# Using the vCenter Orchestrator SOAP Plug-In 1.0.1

vCenter Orchestrator 4.1

vCenter Orchestrator 4.2

EN-000623-01

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

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

# Contents

# Using the vCenter Orchestrator SOAP Plug-In

*Using the vCenter Orchestrator SOAP Plug-In* provides information and instructions about configuring and using the VMware® vCenter Orchestrator SOAP plug-in.

## Intended Audience

This information is intended for anyone who is installing and configuring the plug-in, using the API of the plug-in, and using the workflow library. *Using the vCenter Orchestrator SOAP Plug-In* is written for experienced users who are familiar with virtual machine technology, with Orchestrator workflow development, and with SOAP.

For more information about Orchestrator, see
http://www.vmware.com/support/pubs/orchestrator_pubs.html.

# Updated Information

*Using the vCenter Orchestrator SOAP Plug-In* is updated with each release of the product or when necessary.

This table provides the update history of *Using the vCenter Orchestrator SOAP Plug-In*.

| Revision | Description |
|---|---|
| EN-000623-01 | ■ Added information about the new workflow Update a SOAP host with an endpoint URL in "Configuration Workflows," on page 11.<br>■ Updated the procedure with a new authentication type in "Add a SOAP Host," on page 12.<br>■ Added information about the new API methods and attributes in "SOAPHost Class," on page 18 and "SOAPOperation Class," on page 20.<br>■ Added a new API class "SOAPInterceptor Class," on page 19.<br>■ Updated "Exporting and Importing SOAP Configuration," on page 14 with information about exporting and importing SSL certificates. |
| EN-000623-00 | Initial release. |

# Introduction to the VMware vCenter Orchestrator SOAP Plug-In 1

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

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

This chapter includes the following topics:

## Role of vCenter Orchestrator with the SOAP Plug-In

You must use the Orchestrator configuration interface to install the SOAP plug-in. You use the Orchestrator client to run and create workflows and access the plug-in API.

The SOAP plug-in is powered by vCenter Orchestrator. Orchestrator is a development and process-automation platform that provides a library of extensible workflows to manage the VMware vCenter infrastructure and other technologies.

Orchestrator allows integration with management and administration solutions through its open plug-in architecture. SOAP is one example of a protocol that you can integrate with Orchestrator by using plug-ins.

## Installing the SOAP Plug-In

You must use the Orchestrator configuration interface to install the SOAP plug-in.

### SOAP Plug-In Functional Prerequisites

To be able to install and use the SOAP plug-in, your system must meet the following functional prerequisites.

#### vCenter Orchestrator

Verify that you have a running instance of Orchestrator. You can log in to the Orchestrator configuration interface at http://*orchestrator_server*:8282. Version 1.0.1 of the plug-in works with vCenter Orchestrator 4.1 and 4.2.

For information about setting up Orchestrator, see the *vCenter Orchestrator Installation and Configuration Guide*.

### SOAP

Verify that you have access to a SOAP host. The plug-in supports SOAP Version 1.1 and 1.2, and WSDL 1.1 and 2.0.

## Install the SOAP Plug-In

To be able to use the SOAP plug-in, you must download the `.vmoapp` file containing the plug-in and install it using the Orchestrator configuration interface.

#### Prerequisites

- Verify that you are logged in to the Orchestrator configuration interface at http://*orchestrator_server*:8282.
- Verify that you have downloaded the `.vmoapp` file from http://www.vmware.com/products/datacenter-virtualization/vcenter-orchestrator/plugins.html.

#### Procedure

1. On the **General** tab, click **Install Application**.
2. Upload the SOAP plug-in.
   a. Click the magnifying glass icon.
   b. Select the `.vmoapp` file to install.
   c. Click **Open**.
   d. Click **Install**.

   The SOAP plug-in tab appears in the Orchestrator configuration interface.
3. On the **Startup Options** tab, click **Restart service** to complete the plug-in installation.

# Using the SOAP Plug-In 2

The SOAP plug-in workflow library contains workflows that allow you to manage SOAP hosts and run custom SOAP operations.

