# vSphere Web Services SDK Developer's Setup Guide

VMware vSphere 6.0

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-001412-01

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

# About This Book

This book, the *vSphere Web Services SDK Developer's Setup Guide,* provides information about setting up your development environment to use the VMware® vSphere Web Services SDK 6.0.

VMware provides several different APIs and SDKs for various applications and goals. This book provides information about using the vSphere Web Services SDK for developers who are interested in creating client applications for managing VMware® vSphere components available on VMware ESXi and VMware vCenter Server systems.

To view the current version of this book as well as all VMware API and SDK documentation, go to http://www.vmware.com/support/pubs/sdk_pubs.html

## Revision History

This guide is revised with each release of the product or when necessary. A revised version can contain minor or major changes. Table 1 summarizes the significant changes in each version of this guide.

**Table 1.** Revision History

| Revision | Description |
|---|---|
| 11Feb2016 | Correct the procedure for modifying the Web proxy service. |
| 12Mar2015 | vSphere 6.0 release. Changed destination directory for C# DLLs. |
| 20Feb2014 | Corrected instructions for modifying reverse proxy configuration. |
| 04Dec2013 | Clarified location of Web Services SDK. |
| 01Oct2013 | Revised C# setup instructions to reflect simplified procedure in 5.5 release and use new tools. Corrected path to Java sample. Updated Java version. Fixed minor typographical errors in code. |
| 09Sep2012 | Removed Axis support for 5.1 release. Made more corrections and expansions to C# instructions. |
| 17Nov2011 | Corrected setup instructions for C# stubs. |
| 24Aug2011 | Updated for vSphere 5.0 (included information on using JAX-WS bindings). |
| 13Jul2010 | Minor updates for vSphere 4.1 (new WSDL file configuration, example syntax). |
| 07May2009 | Revised release of the *vSphere Web Services SDK Developer's Setup Guide* for vSphere Web Services SDK 4.0. Server-certificate setup information is located in the reference section. Changed directory name for WSDLFILE environment variable. |
| 29Nov2007 | Initial release of *vSphere Web Services SDK Developer's Setup Guide* for VMware Infrastructure SDK 2.5. |

## Intended Audience

This book is intended for anyone who wants to develop applications using the VMware vSphere Web Services SDK. vSphere Web Services SDK developers typically include software developers creating client applications using Java or C# (in the Microsoft .NET environment) targeting VMware vSphere.

## VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation go to http://www.vmware.com/support/pubs.

## Document Feedback

VMware welcomes your suggestions for improving our documentation. Send your feedback to docfeedback@vmware.com.

# About the vSphere Web Services SDK 1

The VMware vSphere® Web Services SDK includes all the components necessary to work with the VMware vSphere API, including WSDL files, sample code, and libraries. The vSphere Web Services SDK facilitates development of client applications that target the VMware vSphere API. With the vSphere Web Services SDK, developers can create client applications to manage, monitor, and maintain VMware vSphere components, as deployed on VMware® VMware vSphere® ESX®, ESXi™, and VMware® vCenter™ Server systems.

This *vSphere Web Services SDK Programming Guide* explains how to set up the development environment to create new applications with Java and the Microsoft .NET environment, using the C# programming language. This guide also includes information about running the sample applications included with the vSphere Web Services SDK.

This chapter includes these topics:



## Knowledge Required for Using the vSphere Web Services SDK

Developing applications with the vSphere Web Services SDK requires expertise with Java, C#, or another programming language. You must also understand the following Web services programming concepts:

- Web services technology provides operations, also known as *methods* in the context of client applications. Using the vSphere Web Services SDK and your choice of programming language, you can create client applications that invoke these operations to perform the full range of server-side management and monitoring tasks.
- The Web services API is defined in Web Services Description Language (WSDL) files. The WSDL files are used by client-side Web-services utilities to create proxy code (stubs) that client applications use to interact with the server.
- Client applications invoke operations by calling proxy interface methods. The client proxy encodes an operation invocation into a SOAP message and sends it to the server. Simple Object Access Protocol (SOAP) is a programming-language neutral XML format. SOAP message translation is transparent to the developer.
- Communications between client and server occur over HTTP or HTTPS. HTTPS is a secure form of HTTP that uses SSL to encrypt client-server communications. The default is HTTPS, but you can configure the VMware vSphere Web server to support HTTP. (See "Modifying Server Configurations to Support HTTP" on page 27.)

You should also know about basic ESX, ESXi, and vCenter Server operations. See the VMware vSphere Documentation page on the VMware Web site.

## Programming Languages Supported by the vSphere Web Services SDK

Because the vSphere API is based on Web services, you can use any programming or scripting language that provides utilities for generating client-side stubs from Web-services WSDL files. The vSphere Web Services SDK package includes sample client applications developed in both Java and C#. SOAP toolkits are readily available for both languages.

See "vSphere Web Services SDK Package Contents" on page 9 for additional packaging details and for some caveats about the Java samples and for specific version requirements for the JDK, the Java API for XML Web Services libraries, and the JAX-WS libraries.

**Table 1-1.** Language and Tools Matrix for Client Application Development

| Language/Tool Context | Java | C# |
|---|---|---|
| Development environment or framework | J2SE 7, also known as J2SE 1.7. For best results, use J2SE 1.7 or higher. | Microsoft Visual Studio<br>Microsoft Visual C# |
| Web-services-client application development toolset, also known as a SOAP toolkit | JAX-WS 2.1 (Java API for XML Web Services). | Microsoft .NET Framework |

Developers, scripters, and administrators using Microsoft PowerShell or Perl can use the vSphere Web Services API through toolkits that VMware provides. For more information, see http://communities.vmware.com/community/developer.

## Types of Applications That You Can Build Using This SDK

You can use the vSphere Web Services SDK to develop system administration, provisioning, and monitoring applications for VMware vSphere systems.

The VMware vSphere Client application and VMware vSphere Web Access are two examples of client applications that were developed using vSphere API. The vSphere Client is a traditional Windows client application. Web Access is a browser plug-in that is available through the Web server port on ESX, ESXi, and vCenter Server systems.

With the vSphere Web Services SDK, you can create your own client applications that automate many administration, provisioning, or monitoring tasks associated with virtual infrastructure management and operations. The following examples are operational tasks that you can automate using the vSphere Web Services API:

- Create, configure, power cycle, or suspend virtual machines explicitly or by using profiles or templates to facilitate faster provisioning.
- Create, configure, and manage virtual devices, such as virtual CD-DVD drives, virtual network interface cards, virtual switches, and other components.
- Connect, power cycle, and disconnect ESX and ESXi host systems.
- Capture the state of a virtual machine to a snapshot and restore the state of a virtual machine from a snapshot, such as in a backup application.
- Gather statistics about host system and virtual machine performance.
- Manage events generated by the server, such as those created by alarms set for specific thresholds.
- Move virtual machines between hosts automatically.
- Manage load balancing and failover through the distributed resource scheduler (VMware DRS) and high availability (VMware HA) subsystems. VMware DRS and VMware HA require vCenter Server.

This list is not comprehensive. Also, some of the operations pertain to the service as a whole, not specific hosts or virtual machines. For example, load balancing can be a service-wide operation rather than a per-host or per-virtual machine operation.

# Downloading the vSphere Web Services SDK

The vSphere Web Services SDK, along with other VMware SDKs, is contained in the vSphere Management SDK. You can find the vSphere Management SDK at https://developercenter.vmware.com, which links to the My VMware service.

After you expand the vSphere Management SDK package, the Web Services SDK is in the subdirectory `SDK/vsphere-ws`. You also need the `SDK/ssoclient` subdirectory for client authentication.

