Installing and Using the VMware vCenter Orchestrator Plug-In SDK

vCenter Orchestrator 5.5

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-001138-00

vmware®

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

Installing and Using the VMware vCenter Orchestrator Plug-In SDK

Installing and Using the VMware vCenter Orchestrator Plug-In SDK provides information about installing the VMware vCenter Orchestrator Plug-in SDK and using its functionality in the Eclipse interface.

Intended Audience

This information is intended for plug-in developers who are familiar with virtual machine technology, datacenter operations, vCenter Orchestrator, and the Eclipse framework.

Getting Started with the vCenter Orchestrator Plug-In SDK 1

By using the vCenter Orchestrator Plug-in SDK (vCO Plug-in SDK) within the Eclipse for Java Developers IDE, you can create a Orchestrator plug-in project, customize the plug-in, and package it.

The vCO Plug-in SDK is integrated with the Eclipse for Java Developers IDE. Formerly, Orchestrator plug-in development was only available by command line.

This chapter includes the following topics:

Introduction to the vCO Plug-In SDK

The vCO Plug-in SDK integrates Orchestrator plug-in examples and APIs into the Eclipse for Java Developers IDE. This allows you to take full advantage of the features of the integrated Eclipse Java Development Tools, such as Java editor, auto-build, and Java package explorer.

The Orchestrator plug-in API provides Java interfaces that you implement to create the plug-in adapter and plug-in factory. The plug-in adapter and factory expose the objects and operations of the plugged-in technology to the Orchestrator server.

The plug-in API includes interfaces, classes, and annotations that you can use when you create the plug-in adapter, factory, and event management implementations.

vCO Plug-In SDK Hardware and Software Prerequisites

Verify that your system meets the minimum hardware and software requirements before you install the vCO Plug-in SDK.

- A 64-bit computer for development of your plug-in
- Java Development Kit 6 or later
- vCenter Orchestrator 5.5
- Eclipse

vCO Plug-In SDK Supported Platforms

To be able to install and use the vCO Plug-in SDK, you must install a supported version of Eclipse on a supported operating system.

Supported Eclipse Versions

- Eclipse 4.3 (32-bit and 64-bit)
- STS 3.3 (32-bit and 64-bit)

Supported Operating Systems

- Windows 8 (64-bit)
- Windows 7 (64-bit)
- RedHat 6 (64-bit)
- CentOS 6 (64-bit)
- Mac OS X Mountain Lion 10.8.4 (64-bit)

Set the HTTP and HTTPS Proxies in Eclipse

The first time you start working with Orchestrator in Eclipse, you must set proxies for HTTP and HTTPS as well as proxy bypasses for local machines that you will connect to.

Procedure

1 In the Eclipse interface, select **Window > Preferences**.

2 Open **General** and select **Network Connections**.

3 From the **Active Provider** drop-down menu, select **Manual**.

4 Set the HTTP proxy settings.

   a Select **HTTP** and click **Edit**.

   b Verify the proxy server name and port number are correct, or type the correct proxy and port information.

   c Click **OK**.

5 Set the HTTPS proxy settings.

   a Select **HTTPS** and click **Edit**.

   b Verify the proxy server name and port number are correct, or type the correct proxy and port information.

   c Click **OK**.

CAUTION Do not edit the SOCKS proxy. Modifying the SOCKS proxy might cause errors.

6 In the **Proxy bypass** list, add systems on your intranet that you want Eclipse to be able to find.

The following example shows various types of entries you might want to add.

```
10.*
*.example.com
127.0.0.1
localhost
```

Wildcards can be used for IP addresses and domain names. Domain names can only have three parts.

7 Click **Apply** to save changes, and click **OK**.

Install the vCO Plug-In SDK

To be able to use the vCO Plug-in SDK, you must download a ZIP file and install it in the Eclipse interface.

The vCO Plug-in SDK requires Workbench IS to function properly. Both the Workbench IS and vCO Plug-in SDK `.zip` files are available in the `VMware-VCO-Plug-In-SDK-5.5.0-`*`build_number`*`.zip` file, which you can download from the VMware Communities site.

