Using VMware vCenter Orchestrator Plug-Ins

vCenter Orchestrator 4.2.1

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.

EN-000784-00
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

Copyright © 2008 – 2012 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at http://www.vmware.com/go/patents.

VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

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

Using VMware vCenter Orchestrator Plug-Ins  7

1 Introduction to Orchestrator Plug-Ins  9
   Orchestrator Architecture  9
   Default Orchestrator Plug-Ins  10
   Access the Orchestrator API Explorer  12

2 Using the vCenter Server 5.0.1 Plug-In  15
   Configure the vCenter Server 5.0.1 Plug-In  15
   vCenter Server 5.0.1 Plug-In Scripting API  16
   Using the vCenter Server Plug-In Inventory  16
   Access the vCenter Server Plug-In Workflow Library  16
   vCenter Server Plug-In Workflow Library  17
      Batch Workflows  17
      Cluster and Compute Resource Workflows  17
      Custom Attributes Workflows  18
      Datacenter Workflows  18
      Datastore and Files Workflows  18
      Datacenter Folder Management Workflows  19
      Host Folder Management Workflows  19
      Virtual Machine Folder Management Workflows  19
      Basic Host Management Workflows  20
      Power Host Management Workflows  20
      Host Management Registration Workflows  20
      Networking Workflows  21
      Distributed Virtual Port Group Workflows  21
      Distributed Virtual Switch Workflows  21
      Standard Virtual Switch Workflows  22
      Resource Pool Workflows  22
      Storage Workflows  23
      Storage DRS Workflows  23
      Basic Virtual Machine Management Workflows  24
      Clone Workflows  25
      Linked Clone Workflows  25
      Linux Customization Clone Workflows  26
      Tools Clone Workflows  26
      Windows Customization Clone Workflows  27
      Device Management Workflows  27
      Move and Migrate Workflows  28
      Other Workflows  28
      Power Management Workflows  29
      Snapshot Workflows  29
3 Using the vCO Library Plug-In 31
   vCO Library Plug-In Workflows 31

4 Using the Database Plug-In 33
   Database Plug-In Scripting API 33
      Connection Class 33
      JDBCConnection Class 33
      PreparedStatement Class 34
      ResultSet Class 34
   Running the JDBC Sample Workflows 35
      Generate a JDBC URL 35
      Test a JDBC Connection 36
      Create a Table by Using JDBC 36
      Insert a Row into a JDBC Table 37
      Select Rows from a JDBC Table 37
      Delete an Entry from a JDBC Table 38
      Delete All Entries from a JDBC Table 38
      Drop a JDBC Table 39
      Run a Complete JDBC Cycle 39

5 Using the SSH Plug-In 41
   Configure the SSH Plug-In 41
   SSH Plug-In Scripting API 42
      SSH:File Type 42
      SSH:Folder Type 42
      SSH:RootFolder Type 42
      SSH:SshConnection Type 42
      KeyPairManager Class 42
      SSHCommand Class 43
      SSHFile Class 44
      SSHFolder Class 44
      SSHSession Class 45
   Running the SSH Plug-In Sample Workflows 46
      Generate a Key Pair 46
      Change the Key Pair Passphrase 47
      Register an Orchestrator Public Key on an SSH Host 47
      Run an SSH Command 48
      Copy a File from an SSH Host 48
      Copy a File to an SSH Host 49

6 Using the XML Plug-In 51
   XML Plug-In Scripting API 51
      XMLDocument Class 51
      XMLElement Class 52
      XMLManager Class 53
      XMLNamedNodeMap Class 53
Using VMware vCenter Orchestrator Plug-Ins

The 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.
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 Orchestrator. Before you use some of them, for example the vCenter Server plug-in, you should configure the plug-ins.

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 9
- “Default Orchestrator Plug-Ins,” on page 10
- “Access the Orchestrator API Explorer,” on page 12

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 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>vCenter Server 5.0.1</td>
<td>Provides access to the vCenter Server 4.1 API, so that you can incorporate all of the vCenter Server objects and functions into the management processes that you use Orchestrator to automate.</td>
<td>See “Configure the vCenter Server 5.0.1 Plug-In,” on page 15.</td>
<td>See <a href="http://www.vmware.com/support/orchestrator/doc/vco_vsphere50_api/index.html">http://www.vmware.com/support/orchestrator/doc/vco_vsphere50_api/index.html</a>.</td>
<td>See <a href="http://www.vmware.com/support/orchestrator/doc/vco_vsphere50_api/index.html">http://www.vmware.com/support/orchestrator/doc/vco_vsphere50_api/index.html</a>.</td>
<td>Exposes all vCenter Server objects.</td>
</tr>
<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>
<tr>
<td>Plug-In</td>
<td>Purpose</td>
<td>Configuration</td>
<td>Input Types</td>
<td>Scripting Objects</td>
<td>Inventory</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------------</td>
<td>-------------</td>
<td>------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Database</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>None</td>
<td>Connection, JDBCConnection, PreparedStatement, ResultSet</td>
<td>Exposes nothing 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 41.</td>
<td>File, Folder, RootFolder, SshConnection</td>
<td>KeyPairManager, SSHCommand, SSHEFile, SSHFolder, SSHSession</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, XMLElement, 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 59.</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>
</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. See Administering VMware vCenter Orchestrator, Weboperator Web View.</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 5.0.1</td>
<td>o11nplugin-vsphere501.dar</td>
<td>vCenter</td>
<td>VC</td>
</tr>
<tr>
<td>vCO Library</td>
<td>o11nplugin-library.dar</td>
<td>Locking</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Orchestrator Troubleshooting</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>o11nplugin-jdbc.dar</td>
<td>JDBC</td>
<td>Database</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>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 to allow you 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

- Access the API Explorer from either the Orchestrator client or from the Scripting tabs of the workflow, policy, and action editors.
  - To access the API Explorer from the Orchestrator client, click Tools > API Explorer in the Orchestrator client tool bar.
  - To access the API Explorer from the Scripting tabs of the workflow, policy, and action editors, click Search API on the left.