## Configurations for Sample Authentication with Single Sign-On

The vSphere 6.0 release introduces a Platform Services Controller that can run on the same host as the vCenter Server, or can be configured on a separate host. At this time, the two possible configurations are:

- vCenter Server with an embedded Platform Services Controller (vCenter Single Sign-On Server, Lookup Service Server,...). This is like traditional vSphere deployments.
- vCenter Server with an external Platform Services Controller.

You will have to provide the vCenter Single Sign-On URL explicitly in order to run samples with the second configuration. With the first configuration, there is no need to provide the vCenter Single Sign-On URL (since the vCenter Single Sign-On service is embedded in the management node) and our SDK will continue to work as before.

The SDK samples have always had the option to explicitly specify the vCenter Single Sign-On URL (whether the vCenter Single Sign-On service is running inside or outside the management node). This is useful in cases where the vCenter Single Sign-On service is deployed outside the vSphere management node like the second configuration above.

When you log in to a vCenter Single Sign-On Server, you must be in a domain that has been added as a vCenter Single Sign-On identity source. If that domain is not the default domain, you must include the domain name as part of your user name, such as, administrator@vsphere.local. To learn more about configuring the vCenter Single Sign-On Server, see vSphere Security.

# vSphere Web Services SDK Package Contents

The vSphere Web Services SDK is a bundle that includes the following items:

- WSDL files that define the API available on a VMware vSphere server (ESX, ESXi, and vCenter Server) Web service.
- Precompiled client-side libraries (`vim.jar`, `vim25.jar`) available for test purposes that were generated from the WSDL. The vSphere Web Services API is packaged in the `vim25.jar` file and is available in the `SDK\vsphere-ws\wsdl\vim25` subdirectory.
- Sample code demonstrating common use cases associated with managing virtual infrastructure. The sample code includes compiled and ready-to-run Java class files and both Java and C# source code files. (For C# developers, the Microsoft Visual Studio project files (`.sln`) are included.)

  **NOTE** The precompiled Java samples (`samples.jar`) were compiled using JDK 1.7 from stubs generated by the Java API for XML Web Services (JAX-WS) libraries in J2SE 7.0, and work only with these specific versions of Java and JAX-WS. To use a different version of Java, or a different client-side Web services library, use the build script to rebuild the samples.

- Batch files and shell scripts (`build.bat` and `build.sh`) that automate the build process for Java and C# client applications.

- Batch files and shell scripts (`run.bat` and `run.sh`) that facilitate running the Java samples from the Windows command prompt.
- The *vSphere API Reference*, which provides language-neutral descriptive information about the VMware vSphere API and the object model, such as object type definitions, properties, and method signatures.

Complete information about setting up the environment, and about generating, compiling, and running applications is included in Chapter 2, "Setting Up for Java Development," on page 11, and in Chapter 3, "Setting Up for Microsoft C# Development," on page 17.

## SDK Versions and VMware vSphere Product Compatibility

VMware has released SDK products to support various versions of the VMware vSphere product family. You can use the VMware vSphere Web Services SDK 6.0 with many previous versions of VMware vSphere servers and its predecessor, VMware Infrastructure, including:

- ESX/ESXi 6.0, 5.5, 5.1, and 5.0
- ESX/ESXi 4.1 and 4.0
- ESX/ESXi 3.5 Update 5
- vCenter Server 6.0, 5.5, 5.1, 5.0, 4.1, 4.0
- VirtualCenter Server 2.5 Update 5

All versions are supported by using the appropriate WSDL files, as follows:

- `SDK\vsphere-ws\wsdl\vim25` contains WSDL files for use with ESXi 5, ESX/ESXi 4, vCenter Server 5, vCenter Server 4, ESX 3.5, and VirtualCenter 2.5 systems. As of vSphere 4.1, the vSphere API WSDL definitions are divided into several files. Backwards compatibility is achieved because both WSDL configurations (`vim25` and `vim` directories) use a top level file with the same name (`vimService.wsdl`).

The VMware vSphere API is a Web service that runs on VMware vSphere servers, including ESX, ESXi, and vCenter Server. The API exposed is the same in all products. However, vCenter Server provides the following capabilities which are not available through an ESX or ESXi Web service:

- Collecting historical performance data
- Optimizing resources, including managing distributed resources
- Enabling migration from one host system to another by using VMware vMotion
- Providing distributed resource management, including recovery, across all host systems under its control

If you attempt to invoke an operation on an ESX or ESXi system that is supported only on vCenter Server, the server returns a fault saying "not implemented" or "not supported." For example, the ExtensionManager API is available only on VirtualCenter Server 2.5 and subsequent releases of vCenter Server. Attempting to register an extension to an ESX system returns a fault, "not supported."

# Setting Up for Java Development 2

This chapter explains how to set up an environment to develop Java clients.

This chapter includes these topics:



## Java Development Requirements

Developing Java Web-services client applications using the VMware vSphere Web Services SDK requires the Java SDK and a Java Web services development toolset. For best results, use Java 2, Standard Edition, version 7.0 (J2SE 1.7.x), specifically JDK 1.7 or later.

The Java Web services development toolset must be a SOAP implementation that can be deployed to a Tomcat server. For example, you can use the client-side libraries in JAX-WS version 2.1. The JAX-WS 2.1 libraries are included with the JDK 1.7.

You can use other client-side tools and libraries, such as IBM WebSphere and several open source implementations, with the vSphere Web Services SDK. However, only the JAX-WS client libraries were tested with this guide.

The samples archive, in `samples.jar`, includes all vSphere Web Services SDK samples. The samples include client-side stub classes generated using the JAX-WS libraries. Samples using JAX-WS were generated using JDK 1.7.

**NOTE** If you are not using JDK 1.7, you must use the `build.bat` on Windows, or the `build.sh` on Linux, to generate stubs and compile the sample files (`vim25.jar` and `samples.jar`). The build scripts perform all necessary tasks for you, including setting the CLASSPATH and PATH environment variables. See "Generating Stubs and Compiling Classes" on page 14 for details.

## Set Up for Java Development

Specific setup instructions depend on whether your development workstation already meets some or all of the requirements, which client-side Web service library you plan to use, and whether you plan to use the provided samples. Specific setup instructions also depend on whether your target server uses the HTTPS protocol or HTTP.

### Software Downloads

You can obtain the software you need for Java client development from the following Web sites:

- The J2SE is available from http://www.oracle.com. For best results, use JDK 1.7 or later.

- Obtain the VMware vSphere Web Services SDK from https://developercenter.vmware.com. It is included in the vSphere Management SDK package.

## Set Up for Development Using JAX-WS

The samples generated using JAX-WS libraries and compiled using Java JDK 1.7 include `vim25.jar` and `samples.jar`. You can use these libraries without generating new stubs and recompiling if you are using the same version of the JDK.

The following instructions assume that the target server uses HTTPS, which is the default server configuration.

### To set up a development workstation to use Java and JAX-WS

1 If the JDK is not installed, create directories for the JDK and for the vSphere Web Services SDK package.

Do not use spaces in the directory names, to avoid issues with some of the included SDK batch and script files.

2 Install the Java 2 Platform, Standard Edition (J2SE) 6.0.

3 Unpack the components into subdirectories created in Step 1, using the provided installer if appropriate.

The J2SE uses an installation wizard. The SDK ZIP file unpacks into the directory you specify.

- Unpack with **Use folder names** selected, to maintain the organizational structure.
- On UNIX development systems, use the `unzip` command with the -a modifier, to ensure proper line-endings in the shell scripts. For example:

```
unzip –a VMware–vSphere–SDK–4.1.0–251329.zip
```

4 (optional) Import server-certificates and use the Java keytool utility to create a `vmware.keystore`. See "Import Server Certificates into the Java Keystore" on page 13 for details.

As an alternative, pass the `––ignorecert` argument at runtime to ignore server-certificate verification for any of the sample Java applications.

