

Customizing the vSphere Client

VMware vSphere Web Services SDK 4.1

The `ExtensionManager` service interface supports centralized management of plug-ins (sometimes called extensions) for vCenter Server systems. Plug-ins have both server and client components. A server plug-in component extends vCenter Server capability in some way. For example, the experimental Storage Monitoring Service (SMS) provided with vCenter Server 4.1 has a server plug-in component that uses information from the vCenter Server database to provide information about storage. SMS has a client component that extends the vSphere Client with text and graphical user interface (GUI) items that support user interaction with the server. SMS reports about storage are available on a storage tab and in several context-menus in the vSphere Client. Both the `The ExtensionManager` contains the information needed by both server and client to support the features of any specific plug-in.

You can use the `ExtensionManager` in an application to add your own product-specific menu selections, tabs, toolbar, and view icons to the vSphere Client. For example, you might add a custom button that directs users to your company Web site, invokes a JSP application, or executes a script on the vCenter Server. The plug-ins you create must be defined in a configuration file and registered with the `ExtensionManager` as described in this document.

This type of plug-in is sometimes called a scripting plug-in because it supports running scripts on the vCenter Server, from a vSphere Client. Scripting plug-ins have an `Extension` object that specifies server extension type of `com.vmware.vim.viClientScripts`. Scripting plug-ins require that all components, including the configuration file, are available using the HTTP/HTTPS protocol, with their locations specified as URLs.

This technical note does not include information about customizing the UI by using C# or other libraries that make-up the vSphere Client. This document includes the following topics:

- [“Overview of vSphere Client Plug-Ins”](#) on page 1
- [“Understanding vSphere Client UI Extension Points”](#) on page 3
- [“Creating the Configuration File”](#) on page 7
- [“Overview of the ExtensionManager Managed Object”](#) on page 11
- [“Registering the Extension”](#) on page 11

Overview of vSphere Client Plug-Ins

You extend the vSphere Client by identifying the Web-server based applications, scripts, or other components that you want to make available through the vSphere Client. You create a configuration file that references each of the components and identifies where in the vSphere Client each should display at runtime. In the configuration file, you define each of these components as an `extension` element. Using the attributes and other elements defined in [“Creating the Configuration File”](#) on page 7, you configure the properties of each extension element, including the URL of the Web server where the component is located. For example, for a script that runs on the vCenter Server, the extension element defined in the configuration file includes the URL of the vCenter Server with the appropriate syntax for running the script.

Finally, you register information about the plug-in, including the location of the configuration file, with the `ExtensionManager` on the vCenter Server. You can create a Java or C# utility to register the plug-in, or a script, or you can use the MOB (Managed Object Browser). For information about using the MOB, see the *vSphere Web Services SDK Programming Guide*.

At runtime, these components interact over the intranet in the following sequence:

- 1 Users connect to the vCenter Server as they always do, using the vSphere Client. When users authenticate successfully to vCenter Server, the `SessionManager` sends the vSphere Client a user session (`sessionId`) that identifies them to the system.
- 2 The vCenter Server looks up the list of plug-ins registered with `ExtensionManager`, obtains the location of the configuration file, redirects the vSphere Client to the location specified. These are the plug-ins available to users based on the permissions associated with their account.
- 3 The vSphere Client obtains the configuration file from the Web server. The vSphere Client parses the configuration file, obtaining the various components defined in the file from the locations specified, and populating the UI with these components.
- 4 When users select a menu item or click on a button for your plug-in, the vSphere Client connects to the Web server specified for that extension element in the configuration file, sending context information, including the `sessionId` and other details listed in [Table 1](#) in the string to the Web server.

The entire process is transparent to vSphere Client end-users. However, end-users of your plug-in can enable or disable its functionality by using the **Manage Plugins...** menu selection in the vSphere Client. For example, the Storage Views tab displays when the Storage Monitoring Service plug-in is enabled, but is removed from the UI when users disable the plug-in.

Requirements

You can implement the Web server functionality using any of the following programming languages:

- Java Servlets or Java Server Pages (JSPs)
- Microsoft Active Server Pages (ASP.NET)
- Traditional CGI Scripting
- Static or dynamic HTML pages

Your plug-in may also invoke vCenter Server scripts and scripts on virtual machines. The guest OS on the virtual machine must have VMware Tools installed to support scripting.

