLNamedNodeMap class 65
XMLNode class 66
XMLNodeList class 67
XML plug-in sample workflows
creating a simple XML document 67
creating an address book 69
finding an element 68
modifying an XML document 68