5 Create the JAVAHOME environment variable.

The JAVAHOME environment variable must be set to the root path of the Java Runtime Environment (JRE), such as `C:\Program Files\Java\jdk1.7.0_21`. The root directory of your Java installation contains `bin\javac` and other binary files needed to build the stubs and the samples.

6 If you are unable to use the `run.bat` script to run Java samples, add the precompiled sample files, `vim25.jar` and `samples.jar`, to your system CLASSPATH environment variable.

To test your setup, run the Java version of SimpleClient, as described in "Running the SimpleClient Sample Application to Validate Setup" on page 15.

## Batch Files and Shell Scripts for Building and Running Samples

The vSphere Web Services SDK includes several batch files for Windows and shell scripts for Linux that facilitate building and running the sample applications.

**NOTE** If you are using the JAX-WS 2.1 libraries with JDK 1.7, you do not need to rebuild the samples.

Some of the batch files are used by other batch files. For example, `build.bat` calls the `lcp.bat` and `clean.bat` scripts. If you modify the batch files for any reason, be aware of the dependencies among them.

**Table 2-1.** Batch Files and Shell Scripts for Java

| Filename | Description | Usage note |
|---|---|---|
| `build.bat`<br>`build.sh` | Checks for environment variable `JAVAHOME` and sets `PATH`, using the `JAVAHOME` variable. Cleans up existing Java files (by calling `clean.bat` or `clean.sh`). `build.bat` sets the local classpath (by calling `lcp.bat`). Creates the `vim25.jar`, and `samples.jar` files. | Use this script to generate client stubs and rebuild all sample applications. Use the -w flag to recompile without regenerating stubs. |
| `lcp.bat` | Sets the local classpath on the workstation. Called by `build.bat` and by `run.bat`. | Optional. Use to set local classpath. |
| `run.bat`<br>`run.sh` | Batch file that enables running any of the sample applications. Sets the Java trustStore property to the local trust store and invokes the Java runtime with the name of the application passed as a parameter. | Use this script to run any Java sample applications. |
| `clean.bat` | Removes any existing artifacts before building the samples, deleting Java class files in the samples packages and `samples.jar` file. Called by build script. | Optional. Deletes all generated source code files. |

## Import Server Certificates into the Java Keystore

Import server certificates if you plan to use the HTTPS protocol and if you do not plan to use the `--ignorecert` command-line argument.

To use HTTP, rather than HTTPS, and avoid the use of certificates entirely, follow the procedure detailed in "Modifying Server Configurations to Support HTTP" on page 27. However, using HTTPS provides better security for production environments.

The JAVAHOME environment variable must be set and added to the PATH environment variable. The certificate for each target server must be located in the `C:\VMware-Certs` subdirectory. See "Obtaining Server Certificates" on page 26.

### To import certificates into a local Java keystore

1 Open the Windows command prompt or Linux shell command.

2 Create the directory for the Java certificate store.

Create the directory only. The actual keystore file, `vmware.keystore`, is created during the process of importing the certificates.

| Operating System | Path |
|---|---|
| Windows | `C:\VMware-Certs\vmware.keystore` |
| Linux | `~/vmware-certs/vmware.keystore` |

3 Navigate to the directory.

For example, on Windows use the following directory:

```
cd vmware-certs\vmware
```

4 Use the Java `keytool` utility to import a certificate.

The syntax is as follows:

**`keytool -import -trustcacerts -alias` *`server-name`* `-file` *`certificate-filename`* `-keystore` *`keystore-name`***

For example:

```
C:\VMware-Certs>keytool -import -trustcacerts -alias root -file root.cer -keystore keystore.jks
```

A prompt requesting a password for the keystore appears:

```
Enter keystore password:
```

5 Create a password for the keystore by entering it at the prompt.

The keystore utility displays the certificate information at the console. For example:

```
Owner: OID.1.2.840.113549.1.9.2="1183400896,564d7761726520496e632e",
               CN=sdkpubslab-01.vmware.com, EMAILADDRESS=ssl-certificates@vmware.com,
               OU=VMware ESX Server Certificate, O="VMware, Inc.", L=Palo Alto,
               ST=California, C=US Issuer:
               OID.1.2.840.113549.1.9.2="1183400896,564d7761726520496e632e",
               CN=sdkpubslab-01.vmware.com, EMAILADDRESS=ssl-certificates@vmware.com,
               OU=VMware ESX Server Certificate, O="VMware, Inc.", L=Palo Alto,
               ST=California, C=US Serial number: 0 Valid from: Mon Jul 02 11:28:17 PDT 2007
               until: Mon Aug 31 11:28:17 PDT 2026
Certificate fingerprints:
MD5: . . .61:35:C0:C4
SHA1: 4C:...78:B2
```

At the end of the certificate information, a prompt displays a request for confirmation that the certificate is trusted:

```
Trust this certificate? [no]:
```

6 Type **yes** and press Enter to respond to the prompt and import the certificate into the `vmware.keystore` keystore.

The console displays this message:

```
Certificate was added to keystore
```

7 Repeat Step 4 through Step 6 for each target server.

# Generating Stubs and Compiling Classes

The vSphere Web Services SDK includes a set of Java archive files for the sample programs. The sample `.jar` files were created using JDK 1.7 and the JAX-WS Web services libraries.

## Precompiled JAX-WS Samples

The JAX-WS samples include the `vim25.jar` and `samples.jar` files that were created using the JAX-WS libraries included with JDK 1.7. These files are located in the `%WS_SDK_HOME%\java\JAX-WS\lib` directory.

If your development environment is using the JAX-WS libraries and JDK 1.7, you can use these precompiled libraries. Try running the SimpleClient by following the instructions in "Running the SimpleClient Sample Application to Validate Setup" on page 15.

## Use the Included Build Scripts

If your development environment uses different versions of the JDK or client-side libraries than the ones used for the precompiled sample files, you must regenerate the client-side stubs and recompile them to create Java archive files. The `build.bat` or `build.sh` script included with the SDK performs all necessary tasks for you.

You must regenerate the stubs if you are using the JAX-WS libraries with a Java version other than JDK 1.7. To use the included build scripts, the JAVAHOME environment variable must be set.

### To generate stubs and compile using the build.bat or build.sh script

1 Open a command prompt.

2 Navigate to the subdirectory containing the `build.bat` and `build.sh` files.

```
cd %WS_SDK_HOME%\java\JAX-WS\
```

3 Run the `build.bat` (or `build.sh`) script by entering its name at the command prompt.

```
build
```

The console displays output, starting with `Generating stubs from wsdl`. In a few minutes, the process finishes. The word `Done` appears at the command prompt, as shown in Example 2-1. The `Generating stubs from wsdl` message appears twice, because this build file generates client stubs using both sets of WSDL declarations, found in the `\vim` and `\vim25` subdirectories.

**Example 2-1.** Successful Stub Generation and Compilation Using the build.bat Script

```
Generating stubs from wsdl
Compiling stubs.
...
Done.
C:\devprojects\visdk21\SDK\vsphere-ws\java\JAX-WS>
```

When the process finishes, the appropriate sample `.jar` files show the current date and time.

To compile without re-generating the stubs from the WSDL, use the `-w` flag with the build script, as follows:

```
build -w
```

You can run any of the sample applications by following the instructions in "Running the SimpleClient Sample Application to Validate Setup."

# Running the SimpleClient Sample Application to Validate Setup

You can test your setup and connectivity by running one of the sample applications, such as SimpleClient. SimpleClient is a Java class that connects to the server and obtains a listing of the top-level inventory entities, their properties, and references. You can run any of the samples using the `run.bat` (or `run.sh`) script.

If you are using stubs generated by JAX-WS, these scripts require the JAVAHOME environment variable to be set.