Prerequisites

- Obtain the `VMware-VCO-Plug-In-SDK-5.5.0-`*`build_number`*`.zip` file from the VMware Communities site.
- Verify that you have installed the `UpdateSite-WBIS-com.vmware.vide-3.0.0.`*`build_number`*`.zip` file in Eclipse.

Procedure

1 In the Eclipse interface, select **Help > Install New Software**.

2 Click **Add**.

3 In the **Location** text box, provide the URL of the `UpdateSite-VCOPSK-Kepler-com.vmware.vide-3.0.0.`*`build_number`*`.zip` archive and click **OK**.

4 Select the check box next to **VMware Workbench vCenter Orchestrator Starter Kit**, and click **Next**.

This starts the install process and also downloads and installs the necessary Eclipse plug-ins and features.

5 Review the **Install Details** and click **Next**.

6 To continue with the installation, review the license notice, select **I accept the terms of the license agreements**, and click **Finish**.

You cannot install the software without agreeing to the license terms. The installation can take several minutes to complete.

When the installation is complete, a dialog box advises you to restart Eclipse for the changes to take effect.

7 Click **Restart Now**.

Using the vCenter Orchestrator Plug-In SDK 2

You can use the vCO Plug-in SDK to create plug-in projects from an existing Java library, an inventory definition, or from samples, as well as copy and paste project files.

This chapter includes the following topics:

Create an Orchestrator Plug-in Project from Wizard

You can create a plug-in project by using an existing Java library, an inventory definition, or you can create an empty plug-in project.

You can generate an Orchestrator plug-in from an external Java library, which contains a simple set of classes with methods and attributes that you want to expose within Orchestrator as scripting objects. After you generate the plug-in and install it in Orchestrator, you can use those scripting objects inside your actions and workflows.

If you want to start developing a plug-in based on what users see in the Inventory view of the Orchestrator client, you can generate an Orchestrator plug-in from a simple inventory structure definition. The inventory defines the set of objects that are directly accessible to the plug-in users, as well as the relationships between those objects.

Prerequisites

- Review the plug-in attributes that you can modify. See "Orchestrator Plug-in Project Attributes," on page 13.
- If you create a project based on an existing Java library, review the options for ignoring classes and methods. See "Ignored Classes and Methods," on page 13.
- If you create a project based on an inventory definition, review the example XML syntax. See "Inventory Definition Examples," on page 14.
- If you use the Maven build tool, install Apache Maven 3.0.4 or later, verify that the `M2_HOME` and `JAVA_HOME` variables are set correctly on your system, and install the Maven Integration plug-in (m2e) 1.4 in Eclipse.

Procedure

1 If the Package Explorer pane is not showing, open the Java Perspective in the Eclipse interface by selecting **Window > Open Perspective > Java**.

2 Right-click in the Package Explorer pane and select **New VMware Project/File > Development Kit Projects**.

3 In the New VMware Development Kit Project wizard, expand **vCenter Orchestrator Plug-in Development**.

4 Select **Create a vCenter Orchestrator plug-in project from wizard** and click **Next**.

5 On the New VMware Development Kit Project page, type the project name in the **Project name** text box.

By default, the project directory is created under the workspace, and has the specified project name.

6 (Optional) To create the project in a custom location, deselect the **Use default location** check box and specify a custom location.

7 Click **Next**.

8 In the Project Information page, you can specify plug-in name, alias, default version number, and default package name.

9 From the **Preferred build tool** drop-down menu, select the type of build tool that you want to use.

Option	Description
ANT	The generated plug-in is compatible with the Apache Ant build tool.
MAVEN	The generated plug-in is compatible with the Apache Maven build tool.

10 From the **Preferred template** drop-down menu, select the type of template that you want to use.

Option	Description
standard-template	Allows you to use standard features.
spring-template	Allows you to use Spring features.