This chapter includes the following topics:

## Using the SOAP Plug-In Inventory

The SOAP plug-in exposes all objects in the connected SOAP hosts in the **Inventory** view. You can use the **Inventory** view to add authorization elements or to run workflows on SOAP objects.

You can enable the **Use contextual menu in inventory** option to display the workflows that are available for an inventory object. When the option is enabled and you right-click an object in the Orchestrator inventory, all available workflows for the object are displayed.

## Configuring the SOAP Plug-In

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

### Configuration Workflows

The Configuration workflow category contains workflows that allow you to manage SOAP hosts.

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

| Workflow Name | Description |
|---|---|
| Add a SOAP host | Adds a SOAP host to the plug-in's inventory. |
| Manage SSL certificates | Verifies a host URL, and if required, shows a user interaction message for the approval of SSL certificates. |
| Reload plug-in configuration | Refreshes the list of SOAP hosts in the plug-in's inventory. |
| Remove a SOAP host | Removes a SOAP host from the plug-in's inventory.<br>CAUTION When you remove a host from the inventory, all workflows generated from it will stop working. |

| Workflow Name | Description |
|---|---|
| Update a SOAP host | Updates a SOAP host in the plug-in's inventory. |
| Update a SOAP host with an endpoint URL | Updates a SOAP host with a preferred endpoint address. The new endpoint address is used for sending and receiving SOAP messages, instead of the endpoint address defined within the WSDL. |

## Add a SOAP Host

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

### Procedure

1. Log in to the Orchestrator client as an administrator.
2. Click the **Workflows** view in the Orchestrator client.
3. In the workflows hierarchical list, select **Library > SOAP > Configuration** and navigate to the **Add a SOAP** host workflow.
4. Right-click the **Add a SOAP** host workflow and select **Start workflow**.
5. In the **Name** text box, type the name of the host.
6. Select whether to provide the WSDL content as text.

| Option | Action |
|---|---|
| **Yes** | Copy the text in the **WSDL content** text box. |
| **No** | Type the correct path in the **WSDL URI** text box. |

7. In the **Connection timeout** text box, specify the number of seconds before a connection times out.
8. In the **Request timeout** text box, specify the number of seconds before a request times out.
9. Select the authentication type.

| Option | Description |
|---|---|
| **None** | No authentication is required. |
| **Basic** | Provides basic access authentication.<br>Select the session mode.<br>■ If you select **Shared Session**, provide credentials for the shared session.<br>■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in. |
| **Digest** | Provides digest access authentication that uses encryption.<br>Select the session mode.<br>■ If you select **Shared Session**, provide credentials for the shared session.<br>■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in. |
| **NTLM** | Provides NT LAN Manager (NTLM) access authentication within the Window Security Support Provider (SSPI) framework.<br>Select the session mode.<br>■ If you select **Shared Session**, provide credentials for the shared session.<br>■ If you select **Per User Session**, the Orchestrator client retrieves credentials from the user who is logged in.<br>Provide the NTLM settings. |

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

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

### What to do next

You can explore the SOAP host objects and run workflows on them from the **Inventory** view.

## Generate a New Workflow from a SOAP Operation

You can create a custom workflow from a SOAP operation.

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

### Prerequisites

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

### Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, select **Library > SOAP** to navigate to the Generate a new workflow from a SOAP operation workflow.
3. Right-click the Generate a new workflow from a SOAP operation workflow and select **Start workflow**.
4. Select the SOAP operation from the list of available operations.
5. In the **Name** text box, type the name of the workflow to generate.
6. Select the workflow folder in which to generate the new workflow.

   You can select any existing folder from the workflow library.
7. Click **Submit** to run the workflow.

### What to do next

You can test the generated workflow.

### Test a Custom-Generated Workflow

You can run a custom workflow generated from a SOAP operation to get the output parameters of the operation.

#### Prerequisites

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

#### Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. Navigate to the workflow location.
3. Right-click the custom workflow and select **Start workflow**.
4. Provide the input parameters that the SOAP operation requires.
5. Click **Submit** to run the workflow.
6. (Optional) In the **Logs** tab, review the list of available output parameters.