## Run a Sample Application Using the Provided Scripts

You can use the `run.bat` or `run.sh` script to run any of the Java samples. The `SimpleClient` sample is a good choice to verify that your installation is correct. The path to the source file for SimpleClient is:

```
%WS_SDK_HOME%\java\JAXWS\samples\com\vmware\general\SimpleClient.java
```

When you run the script, specify the Java class for the sample application along with the `--url`, `--username`, and `--password` switches on the command line. Include the complete package name in the Java class specification. The following statement shows the general format for using the `run.bat` script to run the `SimpleClient` sample application from the Java samples subdirectory for JAX-WS:

```
run.bat com.vmware.samples.general.SimpleClient --url https://yourFQDNservername/sdk
              --username username --password password [--ignorecert ignorecert]
```

Example 2-2 shows sample output from the `SimpleClient` sample program.

**Example 2-2.** Sample Output of a Successful Run of SimpleClient, Using Precompiled Java Sample

```
Object Type : Folder
Reference Value : ha-folder-vm
   Property Name : name
   Property Value : vm
Object Type : HostSystem
Reference Value : ha-host
   Property Name : name
   Property Value : sdkpubslab-02.eng.vmware.com
Object Type : ResourcePool
Reference Value : ha-root-pool
   Property Name : name
   Property Value : Resources
Object Type : Folder
Reference Value : ha-folder-host
   Property Name : name
   Property Value : host
Object Type : ComputeResource
Reference Value : ha-compute-res
   Property Name : name
   Property Value : sdkpubslab-02.eng.vmware.com
Object Type : VirtualMachine
Reference Value : 16
   Property Name : name
   Property Value : Windows_2K3_VM
...
Object Type : Datacenter
Reference Value : ha-datacenter
   Property Name : name
   Property Value : ha-datacenter
Object Type : Folder
Reference Value : ha-folder-root
   Property Name : name
   Property Value : ha-folder-root
```

### To run the precompiled SimpleClient from the command prompt

1 Open a Windows command prompt or shell prompt on Linux.

2 Navigate to the Java samples subdirectory.

```
cd %WS_SDK_HOME%\java\JAXWS\samples
```

3 Invoke the Java runtime, providing the full package name of the `SimpleClient`, server URN, credentials, and Java `keyStore` location, or the `--ignorecert` argument. The complete syntax is as follows:

```
java -Djavax.net.ssl.trustStore=keystore-path-or-%KEYSTORE%-environment-variable
               package-hierarchy-classname --url server-url --username username
               --password password [--ignorecert ignorecert]
```

For example:

```
java -Djavax.net.ssl.trustStore=%VMKEYSTORE% com.vmware.general.SimpleClient
               --url https://example.com/sdk --username pubs --password ***
               --ignorecert ignorecert
```

**NOTE** If error messages occur due to system heap or other memory problems, you can give the Java VM more memory, as follows:

```
java -Djavax.net.ssl.trustStore=%VMKEYSTORE% -Xms512M -Xmx1024M
               com.vmware.general.SimpleClient https://sdkpubslab-02.eng.vmware.com/sdk
               --username username
               --password password --ignorecert ignorecert
```

# Setting Up for Microsoft C# Development

3

This chapter explains how to set up an environment to develop C# clients for the vSphere Web Services SDK.

This chapter includes these topics:



## C# Development Requirements

The vSphere Web Services SDK includes C# (`.cs`) source files and Microsoft Visual Studio project files (solutions, or `.sln`) for Microsoft Visual Studio. In addition, Web services client application development for C# requires:

- Development environment for C#, such as Microsoft Visual C# or Microsoft Visual Studio.
- Microsoft .NET Framework, which is included with Microsoft Visual Studio.
- Microsoft Web Services Enhancements, a runtime and tools package.

### Software Downloads for C# Development

You can obtain the VMware vSphere Web Services SDK from https://developercenter.vmware.com. It is included in the vSphere Management SDK package.

## Setting Up for C# Development

These instructions show how to install all the required software. If your development workstation already meets some or all of the requirements, you generally do not need to re-install the software you already have.

### Set Up a Development Workstation To Use C#

For general Web development work, you need a C# development environment and the .NET Framework. To work with the VMware vSphere Web Services API, you need the vSphere Web Services SDK and additional tools available from Microsoft.

### To set up a development workstation to use C#

1 Install the Microsoft Visual programming environment, such as Microsoft Visual C# or Microsoft Visual Studio.

Use Microsoft Visual Studio 2008 or later, which includes the required .NET Framework.

2 Obtain the Microsoft .NET Framework, if it is not included in the Microsoft Visual programming environment.

Use .NET version 3.5 or later, depending on your Visual Studio version.

3 Download and install the Microsoft .NET Framework 2.0 SDK (x64) from http://www.microsoft.com/en-us/download/details.aspx?id=15354

**NOTE** Virtual Studio includes a version of .NET version 2.0, but that version does not contain the tools needed to build the DLLs.

4 Find the location where the Microsoft .NET Framework 2.0 SDK (x64) files were installed. Verify that the installation directory contains a subdirectory named `Bin`, which contains a file named `wsdl.exe`. Typically, the Framework SDK is installed at `C:\Program Files\Microsoft.NET\SDK\v2.0 64bit`.

5 Edit your Windows registry to identify the location where the Microsoft .NET Framework 2.0 SDK (x64) was installed.

a Under the key `HKLM\SOFTWARE\Microsoft\.NETFramework`, verify the existence of a string value named `sdkinstallRootv2.0` with a string value of the full path name to the installation directory.

b If `sdkinstallRootv2.0` is not present, add it.

6 Download and install Microsoft Web Services Enhancements (WSE) 3.0 from http://www.microsoft.com/en-us/download/details.aspx?id=14089.

**NOTE** The default option for the WSE installer includes only the runtime, not the WSDL tool. Select another option to include the Tools directory.

7 Download and unzip the VMware vSphere Web Services SDK package from the VMware Web site at https://developercenter.vmware.com.

## Set Environment Variables To Build C# DLLs and Samples

Before you build the SDK sample programs, you must set your environment variables.

If your Microsoft software setup varies from the default paths, create a `VSINSTALLDIR` environment variable.

**Table 3-1.** Environment Variable Names Used To Build Samples

| Variable Name | Description | Usage Note |
|---|---|---|
| VSINSTALLDIR | Location of Microsoft ...\Common7 and ...\SDK subdirectories. | Required only if your development setup varies from Microsoft default installation paths.<br>Use double quotation marks around directory path names that include spaces. For example:<br>"C:\devstuff\Microsoft Visual Studio 8\Common7"<br>"C:\devstuff\Microsoft Visual Studio 8\SDK" |
| WSE_HOME | Location of WSE. | Directory where you installed WSE. |
| WS_SDK_HOME | Location of the WS SDK files. | Directory containing unzipped WS SDK files (usually *Location of zip file*\SDK\vsphere-ws). |
| WSDLHOME | Location of the WSDL files. | Directory containing unzipped WSDL files (usually *Location of zip file*\SDK\vsphere-ws\wsdl). |

#### To set environment variables

1 Open a Visual Studio command shell window.

2 Create the WSE_HOME environment variable, setting its value to the absolute path to the directory where WSE was installed. For example:

```
set WSE_HOME="C:\Program Files (x86)\Microsoft WSE\v3.0"
```

3 Add the WSE tools directory to the PATH environment variable:

```
set PATH=%PATH%;%WSE_HOME%\Tools
```

4 Create the WS_SDK_HOME environment variable, setting its value to the absolute path to the directory containing all the SDK files extracted from the zip file you downloaded. For example:

```
set WS_SDK_HOME="C:\Documents and Settings\yourusername\My Documents\Downloads\SDK"
```

5 Create the WSDLHOME environment variable, setting its value to the absolute path to the directory where the WSDL files were stored when you uncompressed the SDK download file. For example:

```
set WSDLHOME=%WS_SDK_HOME%\vsphere-ws\wsdl\vim25
```

6 If your Microsoft development and .NET software was not installed using default paths, create and set the VSINSTALLDIR environment variable to contain the location where the software was installed. For example:

```
set VSINSTALLDIR="C:\devstuff\Microsoft Visual Studio 8\SDK"
```

# Building the C# SSO DLL

In the vSphere Web Services SDK, VMware supplies several sample Single Sign-On (SSO) clients for Visual Studio 2008. The SDK includes a project (`.csproj`) file for each sample, and a solution (`.sln`) file for the whole set of samples. The project files reference the DLL through which a client communicates with the SSO service.