If your plug-in includes a script or Web application, the script or application must be able to parse the string that is sent to it by the vSphere Client. For example, if the plug-in executes a script on the vCenter Server that power cycles virtual machines, the URL specified for the extension element in the configuration file might look like the following example:

```
http://dev:8000/vmAction.cgi?cmd=powerOn
```

[Table 1](#) lists the other parameters in the string.

Table 1. vSphere Client Session Context Parameters

Parameter	Description	Example
<code>sessionId</code>	String value obtained by vSphere Client after logging in to vCenter Server. The <code>sessionId</code> associates access to the plug-in application with the same privileges as were applied at initial logon.	<code>sessionId=9241E7B8-A37B-4264-A8D1-945628F9E0D6</code>
<code>moref</code>	Managed object reference of the entity selected in the client. The type of managed object reference and its value are separated by a colon (:).	<code>moref=VirtualMachine:16</code>

Table 1. vSphere Client Session Context Parameters

Parameter	Description	Example
serviceUrl	Server name and path to the Web services API hosted on vCenter Server.	serviceUrl=https://serverFQDN/sdk
locale	Name of the locale configured for the vSphere Client. Used for localizing content from the Web server.	locale=en locale=ja locale=de

Your application or script must be able to parse a string consisting of the parameters shown in [Table 1](#). For example, if the extension runs a script on the vCenter Server, the full string might look like the following string:

```
http://dev:8000/vmAction.cgi?cmd=powerOn&moref=VirtualMachine:16&sessionId=9241E7B8-A37B-4264-A8D1-945628F9E0D6&locale=en&serviceUrl=https://localhost/sdk
```

Understanding vSphere Client UI Extension Points

The vSphere Client is the principal user interface for administering vCenter Server and ESX/ESXi. Plug-ins are not supported on ESX/ESXi systems. The vSphere Client user interface configuration is dynamic in that it varies depending on the specifics of the server to which it is connected. When the server is a vCenter Server system, the vSphere Client displays all the options available to the vSphere environment as defined in the licensing configuration and user permissions, including plug-ins.

When users first log in to a vCenter Server system from the vSphere Client, it displays a Home page with icons for accessing various vSphere Client functions. When users log out of the vCenter Server, the vSphere Client retains the closing view. The next time users log in to vCenter Server, they are returned to the last view that was displayed.

Each of the following areas of the vSphere Client has specific places, called extension points, that you can extend with the specific UI components for your plug-in:

- Home view
- Inventory views
- Main menus
- Inventory menus
- Toolbars

Lists of extension points available in each of these areas are contained in [Table 2](#) through [Table 6](#) starting on [page 4](#). To customize the UI at one of these extension points, you add an extension element to the configuration file and set its parent attribute to the appropriate extension point. See [“Creating the Configuration File”](#) on [page 7](#) for complete information.

vSphere Client Home View

The vSphere Client home view displays shortcuts for Inventory, Administration, Management, and Applications features. The extension points defined for this view include the HomeView.Inventory, HomeView.Admin, and HomeView.Management as shown in [Figure 1](#), as well as HomeView.Applications.

To add an icon to one of these areas, you define an extension element in the configuration file and set the parent attribute of the element to the appropriate extension point. Other details about how to create the configuration and define the elements needed are included in [“Creating the Configuration File”](#) on [page 7](#).

Figure 1. vSphere Client Home View and Some Extension Points

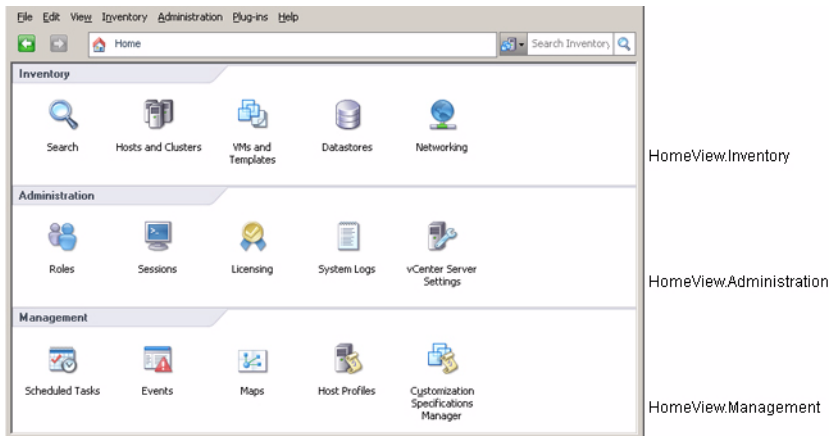


Table 2 lists the available view areas and defined extension points.