11 Select the type of plug-in project that you want to create.

Option	Action
Create a plug-in project by using both an existing Java library and an inventory definition	a Click **Source Library**. b Click **Browse** to locate the Java library `.jar` file. **NOTE** You must provide the absolute path to the `.jar` file in the **Source Library** text box. c Specify the ignored classes, methods, and method classes that you do not want to expose to plug-in users. **NOTE** You must specify fully qualified names for ignored classes and method classes, and simple method names for ignored methods. d Click **OK**. e Click **Inventory XML**. f In the **Inventory XML** text box, provide the inventory in XML form. g Click **OK**.
Create a plug-in project by using an existing Java library	a Deselect the **Generate the plug-in inventory with mocked objects** check box. b Click **Source Library**. c Click **Browse** to locate the Java library `.jar` file. **NOTE** You must provide the absolute path to the `.jar` file in the **Source Library** text box. d Specify the ignored classes, methods, and method classes that you do not want to expose to plug-in users. **NOTE** You must specify fully qualified names for ignored classes and method classes, and simple method names for ignored methods. e Click **OK**.

Option	Action
Create a plug-in project by using an inventory definition	a Deselect the **Wrap classes from an existing Java library as plug-in objects** check box. b Click **Inventory XML**. c In the **Inventory XML** text box, provide the inventory in XML form. d Click **OK**.
Create an empty plug-in project	Deselect the **Wrap classes from an existing Java library as plug-in objects** and **Generate the plug-in inventory with mocked objects** check boxes.

12 Click **Finish**.

Project creation starts and compiles the project.

Orchestrator Plug-in Project Attributes

When you create a new Orchestrator plug-in project, you must provide values for some plug-in attributes.

The following table explains the standard plug-in attributes and provides example values.

Attribute	Description	Example
Name	The name of the plug-in that appears in Orchestrator.	My New Plug-in
Alias	An alias or short name for the plug-in. The alias is also used as a prefix for some of the generated components like core classes or scripting objects. Each plug-in must have a unique alias.	MyPlugin
Version	The initial version of the plug-in.	1.0.0
Package	The root Java package for the generated classes.	com.example.myplugin

Ignored Classes and Methods

When you create a new Orchestrator plug-in project based on an existing Java library, you can specify ignored classes, methods, and method classes that you do not want to expose to plug-in users.

The following table describes the different types of ignored attributes.

Attribute	Description
Ignored Classes	List of classes from the Java library to be ignored from the auto-generation. No inventory classes or scripting objects are generated for them.
Ignored Methods	List of method names from the methods to be ignored from the auto-generation. Any method from any class or interface that matches any of these names is ignored. Those methods will not be available from the scripting API.
Ignored Methods Classes	List of classes and interfaces whose methods are ignored from the auto-generation. Those methods will not be generated for classes that extend those classes or implement those interfaces. You can use this attribute to ignore unnecessary methods from common or base classes and interfaces, such as `java.lang.Object` or `java.lang.Enum`.

Inventory Definition Examples

When you create a plug-in project based on an inventory definition, you must provide inventory XML code.

Example Code

The following is an XML code example that defines the inventory structure for hosts.

```
<inventory>
    <item type="Host">
        <item type="Network"/>
        <item type="Datacenter">
            <item type="Media"/>
            <item type="VApp">
                <item type="Vm"/>
            </item>
            <item type="VAppTemplate"/>
        </item>
    </item>
</inventory>
```

Inventory Structure

This inventory definition creates the following inventory structure.

```
+ Host
  + Network
  + Datacenter
    + Media
    + VApp
      + Vm
    + VAppTemplate
```

In this structure, the hosts are the root entities of the plug-in. Each host contains a list of networks and datacenters. Each datacenter contains a list of media, vApps, and vApp templates. Each vApp contains a list of virtual machines.

Example Code

The following is an XML code example that defines the inventory structure for vCloud Director hosts.

```
<inventory>
    <item type="VCloudHost">
      <item type="Org">
        <item type="Catalog">
          <item type="CatalogItem"/>
        </item>
        <item type="OrgNetwork"/>
        <item type="Vdc">
            <item type="Media"/>
            <item type="VApp">
                <item type="Vm"/>
            </item>
            <item type="VAppTemplate"/>
```

```
          </item>
        </item>
    </item>
</inventory>
```

Generated Workflows

This inventory definition creates the following workflows in the **Library > *Plug-in_Name* > Configuration** Orchestrator workflow folder.