The samples demonstrate how to authenticate with the SSO service, using the methods available to vSphere clients. The methods include the use of user credentials to authenticate and retrieve bearer and Holder-of-Key (HoK) tokens. You can use these methods to authenticate when running vSphere samples or other clients, such as the `SimpleClient` sample described in "Running the Microsoft .NET C# Version of SimpleClient" on page 23.

## Build the C# SSO DLL

The samples in the SDK contain code to authenticate with an SSO server. Before you run the SDK samples, you must build the SSO DLL.

### Prerequisites for Building the C# SSO DLL

- install a C# development environment, as described in "Setting Up for C# Development" on page 17.
- Open a Visual Studio command shell window and set environment variables as described in "Set Environment Variables To Build C# DLLs and Samples" on page 18.

#### To Build the C# SSO DLL

1 Open a Visual Studio Tools command shell.

2 Navigate to the .NET subdirectory for SSO client samples.

```
cd %WS_SDK_HOME%\ssoclient\dotnet\cs\samples
```

3 Generate a test certificate and STSService stubs using the build.bat script.

```
.\build.bat
```

4 Copy the SSO DLL to the `%SDK_HOME%\vsphere-ws\dotnet\cs\samples\lib` directory for use by the vSphere sample clients.

```
copy lib\STSService.dll %WS_SDK_HOME%\vsphere-ws\dotnet\cs\samples\lib\.
```

# Building the C# vSphere DLLs

In the vSphere Web Services SDK, VMware supplies sample vSphere clients for Visual Studio 2008. The SDK includes a project (.csproj) file for each sample, and a solution (.sln) file for the whole set of samples. The project files reference the DLLs through which a client communicates with the Web service.

## XML Serializers

For best performance, precompile the XML serializers to a separate DLL. If you precompile the XML serializers and modify the class declaration to use the precompiled serializer DLL, the SDK samples require a shorter initialization time when they instantiate the VimService class.

See the MSDN documentation for more details about XML serialization and startup performance.

## Build the C# vSphere DLLs

Before you can run the SDK samples, you must build the DLLs that provide common services to the samples.

### Prerequisites for Building the C# vSphere DLLs

- Install a C# development environment, as described in "Setting Up for C# Development" on page 17.
- Open a Visual Studio command shell window and set environment variables as described in "Set Environment Variables To Build C# DLLs and Samples" on page 18.

For an example of how to script this procedure using a Windows batch file, see "Scripting the C# DLL Build" on page 31.

#### To build the C# vSphere DLLs

1 Open a Visual Studio Tools command shell.

2 Navigate to the .NET subdirectory for vSphere client samples.

```
cd %WS_SDK_HOME%\vsphere-ws\dotnet\cs\samples
```

3 Generate the VimService.cs file from the WSDL, using the following command syntax with the WSE WSDL tool:

```
wsewsdl3.exe /n:Vim25Api /type:webClient /l:CS %WSDLHOME%\vim.wsdl %WSDLHOME%\vimService.wsdl
```

This command generates VimService.cs, the default output file, in the current directory, using the Vim25Api namespace.

4 Compile VimService.cs to a library, using the following command syntax:

```
csc /t:library /out:Vim25Service.dll /r:"%WSE_HOME%\Microsoft.Web.Services3.dll" VimService.cs
```

This command generates a serializer assembly, a DLL.

5 Use the sgen tool to pregenerate and compile the XML serializers, using the following command syntax:

```
sgen /p Vim25Service.dll
```

This command outputs the Vim25Service.XmlSerializers.dll file in the current directory. This DLL file contains pregenerated XML serializer code.

6 Using a source code editor, find occurrences of the following string in the VimService.cs file that you generated in Step 3.

```
[System.Xml.Serialization.XmlIncludeAttribute
```

Replace occurrences of the string with

```
// [System.Xml.Serialization.XmlIncludeAttribute
```

This will prevent .NET from processing the Xml.Serialization.XmlIncludeAttribute attributes that are the main cause of the slow instantiation of the Vim25Service class.

7 Annotate the `VimService` class in the `VimService.cs` file that you generated in Step 3, adding this `XmlSerializerAssemblyAttribute` to point to the location of the XML serializer assembly:

```
[System.Xml.Serialization.XmlSerializerAssemblyAttribute(AssemblyName =
               "Vim25Service.XmlSerializers")]
```

The result should look something like the following example:

```
// ... Some code here ...
[System.Xml.Serialization.XmlSerializerAssemblyAttribute(AssemblyName =
               "Vim25Service.XmlSerializers")]
public partial class VimService : Microsoft.Web.Services3.WebServicesClientProtocol {
// ... More code here.
```

8 Save the modified `VimService.cs` file.

9 Regenerate the `Vim25Service.dll` library with the following command syntax:

```
csc /t:library /out:Vim25Service.dll /r:"%WSE_HOME%\Microsoft.Web.Services3.dll" VimService.cs
```

10 Copy the generated files `Vim25Service.dll` and `Vim25Service.XmlSerializers.dll` to the `%SDK_HOME%\vsphere-ws\dotnet\cs\samples\lib` directory.

```
copy Vim25Service*.dll lib
```

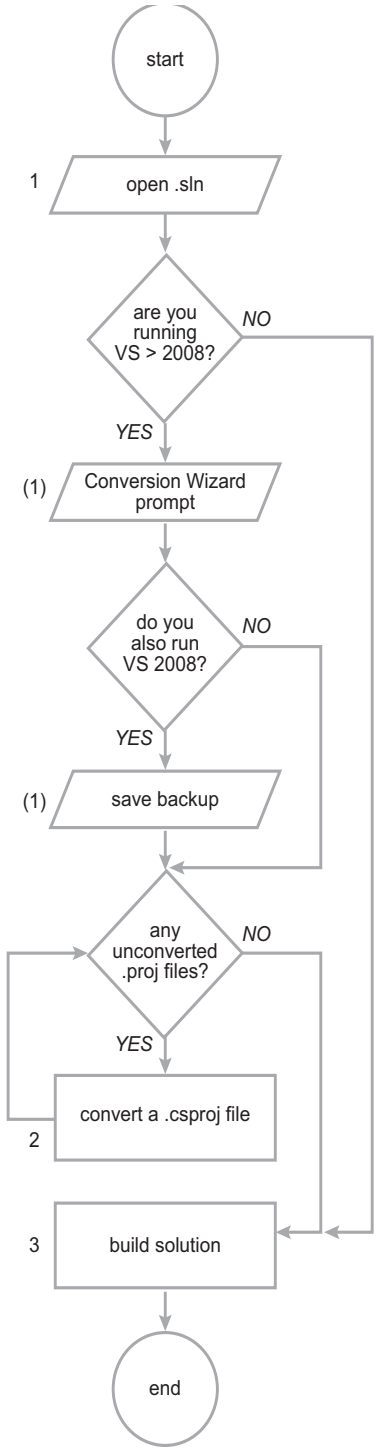
# Building the C# Sample Programs

The vSphere Web Services SDK contains sample clients that demonstrate how to perform functions to manage your datacenter. The samples are organized into individual Visual Studio projects, which are collected in a single solution file. This document explains how to build the entire solution set.