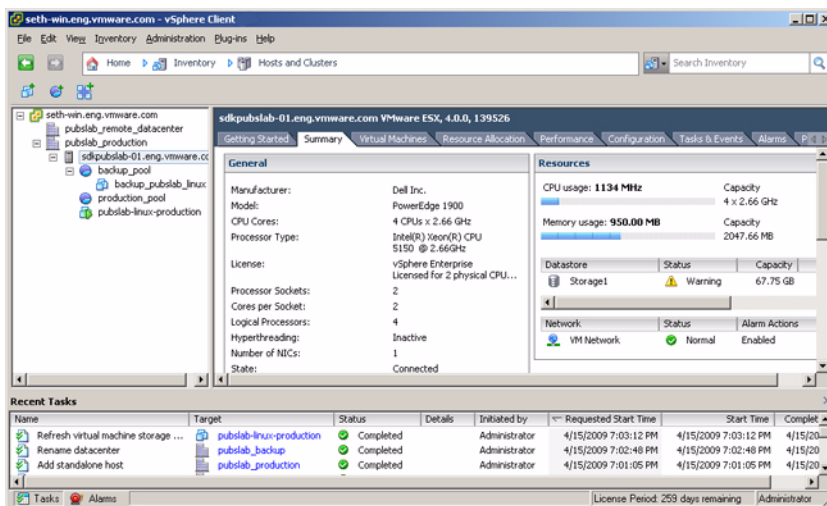
Table 2. Home View Extension Points

Display area of home view	Set parent attribute to extension point
Administration	HomeView.Admin
Applications	HomeView.Applications
Inventory	HomeView.Inventory
Management	HomeView.Management

vSphere Client Inventory View

The vSphere Client inventory view displays entities available in the inventory in its left pane and the operations available for selected inventory entities in its right pane. You can add a tab to this view by associating it with one of the inventory objects in the left-pane.

Figure 2. vSphere Client Inventory View



To add a Tab that displays in the right pane for a specific inventory object, you must add an extension to the configuration file that defines the parent attribute as the appropriate extension point from among the InventoryView.* extension points listed in Table 3.

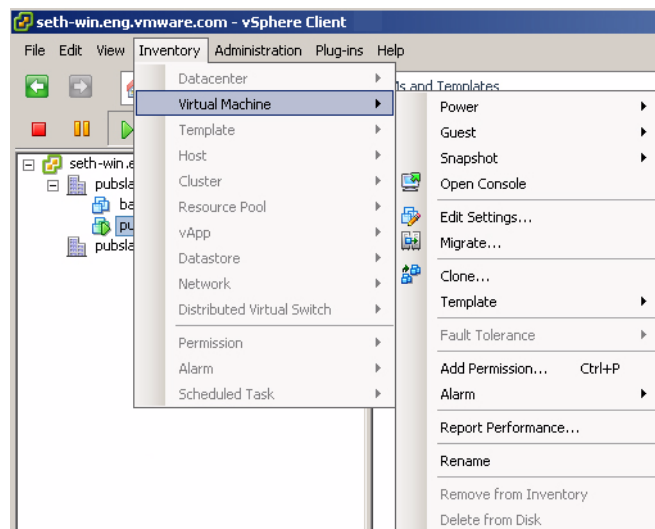
Table 3. InventoryView Extension Points

Inventory object to associate with the Tab	Set parent attribute to extension point
Cluster	InventoryView.Cluster
Datacenter	InventoryView.Datacenter
DatacenterFolder	InventoryView.DatacenterFolder
Datastore	InventoryView.Datastore
DatastoreFolder	InventoryView.DatastoreFolder
DVS	InventoryView.DistributedVirtualSwitch
Host	InventoryView.HostSystem
HostFolder	InventoryView.ComputeResourceFolder
Network	InventoryView.Network
NetworkFolder	InventoryView.NetworkFolder
ResourcePool	InventoryView.ResourcePool
Template	InventoryView.Template
VirtualMachine	InventoryView.VirtualMachine
VirtualMachineFolder	InventoryView.VirtualMachineFolder
VirtualApp	InventoryView.VirtualApp

vSphere Client Context Menus

The menu options and UI items available on the vSphere Client change as different managed objects are selected in the left pane of the interface, or as different menu choices are made in the main menu, at the top of the display.