## Invoke a SOAP Operation

You can call a SOAP operation directly, without generating a new workflow.

#### Prerequisites

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

#### Procedure

1. Click the **Workflows** view in the Orchestrator client.
2. In the workflows hierarchical list, select **Library > SOAP** and navigate to the Invoke a SOAP operation workflow.
3. Right-click the Invoke a SOAP operation workflow and select **Start workflow**.
4. Select the SOAP operation from the list of available operations.
5. Provide the input parameters that the SOAP operation requires.
6. Click **Submit** to run the workflow.
7. (Optional) In the **Logs** tab, review the list of available output parameters.

## Exporting and Importing SOAP Configuration

You can transfer the configuration of the SOAP plug-in from one Orchestrator instance to another by creating a configuration package. You can export all plug-in elements, such as hosts and custom workflows, or export a selection of elements.

You cannot update existing SOAP hosts by importing a configuration package. If you want to import a new version of a host, you must remove the existing host and all elements related to it. You can back up the existing host by exporting it as a package.

---

NOTE You can transfer only the plug-in configuration. The SSL certificates of the hosts are not transferred. You can import the SSL certificates manually by starting an update workflow on the host. This starts the certificate import on the new Orchestrator instance.

---

If a configuration package contains a SOAP host that has the same name as an existing host, but has different content, you should rename the existing host before importing the package.

### Export SOAP Configuration as a Package

You can export elements of the SOAP plug-in configuration as an Orchestrator package.

#### Prerequisites

Verify that you are logged in to the Orchestrator client as an administrator.

#### Procedure

1. Click the **Packages** view in the Orchestrator client.
2. From the **Packages** drop-down menu, select **Add package**.
3. Type the name of the package and click **Ok**.
4. Right-click the package that you added and select **Edit**.

5 From the **Resources** tab, select **Insert Resource Element (tree browsing)**.

| Option | Action |
|---|---|
| **Add all available SOAP hosts** | In the hierarchical list, select **Library > SOAP** and click **Select**. |
| **Add a specific SOAP host** | Expand the hierarchical list to **Library > SOAP**, select a SOAP host, and click **Select**. |

6 (Optional) From the **Workflows** tab, add custom SOAP plug-in workflows.

7 (Optional) From the **General** tab, provide a description of the package.

8 Click **Save and close**.

9 Right-click the edited package and select **Export package**.

10 Type the file name, select a location, and click **Save**.

## Import SOAP Configuration from a Package

You can import elements of the SOAP plug-in configuration from an Orchestrator package.

### Prerequisites

- Verify that you are logged in to the Orchestrator client as an administrator.
- Verify that the SOAP plug-in is installed on the Orchestrator instance.

### Procedure

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

2 From the **Packages** drop-down menu, select **Import package**.

3 Browse to select the package to import and click **Open**.

  Certificate information about the exporter appears.

4 Review the package import details and select **Import** or **Import and trust provider**.

  The **Import package** view appears. If the version of the imported package element is later than the server version, the system selects the element for import.

5 Click **Import checked elements**.

  The imported package appears in the list of packages.

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

7 In the workflows hierarchical list, select **Library > SOAP > Configuration** to navigate to the Reload plug-in configuration workflow.

8 Right-click the Reload plug-in configuration workflow and select **Start workflow** to refresh the list of SOAP hosts.

  The imported SOAP hosts appear in the **Inventory** view.

# 3 SOAP Plug-In Scripting API

The SOAP plug-in scripting API contains classes, with their respective attributes and methods, that allow interaction between vCenter Orchestrator and SOAP hosts. You can use the API to develop custom workflows that interact with SOAP hosts.

This chapter includes the following topics:

## Access the SOAP Plug-In API

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

#### Procedure

1 Log in to the Orchestrator client as an administrator.

2 Access the API Explorer from either the Orchestrator client or from the **Scripting** tabs of the workflow, policy, and action editors.

   - To access the API Explorer from the Orchestrator client, click **Tools > API Explorer** in the Orchestrator client toolbar.
   - To access the API Explorer from the **Scripting** tabs of the workflow, policy, and action editors, click **Search API** on the left.