- Add a VCloudHost
- Remove a VCloudHost
- Update a VCloudHost

Generated Actions

This inventory definition creates the following actions in the `com.vmware.library.plug-in_alias.configuration` Orchestrator package.

- addVCloudHost
- removeVCloudHost
- updateVCloudHost

Create an Orchestrator Plug-in Project from Samples

You can create a plug-in project from sample code.

To create a plug-in, you must have an application to expose for Orchestrator to manage. If you want to familiarize yourself with the Orchestrator plug-in API, VMware supplies a sample application and plug-in for you to experiment with, the Solar System sample.

Procedure

1 If the Package Explorer pane is not showing, open the Java Perspective in the Eclipse interface by selecting **Window > Open Perspective > Java**.

2 Right-click in the Package Explorer pane and select **New VMware Project/File > Development Kit Projects**.

3 In the New VMware Development Kit Project wizard, expand **vCenter Orchestrator Plug-in Development**.

4 Expand **Create vCenter Orchestrator plug-in project from samples**, select a project, and click **Next**.

Option	Description
Create Hello World plug-in project	Creates a simple Hello World plug-in project.
Create Solar System plug-in project	Creates a Solar System plug-in project that demonstrates the basic concepts of Orchestrator plug-in development.
Create Solar System REST plug-in project	Creates a Solar System plug-in project from which you can generate a plug-in that uses RESTful services to communicate with a remote server.
Create Solar System SOAP plug-in project	Creates a Solar System plug-in project from which you can generate a plug-in that uses the SOAP protocol to communicate with a remote server.

5 On the New VMware Development Kit Project page, type the project name in the **Project name** text box.

By default, the project directory is created under the workspace, and has the specified project name.

6 (Optional) To create the project in a custom location, deselect the **Use default location** check box and specify a custom location.

7 Click **Finish**.

Project creation starts and compiles the project.

Create an Orchestrator Service or Client Project from Samples

You can create a service or client project from sample code.

To create a plug-in, you must have an application to expose for Orchestrator to manage. If you want to familiarize yourself with the Orchestrator plug-in API, VMware supplies a sample application and plug-in for you to experiment with, the Solar System sample.

Procedure

1 If the Package Explorer pane is not showing, open the Java Perspective in the Eclipse interface by selecting **Window > Open Perspective > Java**.

2 Right-click in the Package Explorer pane and select **New VMware Project/File > Development Kit Projects**.

3 In the New VMware Development Kit Project wizard, expand **vCenter Orchestrator Plug-in Development**.

4 Expand **Create vCenter Orchestrator service or client project from samples**, select a project, and click **Next**.

Option	Description
Create Solar System REST service project	Creates a simple REST service support project that helps to demonstrate the basic concepts of Orchestrator plug-in development when developing a plug-in that interacts with a third-party REST service.
Create Solar System SOAP client project	Creates a simple SOAP client support project that helps to demonstrate the basic concepts of Orchestrator plug-in development when developing a plug-in that interacts with a third-party SOAP service.
Create Solar System SOAP service project	Creates a simple SOAP service support project that helps to demonstrate the basic concepts of Orchestrator plug-in development when developing a plug-in that interacts with a third-party SOAP service.

5 On the New VMware Development Kit Project page, type the project name in the **Project name** text box.

By default, the project directory is created under the workspace, and has the specified project name.

6 (Optional) To create the project in a custom location, deselect the **Use default location** check box and specify a custom location.

7 Click **Finish**.

Project creation starts and compiles the project.

Copy and Paste Project Files

To view your plug-in in Orchestrator, you can copy and paste files from your Orchestrator plug-in project to your Orchestrator server platform by using the Remote System Explorer perspective.

Procedure

1 In the Package Explorer pane, copy the plug-in DAR file from the `dist` folder in your Orchestrator project.

2 In the Eclipse interface, select **Window > Open Perspective > Other > Remote System Explorer**.

The Remote System Explorer perspective appears.

3 In the Remote Systems pane, select the folder in which the Orchestrator plug-ins' DAR files are located.

The default location for Windows is `C:\Program Files\VMware\Infrastructure\Orchestrator\app-server\server\vmo\plugins`.

NOTE You can use locations for other supported platforms.

4 Paste the plug-in DAR file in the selected folder.

Index