The API Explorer appears, allowing you 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 5.0.1 Plug-In

The vCenter Server 5.0.1 plug-in (VMware vCenter Orchestrator plug-in for vCenter Server 5.0.1) allows interaction between vCenter Orchestrator and vCenter Server.

You can use the vCenter Server 5.0.1 plug-in to manage multiple vCenter Server hosts. You can create workflows that use the vCenter Server plug-in API to automate tasks in your vCenter Server environment.

The vCenter Server 5.0.1 plug-in maps the vCenter Server 5.0 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 5.0.1 plug-in provides a library of standard workflows that automate vCenter Server 5.0.1 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 5.0.1 Plug-In,” on page 15
- “vCenter Server 5.0.1 Plug-In Scripting API,” on page 16
- “Using the vCenter Server Plug-In Inventory,” on page 16
- “Access the vCenter Server Plug-In Workflow Library,” on page 16
- “vCenter Server Plug-In Workflow Library,” on page 17

Configure the vCenter Server 5.0.1 Plug-In

Orchestrator uses the vCenter Web Service API to control vCenter Server. You can set the parameters to enable Orchestrator to connect to your vCenter Sever instances.

Prerequisites

Import the SSL certificates for each vCenter Server instance you define. See Installing and Configuring VMware vCenter Orchestrator.

Procedure

1. Log in to the Orchestrator configuration interface as vmware.
2. Click vCenter Server 5.0.1.
3. Click New vCenter Server Host.
4. From the Available drop-down menu, select Enabled.
5. In the Host text box, type the IP address or the DNS name of the vCenter Server host.
6. In the Port text box, retain the default value, 443.
7 (Optional) Select the **Secure channel** check box to establish a secure connection to your vCenter Server host.

8 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.

9 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 administrative privileges on your vCenter Server, preferably at the top of the vCenter Server tree structure. Orchestrator uses these credentials to monitor the vCenter Web service, typically to operate Orchestrator system workflows. All other requests inherit the credentials of the user who triggers an action.

10 Select the method you use to manage user access on the vCenter Server host.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Share a unique session</strong></td>
<td>Allows Orchestrator to create only one connection to vCenter Server. Type the credentials of a user who is a vCenter Server administrator.</td>
</tr>
<tr>
<td><strong>Session per user</strong></td>
<td><strong>CAUTION</strong> Each user who logs in to Orchestrator creates a new session to vCenter Server. This might rapidly use CPU, memory, and bandwidth. Select this option if your vCenter Server is in an Active Directory domain. Make sure that the user has the necessary permissions to perform the required operations.</td>
</tr>
</tbody>
</table>

11 Click **Apply changes**.

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

12 Repeat **Step 3** through **Step 11** for each vCenter Server instance.

### vCenter Server 5.0.1 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.


### Using the vCenter Server 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 the **Use contextual menu in inventory** option is enabled, all of the workflows that you can run on the selected inventory object appear in a contextual menu.

### Access the vCenter Server Plug-In Workflow Library

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

#### Prerequisites

- The vCenter Server plug-in must be enabled and configured in the Orchestrator configuration interface.
- You must be logged in to the Orchestrator client as a user who can run vCenter workflows.

#### Procedure

1 Click the **Workflows** view in the Orchestrator client.
2 Expand the hierarchical list to Library > vCenter.

What to do next
Review the workflow library.

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

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

You access the batch workflows from Library > vCenter > Batch.

<table>
<thead>
<tr>
<th>Fill batch configuration elements</th>
<th>Populates the configuration elements that the Run a workflow on a selection of objects workflow uses. Performs the following tasks:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Resets the BatchObject and BatchAction configuration elements.</td>
</tr>
<tr>
<td></td>
<td>Fills the BatchObject configuration element with all of the workflows that have only one input parameter.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

| 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 access the cluster and compute resource workflows from Library > vCenter > Cluster and Compute Resource.

| 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.

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 access the custom attributes workflows from Library > vCenter > Custom Attributes.

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 access the datacenter workflows from Library > vCenter > Datacenter.

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 access the datastore and files workflows from Library > vCenter > Datastore and Files.

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 access the datacenter folder management workflows from Library > vCenter > Folder management > Datacenter folder.

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 access the host folder management workflows from Library > vCenter > Folder management > Host folder.

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 access the virtual machine folder management workflows from Library > vCenter > Folder management > VM folder.

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 access the basic host management workflows from Library > vCenter > Host management > Basic.

- **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 into 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 access the power host management workflows from Library > vCenter > Host management > Power.

- **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 Management Registration Workflows

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

You access the host management registration workflows from Library > vCenter > Host management > Registration.