## Build the C# Sample Programs

The C# sample clients demonstrate functions that you can use to manage your datacenter. The build process varies, depending on the version of Visual Studio. The process is illustrated in Figure 3-1 and described in "To build the C# sample programs" on page 23.

**Figure 3-1.** Build Process for C# Sample Programs




## Prerequisites for Building the C# Sample Programs

Before you build a sample program, you must build the relevant DLLs:

- `STSService.dll`
- `Vim25Service.dll`
- `Vim25Service.XmlSerializers.dll`

See "Build the C# vSphere DLLs" on page 20 and "Build the C# SSO DLL" on page 19 for information about building the DLLs.

#### To build the C# sample programs

1 Launch Visual Studio and load the solution file, `Samples2008.sln`.

The solution file is found in the `%SDK_HOME%\vsphere-ws\dotnet\cs\samples` directory.

If you are using a version of Visual Studio later than 2008, the Visual Studio Conversion Wizard prompts you to convert the 2008 solution file to the newer version.

- If you also have Visual Studio 2008, select **Yes** when the wizard prompts you to save a backup of the original solution file.
- If you do not have Visual Studio 2008, you do not need to save a backup of the original solution file.

2 If you are using a version of Visual Studio later than 2008, convert each project to use .NET Framework 4.

a To convert a project to use .NET Framework 4, right-click its name in the Solution Explorer and select **Properties**.

b In the Properties panel, change the **Target Framework** from **.NET Framework 3.5** to **.NET Framework 4**.

3 From the Visual Studio menu, select **Build > Build Solution**.

All the sample programs build. The Output pane at the bottom of the Visual Studio window shows build errors, if any.

4 Correct any errors in the build, and repeat the build.

# Running the Microsoft .NET C# Version of SimpleClient

The SimpleClient sample application connects to a vSphere host, lists the names and reference IDs of the managed objects in the inventory, and disconnects from the host.

## Run the SimpleClient C# application

You can use the SimpleClient sample application to test your setup and connectivity.

### Prerequisites To Run the SimpleClient C# Application

Verify that the following conditions exist:

- Your development environment is set up. See "Setting Up for C# Development" on page 17.
- The SSO DLL is built. See "Building the C# SSO DLL" on page 19.
- The vSphere DLLs are built. See "Building the C# vSphere DLLs" on page 20.

#### To run the SimpleClient application

1 Open a Visual Studio Tools command shell.

2 Navigate to the subdirectory where the compiled object code is located.

From the top-level directory of the SDK download, the directory is as follows:

```
%WS_SDK_HOME%\dotnet\cs\samples\SimpleClient\bin\Debug
```

3 Run the application, passing at least the service URL, an authenticating user name, and a password on the command line.

For a development environment, you do not need to use SSO or certificate authentication. You can authenticate directly with the server by specifying `--disablesso` and `--ignorecert` on the command line. For example:

```
simpleclient --url https://esx.exampledomain.com/sdk --username root --password secret
             --disablesso --ignorecert
```

The application connects to the server and displays a list of inventory objects managed by the server.

**Example 3-1.** Sample Output of a Successful Run of SimpleClient

```
Object Type : Datacenter
Reference Value : ha-datacenter
   Property Name : name
   Property Value : ha-datacenter
Object Type : Folder
Reference Value : ha-folder-root
   Property Name : name
   Property Value : ha-folder-root
```

# Troubleshooting the Setup

If you cannot successfully run the SimpleClient, first check your environment settings and all other setup tasks.

## Proxy Server Connection Problem

You might have a proxy connection problem.

### Problem

The sample application reports that it is unable to connect to the remote server. For example:

```
SimpleClient.exe https://<management-server>/sdk <user> <pass>
Caught Exception : Name : WebException Message : Unable to connect to the remote server
Trace : at System.Net.HttpWebRequest.GetRequestStream() at
System.Web.Services.Protocols.SoapHttpClientProtocol.Invoke(String methodName, Object[]
parameters) at VimApi.VimService.RetrieveServiceContent(ManagedObjectReference _this) ...
Exception disconnecting.
Caught Exception : Name : NullReferenceException Message : Object reference not set to an instance
of an object.
Trace: ...
```

### Cause

Cannot connect to the Web service from Microsoft .NET client sample through the proxy server.

### Solution

Try a different server on the same subnet as the client.

## `sgen` Configuration Issues

The `sgen` tool used to generate XML serializer assemblies might be configured incorrectly.

### Problem

The `sgen` tool gives unexpected results.

### Cause

The behavior of the `sgen` tool varies depending on what you have in your `Machine.config` file. For example, by default `sgen` is supposed to output optimized non-debug code, but that is not always the case.

### Solution

Adjust your `sgen` configuration. To get more debugging information, use the `/k` flag, which causes `sgen` to keep all its temporary generated files, including the source files and command-line option files

# vSphere Server Certificates

A

The VMware vSphere API is available as a secure Web service. Secure Web service means that, by default, ESX, ESXi, and vCenter Server are configured for HTTPS and support SSL to encrypt communications. This appendix explains how to manage the certificates needed for secure communications.

This appendix includes these topics:



## Secure Client-Server Communications

To connect to the server using HTTPS, client applications must verify the identify of the server by using the server's certificate during an initial handshake. The client must obtain the server certificate in advance, so that it is available during the handshake. See "Obtaining Server Certificates" on page 26.

To connect to the server using HTTP requires that you first modify the target server's default configuration so that it supports regular HTTP communications. If you configure the server for HTTP, you do not need to import the server certificates on the client development workstation. See "Modifying Server Configurations to Support HTTP" on page 27. Modifying the server configuration to support HTTP access to the vSphere API is recommended for test or development environments only, not for production deployments. The default protocol, HTTPS, provides better security for production deployments.

## Simplified Security Setup for Development Environment

You can bypass certificate checking while developing software in a non-production environment. To do this, create a custom implementation of the `javax.net.ssl.TrustManager` interface that returns `true` rather than actually verifying certificates during the SSL handshake. You can see examples of such a class in the Java code samples included with the vSphere Web Services SDK.

The Java samples included with the SDK use this technique by accepting an optional command-line argument, `--ignorecert`. If you plan to use the `--ignorecert` option or use this automatic server-certificate verification technique in your own code, you do not need to import certificates. See "Set Up for Java Development" on page 11 for more information.

Use the `--ignorecert` option only for development and testing purposes. Do not use it outside a firewall. If the server-certificate is not verified during the SSL handshake, the client application is subject to man-in-the-middle attacks.

# Obtaining Server Certificates

VMware products use standard X.509 version 3 (X.509v3) certificates to encrypt session information sent over SSL connections between server and client systems. When a client application initiates an SSL session with the server, the server sends its certificate to the client application, which checks the X.509 certificate against a list of known Certificate Authorities (CAs) to verify the authenticity of the certificate. The client then uses the server's public key contained in the X.509 certificate to generate a random symmetric key, which it uses to encrypt all subsequent communications.

The installers for ESX, ESXi, and vCenter Server create server certificates during the process of installation. For ESX and ESXi systems, the certificate name matches the DNS name of the server. For vCenter Server systems, the certificate name is VMware. Because these certificates are not signed by an official root CA, you must obtain the server certificate from each server that you plan to target with your client application and store it locally.

For example, if you are creating a client application to run against the vCenter Server and an ESX system in standalone mode, you must obtain both the vCenter Server certificate and the ESX certificate. If your application is aimed solely at the vCenter Server that might manage any number of ESX systems, you must obtain the certificate only from the vCenter Server.