For example, [Figure 3](#) shows the default context menu available for virtual machines from the Inventory menu of the vSphere Client.

Figure 3. vSphere Client Inventory Context Menu

You can extend the menus available at any managed object by using one of the contextual extension points listed in the tables. Add one extension element for each type of context menu you want to add to the UI. To add custom context menus to the top menu of the UI, use the extension points listed in [Table 4](#).

Table 4. Main Menu Extension Points

Inventory object for the menu	Set parent attribute to extension point
Administration	MainMenus.Administration
Edit	MainMenus.Edit
File	MainMenus.File
File_Export	MainMenus.File_Export
File_New	MainMenus.File_New
File_Report	MainMenus.File_Report
Help	MainMenus.Help
Inventory	MainMenus.Inventory
Plugins	MainMenus.Plugins
View	MainMenus.View
View_Inventory	MainMenus.View_Inventory

To add custom context menus to the left-pane of the UI, use the extension points listed in [Table 5](#).

Table 5. Inventory Menu Extension Points

Inventory object for the menu	Set parent attribute to extension point
Cluster	InventoryMenus.Cluster
Datacenter	InventoryMenus.Datacenter
DatacenterFolder	InventoryMenus.DatacenterFolder
Datastore	InventoryMenus.Datastore
DatastoreFolder	InventoryMenus.DatastoreFolder
DVS	InventoryMenus.DistributedVirtualSwitch
Host	InventoryMenus.HostSystem
HostFolder	InventoryMenus.ComputeResourceFolder
Network	InventoryMenus.Network
NetworkFolder	InventoryMenus.NetworkFolder
ResourcePool	InventoryMenus.ResourcePool
Role	InventoryMenus.Role
ScheduledTasks	InventoryMenus.ScheduledTasks
Tasks	InventoryMenus.Tasks
Template	InventoryMenus.Template
VirtualMachine	InventoryMenus.VirtualMachine
VirtualMachineFolder	InventoryMenus.VirtualMachineFolder
VirtualApp	InventoryMenus.VirtualApp

vSphere Client Inventory Toolbars

The Toolbars that display in the UI are specific to the managed object selected in the left-hand pane. To add an icon to the Toolbar associated with the appropriate inventory object, add an extension element to the configuration file that defines the parent attribute using the appropriate extension point from [Table 6](#). Include an iconSmaller element that specifies the URL location of the icon (the image file).

Table 6. Toolbar Extension Points

Inventory object to associate with the Toolbar	Set parent attribute to extension point
Cluster	Toolbars.Cluster
Datacenter	Toolbars.Datacenter
DatacenterFolder	Toolbars.DatacenterFolder
Datastore	Toolbars.Datastore
DatastoreFolder	Toolbars.DatastoreFolder
DVS	Toolbars.DistributedVirtualSwitch
Host	Toolbars.HostSystem
HostFolder	Toolbars.ComputeResourceFolder
Network	Toolbars.Network
NetworkFolder	Toolbars.NetworkFolder
ResourcePool	Toolbars.ResourcePool
Template	Toolbars.Template
VirtualMachine	Toolbars.VirtualMachine
VirtualMachineFolder	Toolbars.VirtualMachineFolder
VirtualApp	Toolbars.VirtualApp

Creating the Configuration File

The configuration file is a plain text file containing the series of XML elements and attributes that specify your customizations. The file must comply with a format specified by VMware. You do not need to use an XSD, but you must follow the rules defined in this section. [Table 7](#) lists valid elements in the order in which they should be defined in the file, and [Example 1](#) shows a complete example that you can use as a template.