- **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 access the networking workflows from Library > vCenter > Networking.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add port group to distributed virtual switch</td>
<td>Adds a new distributed virtual port group to a specified distributed virtual switch.</td>
</tr>
<tr>
<td>Attach host system to distributed virtual switch</td>
<td>Adds a host to a distributed virtual switch.</td>
</tr>
<tr>
<td>Create distributed virtual switch with port group</td>
<td>Creates a new distributed virtual switch with a distributed virtual port group.</td>
</tr>
</tbody>
</table>

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 access the distributed virtual port group workflows from Library > vCenter > Networking > Distributed virtual port group.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect virtual machine NIC number to distributed virtual port group</td>
<td>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.</td>
</tr>
<tr>
<td>Delete distributed virtual port group</td>
<td>Deletes a specified distributed virtual port group.</td>
</tr>
<tr>
<td>Set teaming options</td>
<td>Provides an interface to manage the teaming options for a distributed virtual port group.</td>
</tr>
<tr>
<td>Update distributed virtual port group</td>
<td>Updates the configuration of a specified distributed virtual port group.</td>
</tr>
</tbody>
</table>

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 access the distributed virtual switch workflows from Library > vCenter > Networking > Distributed virtual switch.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create distributed virtual switch</td>
<td>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.</td>
</tr>
<tr>
<td>Create private VLAN</td>
<td>Creates a VLAN on the specified distributed virtual switch.</td>
</tr>
<tr>
<td>Delete distributed virtual switch</td>
<td>Deletes a distributed virtual switch and all port groups, uplink ports and so on, that are associated with the switch.</td>
</tr>
<tr>
<td>Delete private VLAN</td>
<td>Deletes a VLAN from a specified distributed virtual switch. If a secondary VLAN exists, you should first delete the secondary VLAN.</td>
</tr>
</tbody>
</table>
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 access the standard virtual switch workflows from **Library > vCenter > Networking > Standard virtual switch**.

- **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 to 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 access the resource pool workflows from **Library > vCenter > Resource Pool**.

- **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 access the storage workflows from Library > vCenter > Storage.

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 send and static targets to a vCenter Server host.

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 an already configured iSCSI send and static target.

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 access the storage DRS workflows from Library > vCenter > Storage > Storage DRS.

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.
<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create VMDK anti-affinity rule</td>
<td>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.</td>
</tr>
<tr>
<td>Remove datastore cluster</td>
<td>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.</td>
</tr>
<tr>
<td>Remove datastore from cluster</td>
<td>Removes a datastore from a datastore cluster and puts the datastore in a datastore folder.</td>
</tr>
<tr>
<td>Remove Storage DRS scheduled task</td>
<td>Removes a scheduled Storage DRS task.</td>
</tr>
<tr>
<td>Remove virtual machine anti-affinity rule</td>
<td>Removes a virtual machine anti-affinity rule for a given datastore cluster.</td>
</tr>
<tr>
<td>Remove VMDK anti-affinity rule</td>
<td>Removes a VMDK anti-affinity rule for a given datastore cluster.</td>
</tr>
</tbody>
</table>

### 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 access the basic virtual machine management workflows from **Library > vCenter > Virtual Machine management > Basic**.

<table>
<thead>
<tr>
<th>Action</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>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>
</tbody>
</table>
**Rename virtual machine**
Renames an existing virtual machine on the vCenter Server system or host and not on the datastore.

**Set virtual machine performance**
Changes performance settings such as shares, minimum and maximum values, shaping for network, and disk access of a virtual machine.

**Unregister virtual machine**
Removes an existing virtual machine from the inventory.

**Upgrade VM Hardware (force if required)**
Upgrades the virtual machine hardware to the latest revision that the host supports. This workflow forces the upgrade to continue, even if VMware Tools is out of date. If the VMware Tools is out of date, forcing the upgrade to continue reverts the guest network settings to the default settings. To avoid this situation, upgrade VMware Tools before running the workflow.

**Upgrade virtual machine**
Upgrades the virtual hardware to the latest revision that the host supports. An input parameter allows a forced upgrade even if VMware Tools is out of date.

**Wait for task and answer virtual machine question**
Waits for a vCenter Server task to complete or for the virtual machine to ask a question. If the virtual machine requires an answer, accepts user input and answers the question.

**Clone Workflows**
With clone workflows you can clone virtual machines with or without customizing the virtual machine properties.

You access the clone workflows from **Library > vCenter > Virtual Machine management > Clone**.

- **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 access the linked clone workflows from **Library > vCenter > Virtual Machine management > Clone > Linked Clone**.

- **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.

- **Linked clone, Linux 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.

- **Linked clone, Linux with single NIC**
  Creates a linked clone of a Linux virtual machine, performs the guest operating system customization, and configures one virtual network card.
Linked clone, Windows 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.

Linked clone, Windows with 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.

Linked clone, no customization

Creates the specified number of linked clones of a given virtual machine.

Linux Customization Clone Workflows

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

You access the Linux customization clone workflows from Library > vCenter > Virtual Machine management > Clone > Linux Customization.

Clone, Linux with multiple NICs

Clones a Linux virtual machine, performs the guest operating system customization, and configures up to four virtual network cards.

Clone, Linux with 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 access the tools clone workflows from Library > vCenter > Virtual Machine management > Clone > Tools.

Get Linux customization

Returns the Linux customization preparation.

Get NIC setting map

Returns the setting map for a virtual network card by using VimAdapterMapping.