3 To expand the hierarchical list of SOAP plug-in API objects, double-click the **SOAP** module in the left pane.

#### What to do next

You can copy code from API elements and paste it into scripting boxes. For more information about API scripting, see the *vCenter Orchestrator Developer's Guide*.

## SOAP Plug-In API Classes

The SOAP plug-in exposes JavaScript API classes related to SOAP object management.

### SOAPAuthentication Class

The `SOAPAuthentication` class contains attributes and methods related to authentication information for a SOAP host.

The `SOAPAuthentication` class defines the following attributes.

| Attribute | Returns | Description |
|---|---|---|
| rawAuthProperties | Array of String | Authentication attributes |
| type | String | Authentication type |

The SOAPAuthentication class defines the following method.

| Method | Returns | Description |
|---|---|---|
| getRawAuthProperty(number):String | String | Gets the specified authentication attribute. |

## SOAPAuthenticationManager Class

The SOAPAuthenticationManager class contains methods related to managing SOAP host authentication objects.

The SOAPAuthenticationManager class defines the following methods.

| Method | Returns | Description |
|---|---|---|
| createAuthentication(String,String[]):Authentication | Authentication | Creates an authentication instance. |
| getSOAPAuthentications():String[] | Array of String | Returns all available auhentication types. |
| getSessionModes():String[] | Array of String | Returns the supported session modes. |

## SOAPDynamicInParameter Class

The SOAPDynamicInParameter class represents a dynamic input-parameter object to be used from scripting.

## SOAPDynamicOutParameter Class

The SOAPDynamicOutParameter class represents a dynamic output-parameter object to be used from scripting.

## SOAPHost Class

The SOAPHost class contains attributes and methods related to the representation of an external system as a Web service interface.

The SOAPHost class defines the following attributes.

| Attribute | Returns | Description |
|---|---|---|
| authentication | SOAPAuthentication | Authentication information |
| connectionTimeout | Number | Connection timeout in seconds |
| id | String | Identifier of the host |
| name | String | Name of the host |
| requestTimeout | Number | Request timeout in seconds |
| wsdlFileContent | String | WSDL file content |
| wsdlFileLocal | Boolean | True if the WSDL content is provided locally, False if it is derived from URL |
| wsdlUri | String | URI to the WSDL file of the external service |

The SOAPHost class defines the following methods.

| Method | Returns | Description |
|---|---|---|
| `createWorkflow(String,String,WorkflowCategory):Workflow` | `Workflow` | Creates a new workflow for a SOAP operation. |
| `getOperation(String):SOAPOperation` | `SOAPOperation` | Gets a SOAP operation by name. |
| `getOperations():String[]` | `Array` of `String` | Gets all SOAP operation names. |

## SOAPHostManager Class

The `SOAPHostManager` class contains methods related to CRUD operations for SOAP hosts.

The `SOAPHostManager` class defines the following methods.

| Method | Returns | Description |
|---|---|---|
| `addHost(SOAPHost):SOAPHost` | `SOAPHost` | Adds a SOAP host to the plug-in's inventory. |
| `getHost(String):SOAPHost` | `SOAPHost` | Returns the SOAP host with the specified name from the plug-in's inventory. |
| `getHosts():String[]` | `Array` of `String` | Returns the list of SOAP host names from the plug-in's inventory. |
| `reloadConfiguration():Object` | `Object` | Reloads the plug-in configuration. |
| `removeHost(String):SOAPHost` | `SOAPHost` | Removes a SOAP host from the plug-in's inventory. |
| `updateHost(String,SOAPHost):SOAPHost` | `SOAPHost` | Updates the specified SOAP host in the plug-in's inventory. |

## SOAPHostValidator Class

The `SOAPHostValidator` class contains attributes and methods related to URL validation.