A plug-in configuration file may contain an extension element for each UI item required to support user interaction. However, when adding menu, icon, and other UI items to the vSphere Client, follow some basic guidelines. Take into account the look and feel of the vSphere Client. Use similar fonts, font sizes, and color schemes for icons and other graphical user interface (GUI) components. Do not add too many icons to the Toolbar.

Table 7. Configuration File Elements

Element	Description
<code>scriptConfiguration</code>	<p>Required. Containing element for vSphere Client UI extensions that support scripting plug-ins. The <code><scriptConfiguration version="n.n"></code> tag must identify this as the first element in the file, with the closing tag (<code></scriptConfiguration></code>) last in file.</p> <p>The <code>version</code> attribute is required. It identifies the version of the plug-in to vSphere Client from among possibly multiple versions of the plug-in.</p>
<code>supportNonSecureCommunication</code>	<p>Optional. For non-secure HTTP connections between the vSphere Client and the plug-in Web server that is identified by the <code>url</code> element of an <code>extension</code> element. See the description of the <code>url</code> element below.</p> <p>When the vSphere Client establishes a secure connection to a plug-in Web server, the Client will pass <code>sessionId</code> and <code>webServicesSessionId</code> values in the HTTPS request. If the <code>extension</code> element specifies a standard HTTP connection, by default the vSphere Client does not pass the session identifiers to the plug-in server. To include session identifiers in a standard HTTP request, use the following statement in your configuration file.</p> <pre><supportNonSecureCommunication>true</supportNonSecureCommunication></pre>
<code>key</code>	<p>Required. Unique string that identifies the plug-in. Only one <code>key</code> element per file allowed. String value for <code>key</code> must be unique among all extensions on the vCenter Server. VMware recommends using Java package-naming convention for the key value, such as <code>com.yourcompany.yourplugin-name</code>.</p>

Table 7. Configuration File Elements (Continued)

Element	Description
multiVCsupported	Specifies whether to display (true) the vCenter Server selector or not (false). Only one multiVCsupported element allowed per configuration file.
description	Optional. Description of your plug-in. Only one per file allowed. This description displays in the Plug-in Manager of the vSphere Client.
name	Optional. Name of your plug-in that displays in the vSphere Client Plug-ins Manager.
vendor	Optional. Your company name.
extension	<p>File must contain at least one of these elements defining the extension point with which to associate the UI item. An extension element requires a <code>parent</code> attribute set to the appropriate extension point. Valid extension points are listed in Table 2, Table 3, Table 4, Table 5, Table 6, and Table 7.</p> <p>Optional attributes for extension include <code>privilege</code> and <code>treestyle</code>. Use the <code>privilege</code> attribute to associate one or more privilege requirements with the extension. For example, <code><extension parent="InventoryMenus.DVS" privilege="DVSwitch.Modify"></code>.</p> <p>You can use the <code>treestyle</code> attribute with <code>MainView.*</code> and <code>InventoryView.*</code> extension points to limit the positioning of the extension element to the left pane under the appropriate inventory item. The <code>treestyle</code> attribute supports the following options:</p> <ul style="list-style-type: none"> ■ Physical ■ Virtual ■ Datastores ■ Network <p>Each extension element must be followed by a <code>title</code> element and a <code>url</code> element. Each extension element may also contain <code>icon</code>, <code>iconSmaller</code>, and one or more <code>customAttribute</code> elements as appropriate for the extension point.</p> <p>Multiple extension elements can be nested, to create multi-level context menus. See Example 2, "Comprehensive Configuration File for Plug-In," on page 10.</p>
title	The title that displays in the vSphere Client view window when the extension is activated. Each title element has a <code>locale</code> attribute that must be set. For example, <code><title locale="en"></code> , <code><title locale="ja"></code> , or <code><title locale="de"></code> .
url	<p>Fully qualified Web server location of the script, Web application, icon, button, or other component for the plug-in. Required for each extension element defined in the file.</p> <p>Optional <code>display</code> attribute fine-tunes the action taken on the specified plug-in using one of the following settings:</p> <ul style="list-style-type: none"> ■ <code>window</code>—Open the URL in a new window inside the vSphere Client. ■ <code>modalwindow</code>—Not supported for scripting plug-ins. Opens new window that supports a wizard. ■ <code>none</code>—Execute the script located at the URL.
iconSmaller	A URL containing a 16 x 16 pixel image (.gif file) for display at a <code>ToolBar.*</code> or <code>InventoryView.*</code> extension point.
icon	A URL containing a 32 x 32 pixel image (.gif file) for display in a <code>HomeView.*</code> extension point.
customAttribute	Optional. Associates a menu or view with a custom attribute defined for the managed object on the server. Set the <code>name</code> attribute of a <code>customAttribute</code> to the name of a custom attribute defined on the managed object.