Get Windows customization, 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, 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

Returns customization information about the Microsoft Sysprep process. Workflows for cloning Windows virtual machines use this workflow.

Get a VirtualEthernetCard 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 multiple VirtualEthernetCard device changes

Returns an array of VirtualDeviceConfigSpec objects for add and remove operations on VirtualEthernetCard objects.
Windows Customization Clone Workflows

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

You access the Windows customization clone workflows from **Library > vCenter > Virtual Machine management > Clone > Windows Customization**.

**Customize, Windows 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 thin provisioned, Windows 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, Windows 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 the vCenter Server.

**Clone, Windows 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, Windows 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, Windows 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 access the device management workflows from **Library > vCenter > Virtual Machine management > Device Management**.

**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

Mount floppy disk drive

Disconnects floppy disks, CD-ROM drives, parallel ports, and serial ports from a running virtual machine.

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 access the move and migrate workflows from Library > vCenter > Virtual Machine management > Move and Migrate.

Mass migrate virtual machines with storage vMotion

Mass migrate virtual machines with vMotion

Migrate virtual machine with vMotion

Move virtual machine to another vCenter Server

Quick migrate multiple virtual machines

Quick migration of virtual machine

Relocate virtual machine disks

Uses Storage vMotion to migrate a single virtual machine, a selection of virtual machines, or all available virtual machines.

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.

Migrates a virtual machine from one host to another by using the `migrateVM_Task` operation from the vSphere API.

Moves a list of virtual machines to another vCenter Server system.

Suspends the virtual machines if they are powered on and migrates them to another host using the same storage.

Suspends the virtual machine if it is powered on and migrates it to another host using the same storage.

Relocates virtual machine disks to another host or datastore while the virtual machine is powered off by using the `RelocateVM_Task` operation from the vSphere API.

Other Workflows

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

You access these workflows from Library > vCenter > Virtual Machine management > Others.

Disable FT

Enable FT

Extract virtual machine information

Find orphaned virtual machines

Disables Fault Tolerance for a specified virtual machine.

Enables Fault Tolerance for a specified virtual machine.

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.

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).
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 access the power management workflows from Library > vCenter > Virtual Machine management > Power Management.

- **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** 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 access the snapshot workflows from Library > vCenter > Virtual Machine management > Snapshot.

- **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.
VMware Tools Workflows

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

You access the VMware Tools workflows from Library > vCenter > Virtual Machine management > VMware Tools.

- **Mount 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 tools installer**: Unmounts the VMware Tools CD-ROM.
- **Upgrade tools**: Upgrades VMware Tools on a virtual machine.
- **Upgrade 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 the 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 > Task 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 > Task in the Workflows view of the Orchestrator client.
- **Start workflows in a series**: Runs a workflow multiple times in a series, one instance after the other. You provide workflow parameters in an array, providing a property list, with one property per workflow input, for each instance of the workflow that starts. The number of properties in the array define the number of workflow runs.
- **Start workflows in parallel**: Runs a workflow multiple times, with different parameters. You provide workflow parameters in an array, providing 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, server, configuration, wrapper, and installation log files.
The output directory must exist and write access must be permitted.
Using the Database Plug-In

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

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

This chapter includes the following topics:

- “Database Plug-In Scripting API,” on page 33
- “Running the JDBC Sample Workflows,” on page 35

Database Plug-In Scripting API

The Database 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.

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.
### 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 <code>PreparedStatement</code> object.</td>
</tr>
<tr>
<td>executeQuery():ResultSet</td>
<td>ResultSet</td>
<td>Runs an SQL query in the <code>PreparedStatement</code> object and returns the <code>ResultSet</code> 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 <code>PreparedStatement</code> 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 <code>ResultSet</code> 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 <code>ResultSet</code> 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 <code>ResultSet</code> object. Returns <code>true</code> if the cursor is on a valid row, and <code>false</code> if there are no rows in the result set.</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>Method</td>
<td>Returns</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>getNumberAt(number):number</code></td>
<td>Number</td>
<td>Returns the Number value of a given column index.</td>
</tr>
<tr>
<td><code>getString(string):string</code></td>
<td>String</td>
<td>Returns the String value of a given column name.</td>
</tr>
<tr>
<td><code>getStringAt(number):string</code></td>
<td>String</td>
<td>Returns the String value of a given column index.</td>
</tr>
<tr>
<td><code>getTimestamp(string):Object</code></td>
<td>Object</td>
<td>Returns the Timestamp value of a given column name.</td>
</tr>
<tr>
<td><code>getTimestampAt(number):Object</code></td>
<td>Object</td>
<td>Returns the Timestamp value of a given column index.</td>
</tr>
<tr>
<td><code>last():boolean</code></td>
<td>Boolean</td>
<td>Moves the cursor to the last 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><code>next():boolean</code></td>
<td>Boolean</td>
<td>Moves the cursor down one row. Returns true if the cursor is on a valid row, and false if there are no more rows.</td>
</tr>
<tr>
<td><code>previous():boolean</code></td>
<td>Boolean</td>
<td>Moves the cursor to the previous row in the ResultSet object. Returns true if the cursor is on a valid row, and false if it is off the result set.</td>
</tr>
<tr>
<td><code>wasNull():boolean</code></td>
<td>Boolean</td>
<td>Reports whether the last column read had a value of SQL NULL.</td>
</tr>
</tbody>
</table>

**Running the JDBC Sample Workflows**

You can run the Database plug-in sample workflows from the Orchestrator client to test SQL-based database operations.