You can obtain the certificates in one of the following ways:

- Developers working on the Microsoft Windows platform can use the certificate-handling capabilities of the vSphere Client from the development workstation to connect to each ESX, ESXi, or vCenter Server and accept the certificate into the local cache and export the certificate. See "Obtain Certificates Using the vSphere Web Client" on page 26.
- Developers with access privileges on the target server systems can use a secure shell client utility (SCP, WinSCP, or SSH) to connect directly to the ESX, ESXi, or vCenter Server and copy the certificates directly from the server to the development platform.

## Obtain Certificates Using the vSphere Web Client

Use the vSphere Web Client to obtain certificates, so you don't have to install another client on your development workstation. You can download the VMware Certificate Authority root and leaf certificates and then add them to the operating system root store of the system from which you are connecting to the vCenter Server system.

**To obtain server certificates:**

1. From a client system Web browser, go to the URL of the vCenter Server system or the vCenter Server Virtual Appliance.
2. Click the **Download trusted root CA certificates** link at the bottom of the grey box on the right and download the file.
3. Change the extension of the file to **`.zip`**.
4. The file is a ZIP file of all root certificates and all CRLs in the VMware Endpoint Certificate Store (VECS).
5. Extract the contents of the ZIP file.
6. The result is a `.certs` folder that contains two types of files. Files with a number as the extension (`.0`, `.1`, and so on) are root certificates. Files with an extension that starts with an `r` (`.r0`, `.r1`, and so on) are CRL files associated with a certificate.
7. Install the certificate files as trusted certificates by following the process that is appropriate for your operating system.

Firefox has its own trusted roots store and does not use the operating system store. If you are working with Firefox, download the certificate as described above, and then select **Tools > Options**, click **Advanced**, and click **Certificates** to import the certificate into Firefox.

After you obtain the certificate from each target server, follow the other setup steps appropriate for your programing language. For C# developers, see "Setting Up for C# Development" on page 17. For Java developers, see "Set Up for Java Development" on page 11.

For the latest information about certificates, see the vSphere Security guide at http://www.vmware.com/support/pubs/.

## Updating the Active Directory Group Policy to Accept Certificates

If you have a configuration where the VMware Certificate Authority is an intermediate Certificate Authority, a Custom Certificate, or another certificate that is not trusted in your environment, and:

- you have a Web browser that uses the operating certificate store on Windows (such as Internet Explorer and Google Chrome)
- you can access the vCenter Server from several different machines

you can import the root certificate into the group policy of your Active Directory environment to make the certificates trusted in your Active Directory domain.

### To import the root certificate

1. Go to the URL of the vCenter Server system or the vCenter Server Virtual Appliance using a client system web browser.
2. Click the **Download trusted root CA certificates** link at the bottom of the grey box on the right and download the file.
3. Change the extension of the file to **`.zip`**.
4. The file is a ZIP file of all root certificates and all CRLs in the VMware Endpoint Certificate Store (VECS)
5. Extract the ZIP file.
6. The result is a **`.certs`** folder that contains two types of files. Files with a number extension (`.0, .1,` and so on) are root certificates. Files with a extension that starts with an `r` (`.r0,. r1,` and so on) are CRL files associated with a certificate.
7. Open the **Active Directory Group Policy Management Editor**.
8. Open **Public Key Policies** and select **Intermediate Certification Authorities**.
9. Add the certificate file or files that you downloaded.
10. From your Windows command prompt, run **`gpupdate /force`** to force an update.

Firefox has its own trusted roots store and does not use the operating system store. If you are working with Firefox, download the certificate as described above, and then select Tools > Options, click Advanced, and click Certificates to import the certificate into Firefox.

# Modifying Server Configurations to Support HTTP

ESX, ESXi, and vCenter Server support the vSphere API through their respective Web services (SOAP) engines. By default, these Web services run on port 443, as secure Web services that can be accessed using SSL over HTTP (HTTPS). However, for a development environment, you might want to simplify the connection process from a client application by configuring the target servers to support HTTP.

Connections to the Web services port are handled by a reverse-proxy service. The reverse-proxy service handles requests to the API (through the `/sdk` path) and to the Managed Object Browser (through the `/mob` path). The reverse-proxy service has a configuration file that can be modified to specify support for HTTP as an accepted protocol for the Web service.

The procedure to modify the reverse proxy configuration differs, depending on the server type and the release version. Choose one of the following options that applies to your situation:

- "HTTP Configuration for ESXi 5.1, 5.5, or 6.0" on page 28

- "HTTP Configuration for ESX 4.1, ESXi 4.1, or ESXi 5.0" on page 29
- "HTTP Configuration for vCenter Server" on page 30


## HTTP Configuration for ESXi 5.1, 5.5, or 6.0

You can modify ESXi configuration from a shell window over an SSH connection, using the following procedure. If you do not have SSH enabled, use the appropriate vSphere CLI command to obtain the configuration file from the server, modify the file to support HTTP, and move the file back to the ESXi system. For more information about the vSphere CLI command syntax, see the *vSphere CLI Installation and Reference Guide*.

**To modify the Web proxy service on ESXi 5.1, 5.5, or 6.0 to support HTTP**

1 Log in to a shell window as the root user.

2 Change directories to `/etc/vmware/rhttpproxy`.

```
# cd /etc/vmware/rhttpproxy
```

3 Copy the `endpoints.conf` file to a temporary directory for editing.

```
# cp endpoints.conf /tmp/endpoints.conf
```

4 **Change the permissions on the `endpoints.conf` file to allow editing.**

```
# chmod +w /tmp/endpoints.conf
```

5 Use a text editor to open the temporary file.

```
# vi /tmp/endpoints.conf
```

6 Navigate to the line that specifies the endpoints for SDK connections, which begins with `/sdk`.

The line looks similar to this:

```
/sdk    local    8307              redirect     allow
```

If the `/sdk` line ends with the words `allow allow`, you do not need to change it. HTTP access is already allowed.

7 To enable HTTP connections, change the word `redirect` to `allow`.

When configured to allow both HTTP and HTTPS connections, the `/sdk` line looks like this:

```
/sdk    local    8307              allow     allow
```

8 (Optional) If you prefer to completely disable HTTPS, change the last word to `reject` instead of `allow`.

When configured to allow only HTTP connections, the `/sdk` line looks like this:

```
/sdk    local    8307              allow     reject
```

9 (Optional) Change the setting for the Managed Object Browser as well.

When configured to allow both HTTP and HTTPS connections, the `/mob` line looks like this:

```
/sdk    local    8307              allow     allow
```

10 Save your settings and close the file.

11 Copy the original `endpoints.conf` file to a backup file.

```
# cp endpoints.conf endpoints.conf.old
```

12 Change the permissions on the `endpoints.conf` file to disable editing.

```
# chmod -w /tmp/endpoints.conf
```

13 Copy the temporary file `endpoints.conf` file back, replacing the original `endpoints.conf` file.

```
# cp /tmp/endpoints.conf endpoints.conf
```

14 Signal the reverse proxy service to update its configuration by entering the following command:

```
/etc/init.d/rhttpproxy restart
```

**Example A-1.** An `endpoints.conf` File Modified To Support HTTP connections to the SDK and the MOB

```
/                   local           8309       redirect       allow
/sdk                local           8307       allow          allow
/client/clients.xml local           8309       allow          allow
/ui                 local           8308       redirect       allow
/vpxa               local           8089       reject         allow
/mob                namedpipe       /var/run/vmware/proxy-mob        allow          allow
/wsman              local           8889       redirect       allow
/sdkTunnel          namedpipetunnel /var/run/vmware/proxy-sdk-tunnel allow          reject
/ha-nfc             local           12001      allow          allow
/nfc                local           12000      allow          allow
/folder             local           8309       redirect       allow
/host               local           8309       redirect       allow
/tmp                local           8309       redirect       allow
/screen             local           8309       redirect       allow
/guestFile          local           8309       redirect       allow
/cgi-bin            local           8309       redirect       allow
```

## HTTP Configuration for ESX 4.1, ESXi 4.1, or ESXi 5.0

You can modify ESX configuration from the service console, using the following procedure. The procedure is the same for ESXi, but you must use a shell window over an SSH connection because ESXi does not have a service console.