The `SOAPHostValidator` class defines the following attribute.

| Attribute | Returns | Description |
|---|---|---|
| `url` | `String` | URL to validate |

The `SOAPHostValidator` class defines the following methods.

| Method | Returns | Description |
|---|---|---|
| `getCertificateInfo():Properties` | `Properties` | Retrieves the server's certificate information as a string. |
| `installCertificates():Object` | `Object` | Installs the server's certificate into the JSSE keystore. |

## SOAPInterceptor Class

The `SOAPInterceptor` class contains attributes and methods related to JavaScript handler functions that intercept SOAP requests and SOAP responses.

The `SOAPInterceptor` class defines the following methods.

| Method | Returns | Description |
|---|---|---|
| `setRequestBodyInterceptor(InterceptorHandler):Object` | `Object` | Sets a handler function to intercept the SOAP request body. |
| `setRequestHeaderInterceptor(InterceptorHandler):Object` | `Object` | Sets a handler function to intercept the SOAP request header. |

| Method | Returns | Description |
|---|---|---|
| `setResponseBodyInterceptor(InterceptorHandler):Object` | `Object` | Sets a handler function to intercept the SOAP response body. |
| `setResponseHeaderInterceptor(InterceptorHandler):Object` | `Object` | Sets a handler function to intercept the SOAP response header. |

## SOAPOperation Class

The `SOAPOperation` class contains attributes and methods related to generating SOAP operation templates from WSDL definition.

The `SOAPOperation` class defines the following attribute.

| Attribute | Returns | Description |
|---|---|---|
| `name` | `String` | Name of the SOAP operation |

The `SOAPOperation` class defines the following methods.

| Method | Returns | Description |
|---|---|---|
| `createSOAPRequest():SOAPRequest` | `SOAPRequest` | Creates a new SOAP request for the SOAP operation. |
| `getHost():SOAPHost` | `SOAPHost` | Gets the parent SOAP host of the SOAP operation. |
| `getInHeaders():String[]` | `Array` of `String` | Gets a flat representation of the SOAP operation's input headers. |
| `getInParameters():String[]` | `Array` of `String` | Gets a flat representation of the SOAP operation's input parameters. |
| `getOutParameters():String[]` | `Array` of `String` | Gets a flat representation of the SOAP operation's output parameters. |
| `invoke(SOAPRequest):SOAPResponse` | `SOAPResponse` | Calls a SOAP operation. |
| `invokeWithInterceptor(SOAPRequest,SOAPInterceptor):SOAPResponse` | `SOAPResponse` | Calls a SOAP operation with message interceptor. |

## SOAPRequest Class

The `SOAPRequest` class contains methods related to passing a SOAP request when a SOAP operation is called.

The `SOAPRequest` class defines the following methods.

| Method | Returns | Description |
|---|---|---|
| `addInHeaderAttribute(String,String,String):Object` | `Object` | Adds an attribute to the input header. |
| `addInPrameterAttribute(String,String,String):Object` | `Object` | Adds an attribute to the input parameter in the request. |
| `addRawHeader(String):Object` | `Object` | Adds a raw XML header to the request. |

| Method | Returns | Description |
|---|---|---|
| `setInHeader(String,Object):Object` | `Object` | Sets an input header in the request. |
| `setInParameter(String,Object):Object` | `Object` | Sets an input parameter in the request. |

## SOAPResponse Class

The `SOAPResponse` class contains methods related to receiving a SOAP response when a SOAP operation is called.

The `SOAPResponse` class defines the following methods.

| Method | Returns | Description |
|---|---|---|
| `getOutHeader(String):Object` | `Object` | Gets an output header value from the response. |
| `getOutHeaderAttribute(String,String):String` | `String` | Gets the attribute value from the given output header and attribute name. |
| `getOutHeaderAttributes(String):String[]` | `Array` of `String` | Gets the attribute names from the given output header. |
| `getOutHeaders():String[]` | `Array` of `String` | Gets the full names of all output headers. |
| `getOutParameter(String):Object` | `Object` | Gets an output parameter value from the response. |
| `getOutParameterAttribute(String,String):String` | `String` | Gets the attribute value from the given output parameter and attribute name. |
| `getOutParameterAttributes(String):String[]` | `Array` of `String` | Gets the attribute names from the given output parameter. |
| `getOutParameters():String[]` | `Array` of `String` | Gets the full names of all output parameters. |

# Index