**Generate a JDBC URL**

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

**Prerequisites**

- The Database plug-in must be enabled in the Orchestrator configuration interface.
- You must be logged 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 need to provide the database instance name and database user domain name.

5. Provide the required information to generate a database URL.
   - Type a database server name or IP address.
   - 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**
- The Database plug-in must be enabled in the Orchestrator configuration interface.
- You must be logged 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 test the creation of a database.

**Prerequisites**
- The Database plug-in must be enabled in the Orchestrator configuration interface.
- You must be logged 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 to create a table.
   a Type a password to access the database.
   b Type a JDBC connection URL.
c Type a user name to access the database.

d Type an SQL create statement.

An example syntax is:

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

5 Click Submit to run the workflow.

**Insert a Row into a JBCD Table**

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

**Prerequisites**

- The Database plug-in must be enabled in the Orchestrator configuration interface.
- You must be logged 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 to insert a row into a table.

   a Type a JDBC connection URL.
   
   b Type a user name to access the database.
   
   c Type a password to access the database.
   
   d Type an SQL insert statement.

   An example syntax is:

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

   e Type the values to insert into the row.

5 Click Submit to run the workflow.

**Select Rows from a JDBC Table**

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

**Prerequisites**

- The Database plug-in must be enabled in the Orchestrator configuration interface.
- You must be logged 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 to select rows from a table.
   a. Type a JDBC connection URL.
   b. Type a user name to access the database.
   c. Type a password to access the database.
   d. Type an SQL select statement.
      An example syntax is:
      
      ```
      SELECT * FROM "table_name"
      ```

5. 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**

- The Database plug-in must be enabled in the Orchestrator configuration interface.
- You must be logged 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 to delete an entry from a table.
   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.
   f. Type an SQL delete statement.
      An example syntax is:
      
      ```
      DELETE FROM "table_name" where ("column1" = ?, "column2" = ?)
      ```

5. Click Submit to run the workflow.

Delete All Entries from a JDBC Table

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

**Prerequisites**

- The Database plug-in must be enabled in the Orchestrator configuration interface.
- You must be logged in to the Orchestrator client as a user who can run JDBC workflows.

**Procedure**

1. Click the Workflows view in the Orchestrator client.
In the workflows hierarchical list, open **Library > JDBC > JDBC Examples** to navigate to the JDBC delete all from table example workflow.

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

Provide the required information to delete all all entries from a table.

- Type a JDBC connection URL.
- Type a user name to access the database.
- Type a password to access the database.
- Type an SQL delete statement.

An example syntax is:

```
DELETE FROM "table_name"
```

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**

- The Database plug-in must be enabled in the Orchestrator configuration interface.
- You must be logged 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 to drop a table from the database.
   - Type a password to access the database.
   - Type a JDBC connection URL.
   - Type a user name to access the database.
   - Type an SQL drop statement.

An example syntax is:

```
DROP TABLE "table_name"
```

5. 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**

- The Database plug-in must be enabled in the Orchestrator configuration interface.
- You must be logged 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 to run a complete database cycle.
   a Type a JDBC connection URL.
   b Type a user name to access the database.
   c Type a password to access the database.
   d Type the values to be used as entries in the database.

5 Click Submit to run the workflow.
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 41
- “SSH Plug-In Scripting API,” on page 42
- “Running the SSH Plug-In Sample Workflows,” on page 46

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, enter the host to access with SSH through Orchestrator.
   
   **NOTE** The username and password are not 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.
### 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. 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>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>
</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>path</td>
<td>String</td>
<td>Full folder path</td>
</tr>
<tr>
<td>port</td>
<td>Number</td>
<td>SSH port</td>
</tr>
</tbody>
</table>

<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. 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. 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,SecureString):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>
</tbody>
</table>
### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>executeAndLog(string):Object</code></td>
<td>Object</td>
<td>Runs a single command and waits until it is completed.</td>
</tr>
<tr>
<td><code>executeCommand(string,boolean):string</code></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><code>findAll(Path,string):string[]</code></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><code>findDir(Path,string):string[]</code></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><code>findFirst(Path,string):string[]</code></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><code>getFile(Path,Path):number</code></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><code>listAll(Path):string[]</code></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><code>listDir(Path):string[]</code></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><code>listFile(Path):string[]</code></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><code>putFile(Path,Path):number</code></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

- The SSH plug-in must be enabled and configured in the Orchestrator configuration interface.
- You must be logged 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

- The SSH plug-in must be enabled and configured in the Orchestrator configuration interface.
- You must be logged 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

- The SSH plug-in must be enabled and configured in the Orchestrator configuration interface.
- You must be logged 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**

- The SSH plug-in must be enabled and configured in the Orchestrator configuration interface.
- You must be logged 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. Provide the required information.
   
   a. Type an SSH host name or IP.
   b. Type an SSH command to run.

   **NOTE** The default SSH command is `uptime`. It shows how long the server has been active and the user load for that period.
   
   c. (Optional) Select **Yes** to use password authentication.

   **NOTE** The default option is to use key file authentication.
   
   d. Type the authentication information.
5. Click **Submit** to run the workflow.