If you do not have SSH enabled for your ESXi server, use the appropriate vSphere CLI command to obtain the configuration file from the server, modify the file to support HTTP, and move the file back to the ESXi system. For more information about the vSphere CLI command syntax, see the *vSphere CLI Installation and Reference Guide*.

### To modify the Web proxy service on ESX or ESXi 4.1 or 5.0 to support HTTP

1 Log in to the service console or a shell window as the root user.

2 Change directories to `/etc/vmware/hostd`.

3 Use a text editor to open the `proxy.xml` file.

4 Navigate to the list of endpoints in the file, identified by the `<EndpointList>` tag, that contains settings for the Web service supporting the SDK.

The nested tags might look something like this:

```
...
<e id="1">
    <_type>vim.ProxyService.NamedPipeServiceSpec</_type>
    <accessMode>httpsWithRedirect</accessMode>
    <pipeName>/var/run/vmware/proxy-sdk</pipeName>
    <serverNamespace>/sdk</serverNamespace>
</e>
...
```

5 Change the `accessMode` to `httpAndHttps`.

If you prefer to completely disable HTTPS, set the `accessMode` to `httpOnly`.

6 (Optional) Change the setting for the MOB as well.

7 Save your settings and close the file.

8 Restart the `vmware-hostd` process by entering the following command:

```
service mgmt-vmware restart
```

**Example A-2.** A proxy.xml File Modified To Support the SDK and the MOB

```
<config>
    <EndpointList>
        <_length>7</_length>
        <_type>vim.ProxyService.EndpointSpec[]</_type>
        <e id="0">
            <_type>vim.Proxyservice.NamedPipeServiceSpec</_type>
            <serverNamespace>/</serverNamespace>
            <accessMode>httpsWithRedirect</accessMode>
            <pipeName>\\.\pipe\vmware-vpxd-webserver-pipe</pipeName>
        </e>
        <e id="1">
            <_type>vim.ProxyService.LocalServiceSpec</_type>
            <serverNamespace>/sdk</serverNamespace>
            <accessMode>httpAndHttps</accessMode>
            <port>8085</port>
        </e>
        <e id="2">
            <_type>vim.ProxyService.LocalServiceSpec</_type>
            <serverNamespace>/ui</serverNamespace>
            <accessMode>httpsWithRedirect</accessMode>
            <port>8086</port>
        </e>
        <e id="3">
            <_type>vim.ProxyService.NamedPipeServiceSpec</_type>
            <serverNamespace>/mob</serverNamespace>
            <accessMode>httpAndHttps</accessMode>
            <pipeName>\\.\pipe\vmware-vpxd-mob-pipe</pipeName>
        </e>
        <e id="4">
            <_type>vim.ProxyService.NamedPipeServiceSpec</_type>
            <serverNamespace>/vod</serverNamespace>
            <accessMode>httpsWithRedirect</accessMode>
            <pipeName>\\.\pipe\vmware-vpxd-webserver-pipe</pipeName>
        </e>
```

## HTTP Configuration for vCenter Server

You can modify the reverse proxy configuration for vCenter Server using the following procedure. The configuration file is the same as for ESX 5.0, but in a different location. For more detail about the file contents, see "To modify the Web proxy service on ESX or ESXi 4.1 or 5.0 to support HTTP" on page 29.

**NOTE** Your installation might have the configuration file in a different location, such as `C:\Program Data\VMware VirtualCenter\proxy.xml`.

### To modify the Web proxy service on vCenter Server 4.1 to support HTTP

1 Log in to the vCenter Server system as the Windows Administrator of the machine.

2 Change to the directory containing the `proxy.xml` file:

```
C:\Documents and Settings\AllUsers\Application Data\VMware VirtualCenter
```

3 Use a text editor to open the `proxy.xml` file.

4 Find the section of the file associated with the /sdk.

5 Change the `accessMode` to `httpAndHttps.`

6 Restart the service from a command line or from the Windows Services control panel.

# Scripting the C# DLL Build

B

When you generate the .NET stubs to run C# samples in the SDK, you get best performance if you pre-compile the XML serializers into separate DLLs. The procedure in "Build the C# vSphere DLLs" on page 20 describes how to pre-compile the XML serializers and modify the class declaration to use a pre-compiled serializer DLL. The result of these changes is to decrease initialization time when you instantiate the `VimService` class.

You can choose to script the procedure to build the DLLs. Using a script reduces the chance of errors and is faster than doing the procedure manually. This example script illustrates the procedure to build the DLLs with pre-compiled serializers.

This example script requires several environment variables to locate the source files and tools. The script also requires that you install a SED utility, which it uses to make the necessary changes to the C# source. You can find SED for Windows, or other capable tools, on the Web.

**Example B-1.** Example Script To Build C# DLLs

```
@echo off
if "%WS_SDK_HOME%"=="" (
   echo Set WS_SDK_HOME=^<absolute path to vsphere-ws directory of unzipped SDK^>.
   goto end
)
if "%WSDL_HOME%"=="" (
   rem This long line wraps:
   echo Set WSDL_HOME=%WS_SDK_HOME%\vsphere-ws\wsdl\vim25 ^(or absolute path to directory
               containing vim and vim25 WSDL subdirectories^).
   goto end
)
if "%SED_HOME%"=="" (
   echo Install SED and Set SED_HOME=^<absolute path to directory containing sed.exe^>.
   goto end
)
if not exist %WS_SDK_HOME%\vsphere-ws\dotnet\cs\samples (
   rem This long line wraps:
   echo Did not find %WS_SDK_HOME%\vsphere-ws\dotnet\cs\samples directory.  Please unzip SDK
               files and check WS_SDK_HOME setting.
   goto end
)
cd %WS_SDK_HOME%\vsphere-ws\dotnet\cs\samples

echo "Building Vim25Api namespace DLLs..."
if exist VimService.cs* del VimService.cs*
rem This long line wraps:
wsewsdl3.exe /n:Vim25Api /type:webClient /l:CS "%WSDL_HOME%\vim.wsdl"
               "%WSDL_HOME%\vimService.wsdl"
if exist Vim25Service.dll del Vim25Service.dll
rem This long line wraps:
csc.exe /t:library /out:Vim25Service.dll /r:"%WSE_HOME%\Microsoft.Web.Services3.dll"
               VimService.cs
echo "...Generating Vim25Api serializers..."
if exist Vim25Service.XmlSerializers.dll del Vim25Service.XmlSerializers.dll
sgen /p Vim25Service.dll
```

```
rem These 2 long SED lines wrap:
"%SED_HOME%"\sed.exe "s#\[System.Xml.Serialization.XmlIncludeAttribute#//&#"  <VimService.cs
               >VimService.cs.temp
"%SED_HOME%"\sed.exe "s#public partial class VimService
               #\[System.Xml.Serialization.XmlSerializerAssemblyAttribute\(AssemblyName =
               \"Vim25Service.XmlSerializers\")\]\n   &#" <VimService.cs.temp >VimService.cs
rem This long line wraps:
csc.exe /t:library /out:Vim25Service.dll /r:"%WSE_HOME%\Microsoft.Web.Services3.dll"
               VimService.cs

echo "Copying DLLs into library directory..."
copy Vim25Service*.dll lib
:end
```

# Index

## W



## X