For example, to add a tab to the vSphere Client that displays in the right pane when users select a virtual machine in the left pane of the UI, define an extension in the configuration file that uses the following template:

```
...
<extension parent=InventoryView.VirtualMachine>
<title locale="en">Tab Title Here</title>
<url display="window">http://path/to/your/plugin/component/to/open/in/tab/pane</url>
</extension>
...
```

Example 1 shows a minimal configuration file that adds a small icon to different locations on the vSphere Client with a link to the VMware vSphere API Reference Guide on the VMware Web site. **Example 3, "Using the vSphere API to Register a vSphere Client Plug-in,"** on page 12 provides the Java client that registers the URL for the location of this file.

Example 1. Minimum Configuration File

```
<scriptConfiguration version="1.0">
<key>com.vmware.pubs.sdkteam</key>
<description>SDK Pubs Sample vSphere Client Plug-in</description>
<name>vSphere API Reference Documentation Plug-in</name>
<vendor>VMware, Inc.</vendor>
<multiVCsupported>>false</multiVCsupported>
<extension parent="HomeView.Admin">
<title locale="en">vSphere Web Services SDK API ReferenceGuide</title>
<url display="window">http://www.vmware.com/support/developer/vc-sdk/
visdk400pubs/ReferenceGuide/index.html</url>
<icon>http://www.vmware.com/support/developer/vc-sdk/visdk25pubs/docresources/page.gif</icon>
</extension>
<extension parent="Toolbars.Datacenter">
<title locale="en">vSphere Web Services SDK API ReferenceGuide</title>
<url display="window">http://www.vmware.com/support/developer/vc-sdk/
visdk400pubs/ReferenceGuide/index.html</url>
<iconSmaller>http://www.vmware.com/support/developer/vc-sdk/visdk25pubs/docresources/page.gif</ic
onSmaller>
</extension>
</scriptConfiguration>
```

To create the configuration file, you can use any text editor. However, VMware recommends using a XML editor, such as Altova XML Spy, to ensure you create a well-formed XML file. If the XML file is not formed correctly, the vSphere Client cannot load it. The Plug-In Manager of the vSphere Client displays error messages about the failure, but does not provide debugging information.

See <http://www.altova.com> for information about XML Spy.

Example 2 is a complete configuration file that defines several different extensions, including icons and menus, for demonstration purposes.