Copy a File from an SSH Host

You can run a workflow from 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**

- The SSH plug-in must be enabled and configured in the Orchestrator configuration interface.
- You must be logged 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.
   a. Type an SSH host name or IP address.
   b. Type the SSH authentication information.
   c. Type the path to the directory on the Orchestrator server into which to copy the file.
   d. Type the path to the file to get from the remote SSH host.
5. 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**

- The SSH plug-in must be enabled and configured in the Orchestrator configuration interface.
- You must be logged 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.
   a. Type an SSH host name or IP address.
   b. Type the SSH authentication information.
   c. Type the path to the file that you want to copy from the local Orchestrator server to the remote SSH host.
   d. Type the path to the directory on the remote SSH host into which to copy the file.
5. 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 51
- “Running the XML Plug-In Sample Workflows,” on page 55

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 new child node to the end of the list of child nodes of this node. If the new child 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>
</tbody>
</table>
### Method | Returns | Description
--- | --- | ---
createTextNode(String):XMLNode | XMLNode | Creates a text node.
getChildNodes():Object | Object | Gets all child nodes.
getDocumentElement():XMLElement | XMLElement | Allows direct access to the child node that is at the root element of the document.
getElementsByTagName(String):Object | Object | 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.
insertBefore(XMLNode,XMLNode):Object | Object | 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.
normalize():Object | Object | Normalizes the document.
removeChild(XMLNode):Object | Object | Removes the child node indicated by oldChild from the list of child nodes and returns it.
replaceChild(XMLNode,XMLNode):Object | Object | 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.

### XMLElement Class

The XMLElement class is the main element class.

The XMLElement class defines the following attributes.

| Attribute | Returns | Description |
--- | --- | ---
tagName | String | The tag name
textContent | String | The text content

The XMLElement class defines the following methods.

| Method | Returns | Description |
--- | --- | ---
appendChild(XMLNode):Object | Object | 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.
cloneNode(boolean):Object | Object | Clones a node.
getAttributes():Object | Object | Contains the attributes of an element.
getChildNodes():XMLNodeList | XMLNodeList | Gets all child nodes.
getElementsByTagName(String):Object | Object | 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.
### Method Returns Description

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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.
### 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>nodeName</td>
<td>String</td>
<td>The node name</td>
</tr>
<tr>
<td>nodeValue</td>
<td>String</td>
<td>The node value</td>
</tr>
<tr>
<td>parentNode</td>
<td>Object</td>
<td>The parent node (XMLNode)</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>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>getChildNodes():XMLNodeList</td>
<td>XMLNodeList</td>
<td>Gets all child nodes.</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>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>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>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>
**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>item(number):Object</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 *Administering 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**

- The XML plug-in must be enabled from the Orchestrator configuration interface.
- You must be logged 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

- The XML plug-in must be enabled from the Orchestrator configuration interface.
- You must be logged 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

- The XML plug-in must be enabled from the Orchestrator configuration interface.
- You must be logged 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
   - The XML plug-in must be enabled from the Orchestrator configuration interface.
   - You must be logged 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 59
- “Mail Plug-In Scripting API,” on page 60
- “Email Scripting Examples,” on page 60
- “Using the Mail Plug-In Sample Workflows,” on page 61

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.</td>
</tr>
<tr>
<td></td>
<td>The default SMTP port is 25.</td>
</tr>
<tr>
<td>User name</td>
<td>Enter a valid email account.</td>
</tr>
<tr>
<td></td>
<td>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**

- The Mail plug-in must be enabled and configured in the Orchestrator configuration interface.
- You must be logged 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.

- **Retrieve messages**
  Retrieves the messages of a given email account by using the POP3 protocol.

- **Send interaction**
  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.

- **Send notification**
  Sends an email with specified content to a given email address. If optional parameters are not specified, the workflow uses the default values set in the Orchestrator configuration interface.

- **Send notification to mailing list**
  Sends an email with specified content to a given email address list, CC list, and BCC list. If optional parameters are not specified, the workflow uses the default values set in the Orchestrator configuration interface.
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 59.

Prerequisites

- The Mail plug-in must be enabled and configured in the Orchestrator configuration interface.
- You must be logged 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>
</tbody>
</table>
## Method Returns Description