Example 2. Comprehensive Configuration File for Plug-In

```

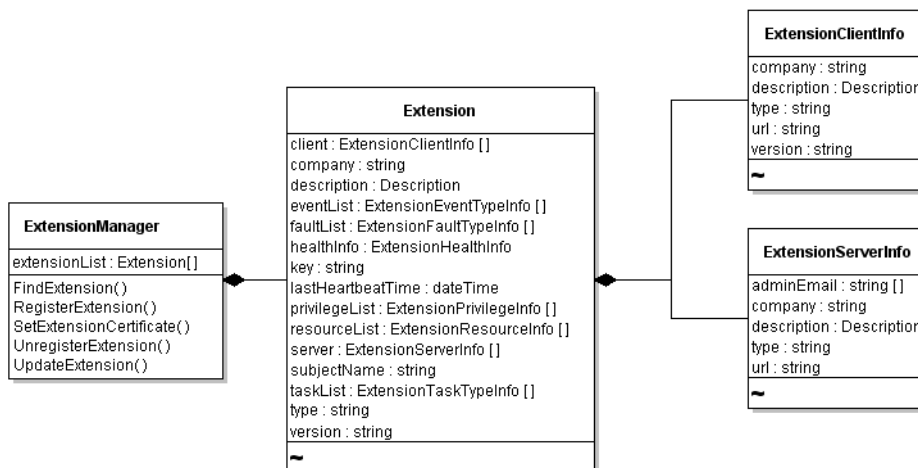
<?xml version="1.0" ?>
- <scriptConfiguration version="1.0">
  <key>com.vmware.test</key>
  <description>Test script plug-in</description>
  <name>vSphere Client 4.1 Extension Points example</name>
  <vendor>VMware, Inc.</vendor>
  <multiVCsupported>>false</multiVCsupported>
- <!-- Adds a shortcut to the home view -->
- <extension parent="Home.Inventory">
  <title locale="en">VMware</title>
  <url display="window">http://vmware.com</url>
  <icon>http://fqdn/path/to/plugin/component/scsi.png</icon>
  <iconSmaller>http://fqdn/path/to/plugin/component/scsi_small.png</iconSmaller>
</extension>
- <!-- Adds a tab to datacenter object -->
- <extension parent="InventoryView.Template">
  <title locale="en">Custom Template</title>
  <url display="modalwindow">http://fqdn/path/to/plugin/component/a.html</url>
</extension>
- <!-- Adds a menu to host system context menus.-->
- <extension parent="InventoryMenus.HostSystem">
  <title locale="en">Custom</title>
  <url>http://fqdn/path/to/plugin/component/a.html</url>
- <!-- You can associate an extension with custom attributes defined on the vCenter Server.-->
  <customAttribute name="A1">A1</customAttribute>
  <customAttribute name="A2">A2</customAttribute>
</extension>
-- <extension parent="InventoryMenus.HostSystem" privilege="Host.Config.Connection,
  Host.Config.Storage">
  <title locale="en">Test</title>
  <url>http://fqdn/path/to/plugin/component/a.html</url>
</extension>
- <extension parent="Toolbars.DatacenterFolder" treestyles="Physical, Virtual">
  <title>Plugin button</title>
  <url display="window">http://fqdn/path/to/plugin/component/a.html</url> -->
  <iconSmaller>http://fqdn/path/to/plugin/component/scsi_small.png</iconSmaller>
</extension>
- <!-- Adds a menu item to the DVS context menu. -->
- <extension parent="InventoryMenus.DistributedVirtualSwitch">
  <title>Plug-in menu item</title>
  <url>http://www.google.com/search?q=Distributed%20Virtual%20Switch</url>
</extension>
- <!-- nested menu -->
- <extension parent="InventoryMenus.Cluster">
  <title>Nested menu-item</title>
- <!-- The first URL is ignored -->
  <url>http://www.google.com/</url>
- <!-- Level 1 nesting -->
  <extension>
    <title>Level 1 - a</title>
  - <!-- Level 2 nesting -->
    <extension> <title>Level 2 - a</title> <url>http://www.google.com/</url> </extension>
  - <!-- Level 2 nesting -->
    <extension> <title>Level 2 - b</title><url>http://www.google.com/</url> </extension>
  </extension>
  - <!-- Level 1 nesting -->
    <extension> <title>Level 1 - b</title> <url>http://www.google.com/</url> </extension>
  - <!-- Level 1 nesting-->
    <extension> <title>Level 1 - c</title> <url>http://www.google.com/</url> </extension>
  </extension>
</scriptConfiguration>

```

Overview of the ExtensionManager Managed Object

ExtensionManager is the service interface for registering plug-ins. The ExtensionManager has a single property, extensionList, that is defined as an array of extensions registered with the vCenter Server. When you register your plug-in with the vCenter Server, it gets added to this array. The Extension data object contains metadata about each extension (plug-in), including its location and type, and other details.

Figure 4. ExtensionManager and Extension Data Object



Many Extension properties are optional. For example, the `subjectName` is used only if client certificates must be handled by the plug-in. See the VMware *vSphere API Reference Guide* for complete information about ExtensionManager and Extension. For information about the vSphere Client extensions supported in VI SDK 2.5, go to <http://www.vmware.com/support/developer/vc-sdk/vcplugin-exp/index.html>.

Registering the Extension

Your plug-in becomes available to end-users only after the configuration file and all other components have been located on the Web servers, and an Extension data object is registered with ExtensionManager. The Extension data object consists of metadata about the plug-in and the URL of its configuration file. When end-users connect to vCenter Server from vSphere Client, the plug-in components are dispatched to vSphere Client as described in “[Overview of vSphere Client Plug-Ins](#)” on page 1.