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getFile(String, String) : Number</td>
<td>Number</td>
<td>Copies a file from a remote host to the Orchestrator server.</td>
</tr>
<tr>
<td>getStatus() : String</td>
<td>String</td>
<td>Provides the status information returned by the server.</td>
</tr>
<tr>
<td>getString(String) : String</td>
<td>String</td>
<td>Gets the content of a remote file.</td>
</tr>
<tr>
<td>isAvailable() : boolean</td>
<td>Boolean</td>
<td>Makes checks on the socket to test whether it is available for use.</td>
</tr>
<tr>
<td>isConnected() : boolean</td>
<td>Boolean</td>
<td>Provides information if the client is currently connected to a server.</td>
</tr>
<tr>
<td>listAll(String) : String[]</td>
<td>Array of String</td>
<td>Lists all files and directories in a path.</td>
</tr>
<tr>
<td>listDir(String) : String[]</td>
<td>Array of String</td>
<td>Lists directories in a path.</td>
</tr>
<tr>
<td>listFile(String) : String[]</td>
<td>Array of String</td>
<td>Lists files in a path.</td>
</tr>
<tr>
<td>login(String, String, String) : boolean</td>
<td>Boolean</td>
<td>Logs in on an FTP server.</td>
</tr>
<tr>
<td>putFile(String, String) : Number</td>
<td>Number</td>
<td>Copies a file from the Orchestrator server to a remote host.</td>
</tr>
<tr>
<td>putString(String, String) : Number</td>
<td>Number</td>
<td>Puts 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): 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 from the server.</td>
</tr>
<tr>
<td>enableSSL(boolean):Object</td>
<td>Object</td>
<td>Enables SSL.</td>
</tr>
<tr>
<td>isAvailable():Boolean</td>
<td>Boolean</td>
<td>Makes checks on the socket to test if it is available for use.</td>
</tr>
<tr>
<td>isConnected():Boolean</td>
<td>Boolean</td>
<td>Provides information if the client is currently connected to a server.</td>
</tr>
</tbody>
</table>
### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>listMessages(): POP3Message[]</code></td>
<td>Array of POP3Message</td>
<td>Lists all messages. If there are no messages, the method returns a zero length array. If the operation fails, the method returns null.</td>
</tr>
<tr>
<td><code>login(String,String): boolean</code></td>
<td>Boolean</td>
<td>Logs in on POP3 server with the given user name and password. You must first connect to the server with the <code>connect()</code> method, before attempting to log in.</td>
</tr>
<tr>
<td><code>loginWithSecret(String,String,String): boolean</code></td>
<td>Boolean</td>
<td>Logs in on POP3 server with the given user name and authentication information.</td>
</tr>
<tr>
<td><code>logout(): Object</code></td>
<td>Object</td>
<td>Logs out of the POP3 server. To fully disconnect from the server, you must call the <code>disconnect()</code> method.</td>
</tr>
<tr>
<td><code>noop(): Object</code></td>
<td>Object</td>
<td>Sends a NOOP command to the POP3 server. This is useful for keeping the connection alive.</td>
</tr>
<tr>
<td><code>reset(): Object</code></td>
<td>Object</td>
<td>Resets the POP3 session. This is useful for undoing any message deletions that might have been performed.</td>
</tr>
</tbody>
</table>

### POP3Message Class

The `POP3Message` class provides POP3 email functionality.

The `POP3Message` class defines the following attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>String</td>
<td>Message body</td>
</tr>
<tr>
<td>from</td>
<td>String</td>
<td>Sender</td>
</tr>
<tr>
<td>id</td>
<td>Number</td>
<td>Message ID in current session</td>
</tr>
<tr>
<td>subject</td>
<td>String</td>
<td>Message subject</td>
</tr>
<tr>
<td>to</td>
<td>String</td>
<td>Recipient</td>
</tr>
</tbody>
</table>

The `POP3Message` class defines the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>deleteFromServer(): Object</code></td>
<td>Object</td>
<td>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 <code>reset()</code> command from the <code>POP3Client</code> class. The server deletes all messages marked for deletion when you call the <code>logout()</code> command from the <code>POP3Client</code> class.</td>
</tr>
<tr>
<td><code>getHeader(String): String</code></td>
<td>String</td>
<td>Gets the header value.</td>
</tr>
</tbody>
</table>
**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>
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 67
- “Time Zone Codes,” on page 68

### Enumeration Plug-In Scripting API

The Enums scripting API contains 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>All: Configuration file and all disks</td>
<td>Kind of backup</td>
</tr>
<tr>
<td></td>
<td>Config: Configuration file only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>None: No backup at all</td>
<td></td>
</tr>
<tr>
<td></td>
<td>System: Configuration file and first disk</td>
<td></td>
</tr>
<tr>
<td>Enums:Disk</td>
<td>0: No disk</td>
<td>Possible disk sizes</td>
</tr>
<tr>
<td></td>
<td>2: 2GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4: 4GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8: 8GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12: 12GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16: 16GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20: 20GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60: 60GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80: 80GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100: 100GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>120: 120GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>160: 160GB</td>
<td></td>
</tr>
<tr>
<td>Enums:Environment</td>
<td>Validation: Validation environments</td>
<td>Type of deployment environment</td>
</tr>
<tr>
<td></td>
<td>Test: Test environments</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Development: Development environments</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Production: Production environments</td>
<td></td>
</tr>
<tr>
<td>Enums:MSTimeZone</td>
<td>See “Time Zone Codes,” on page 68</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><strong>Low</strong>: Low (20%)&lt;br&gt;<strong>Medium</strong>: Medium (60%)&lt;br&gt;<strong>High</strong>: High (80%)</td>
<td>Type of performance</td>
</tr>
<tr>
<td>Enums:RAM</td>
<td>128: 128MB&lt;br&gt;256: 256MB&lt;br&gt;512: 512MB&lt;br&gt;1024: 1GB&lt;br&gt;2048: 2GB&lt;br&gt;3584: 3.5GB (ESX 2 maximum)&lt;br&gt;4096: 4GB&lt;br&gt;6144: 6GB&lt;br&gt;8192: 8GB&lt;br&gt;10240: 10GB&lt;br&gt;12288: 12GB&lt;br&gt;14336: 14GB&lt;br&gt;16384: 16GB</td>
<td>Possible RAM sizes</td>
</tr>
<tr>
<td>Enums:vCPU</td>
<td>1: Virtual machine with 1 virtual CPU&lt;br&gt;2: Virtual machine with 2 virtual CPUs&lt;br&gt;3: Virtual machine with 3 virtual CPUs&lt;br&gt;4: Virtual machine with 4 virtual CPUs&lt;br&gt;5: Virtual machine with 5 virtual CPUs&lt;br&gt;6: Virtual machine with 6 virtual CPUs&lt;br&gt;7: Virtual machine with 7 virtual CPUs&lt;br&gt;8: Virtual machine with 8 virtual CPUs</td>
<td>Number of virtual CPUs</td>
</tr>
</tbody>
</table>