To register the extension with vCenter Server, you create a client utility application or script that creates the necessary Description, ExtensionServerInfo, and ExtensionClientInfo data objects that populate the Extension data object with values on the server. As an alternative, you can use the MOB to submit the same data objects in their XML form. Example 3 provides a complete Java application that registers the sample extension defined in Example 1. The sample extension consists of an icon for the vSphere API Reference Guide and a link to its location on the VMware Web site. This sample is supported on vCenter Server 4.0 and later.

Whether you use a script, a client application, or the MOB to register your plug-in, one of the most important details in Example 3 that is applicable to any extensions you create is that you must set the `type` property of the ExtensionServerInfo to the following specific package name:

```
com.vmware.vim.viClientScripts
```

Updating or Removing Plug-Ins

The `RegisterExtension` method takes an `Extension` data object that includes the URL of the Web site containing the configuration file. If you move the configuration file to a new location after registering its URL, you will need to update the location by using the `UpdateExtension` operation with the new location. To remove a vSphere Client Plug-in completely, use the `UnregisterExtension` operation.

[Example 3](#) shows a Java client that registers the URL for the location of the configuration file

Example 3. Using the vSphere API to Register a vSphere Client Plug-in

```
import com.vmware.vim25.*;
import java.net.URL;
import java.util.*;
public class PubsRegisterPlugin {
    private ManagedObjectReference _svcRef;
    private VimServiceLocator _locator;
    private VimPortType _service;
    private ServiceContent _sic;
    private String companyStr = "VMware, Inc.";
    private String descStr = "SDK Pubs documentation page";
    private String keyStr = "com.vmware.vSphereWebServicesSDK.API ReferenceGuide";
    private String ext_url = "http://www.vmware.com/support/developer/vc-sdk/pubsplugin.xml";
    private String adminEmail = "docfeedback@vmware.com";
    private String versionStr = "1.0";
    private void createServiceRef() throws Exception {
        _svcRef = new ManagedObjectReference();
        _svcRef.setType("ServiceInstance");
        _svcRef.setValue("ServiceInstance"); }
    private void connectLoginRegister(String hostName, String userName, String password) throws
        Exception {
        System.setProperty("axis.socketSecureFactory",
            "org.apache.axis.components.net.SunFakeTrustSocketFactory");
        String url = "https://" + hostName + "/sdk/vim";
        Description description = new Description();
            description.setLabel("vSphere Web Services SDK Main Documentation Page");
            description.setSummary(descStr);
        ExtensionServerInfo esi = new ExtensionServerInfo();
            esi.setUrl(ext_url);
            esi.setDescription(description);
            esi.setCompany(companyStr);
            esi.setType("com.vmware.vim.viClientScripts");
            esi.setAdminEmail(new String[] {adminEmail});
        ExtensionClientInfo eci = new ExtensionClientInfo();
            eci.setVersion(versionStr);
            eci.setDescription(description);
            eci.setCompany(companyStr);
            eci.setType("com.vmware.vim.viClientScripts");
            eci.setUrl(ext_url);
        Extension ext= new Extension();
            ext.setDescription(description);
            ext.setKey(keyStr);
            ext.setVersion(versionStr);
            ext.setSubjectName("blank");
            ext.setServer(new ExtensionServerInfo[] {esi});
            ext.setClient(new ExtensionClientInfo[] {eci});
            ext.setLastHeartbeatTime(Calendar.getInstance());
        createServiceRef();
        _locator = new VimServiceLocator();
        _locator.setMaintainSession(true);
        _service = _locator.getVimPort(new URL(url));
        _sic = _service.retrieveServiceContent(_svcRef);
        if (_sic.getSessionManager() != null) {
            _service.login(_sic.getSessionManager(), userName, password, null);
            ManagedObjectReference extMgrMof = _sic.getExtensionManager();
            _service.registerExtension(extMgrMof, ext);
        }
    }
}
```

```
public static void main(String [] args) throws Exception {
    PubsRegisterPlugin obj = new PubsRegisterPlugin();
    String serverName = args[0];
    String userName = args[1];
    String password = args[2];
    obj.connectLoginRegister(serverName, userName, password);
}
} //PubsRegisterPlugin class
```

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

Copyright © 2009, 2010 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.

Item: EN-000218-00