### Time Zone Codes

You can use the time zone codes as possible values for the Enums:MSTimeZone 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, Monaco</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) Colombo, Kandy, Peradeniya</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) Ulaanbataar, Irkutsk</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>
Index

A
API Explorer, accessing 12
audience 7

B
basic host management workflows 20
basic virtual machine workflows 24
batch workflows 17

C
close workflows 25
cluster and compute resource workflows 17
custom attributes workflows 18

D
Database plug-in
scripting API 33
usage 33
Database plug-in API
connecting to a database 33
Connection class 33
database connection parameters 33
JDBCConnection class 33
precompiled SQL statement 34
PreparedStatement class 34
ResultSet class 34
Database plug-in sample workflows 35
datacenter folder management folder 19
datacenter workflows 18
datastore and files workflows 18
default plug-ins 10
device management workflows 27
distributed virtual port group workflows 21
distributed virtual switch workflows 21

E
Enumeration plug-in
 Enums: MSTimeZone values 68
scripting API 67
usage 67

H
host folder management workflows 19
host management registration workflows 20

I
introduction 9

J
JDBC connection, testing 36
JDBC dropping, testing 39
JDBC entry deletion, testing 38
JDBC full cycle, testing 39
JDBC plug-in, usage 33
JDBC row insertion, testing 37
JDBC row selection, testing 37
JDBC sample workflows 35
JDBC table creation, testing 36
JDBC URL generator 35

L
linked clone workflows 25
Linux customization clone workflows 26
load balancing 59

M
Mail plug-in
sample workflows access 61
scripting API 60
usage 59
Mail plug-in API, EmailMessage class 60
Mail plug-in sample workflows 61
Mail plug-in workflows, example interaction with email 62
move and migrate workflows 28

N
Net plug-in
scripting API 63
usage 63
Net plug-in API
FTPCli classent class 63
POP3Client class 64
POP3Message class 65
TelnetClient class 66
networking workflows 21

O
Orchestrator architecture 9
overview of
Database plug-in 10
Enumeration plug-in 10
| Mail plug-in | 10 |
| Net plug-in | 10 |
| overview of | 10 |
| SSH plug-in | 10 |
| vCenter Server 4.1 plug-in | 10 |
| vCO Library plug-in | 10 |
| vCO Weboperator plug-in | 10 |
| XML plug-in | 10 |

**P**

plug-ins configuration

| Mail plug-in | 59 |
| SSH plug-in | 41 |
| vCenter Server plug-in | 15 |

power host management workflows | 20 |

power management workflows | 29 |

**R**

resource pool workflows | 22 |

**S**

scripting, email examples | 60 |

SFTP | 48, 49 |

SMTP connection | 59 |

snapshot workflows | 29 |

SSH commands, running | 48 |

SSH file copy

SCP get command | 48 |

SCP put command | 49 |

SSH plug-in scripting API | 42 |

usage | 41 |

SSH plug-in API

KeyPairManager class | 42 |

managing private and public SSH keys | 42 |

SSH:File type | 42 |

SSH:Folder type | 42 |

SSH:RootFolder type | 42 |

SSH:SshConnection type | 42 |

SSHCommand class | 43 |

SSHFile class | 44 |

SSHFolder class | 44 |

SSHSession class | 45 |

SSH plug-in sample workflows | 46 |

SSH workflows

changing a key pair passphrase | 47 |

generating a key pair | 46 |

registering a public key | 47 |

standard virtual switch workflows | 22 |

storage DRS workflows | 23 |

storage workflows | 23 |

**T**

tools clone workflows | 26 |

**V**

vCenter Server 5.0.1 plug-in

scripting API | 16 |

usage | 15 |

vCenter Server plug-in

Inventory | 16 |

workflow library | 17 |

workflow library access | 16 |

vCO Library plug-in

usage | 31 |

workflows | 31 |

virtual machine folder management workflows | 19 |

VMware Tools workflows | 30 |

**W**

windows customization clone workflows | 27 |

workflows

basic host management | 20 |

basic virtual machine | 24 |

batch | 17 |

clone | 25 |

cluster and compute resource | 17 |

custom attributes | 18 |

datacenter folder management | 19 |

datancenter | 18 |

datastore and files | 18 |

device management | 27 |

distributed virtual port group | 21 |

distributed virtual switch | 21 |

host folder management | 19 |

host management registration | 20 |

linked clone | 25 |

Linux customization clone | 26 |

move and migrate | 28 |

networking | 21 |

other | 28 |

power host management | 20 |

power management | 29 |

resource pool | 22 |

snapshot | 29 |

standard virtual switch | 22 |

storage | 23 |

Storage DRS | 23 |

tools clone | 26 |

virtual machine folder management | 19 |

VMware Tools | 30 |

windows customization clone | 27 |
XML
  DOM 51
  E4X 51
XML plug-in
  scripting API 51
  usage 51
XML plug-in API
  XMLDocument class 51
  XMLDocument class 52
  XMLManager class 53
  XMLNamedNodeMap class 53
  XMLNode class 54
  XMLNodeList class 55
XML plug-in sample workflows
  creating a simple XML document 55
  creating an address book 57
  finding an element 56
  modifying an XML document 56
