vSphere Web Services SDK Developer's Setup Guide

17 APR 2018
VMware vSphere 6.7
vCenter Server 6.7
VMware ESXi 6.7

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

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.

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. The Revision History summarizes the significant changes in each version of this guide.

Table 1. Revision History

Revision	Description
17Apr2018	Minor updates for vsphere 6.7.
05Nov2016	Update and restructure procedures for modifying service endpoints with vSphere 6.5. Simplify process of building C# DLLs and samples.
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).

Table 1. Revision History (Continued)

Revision	Description
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 Developer's Setup 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 the following 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 Chapter 5 Endpoint Configuration for HTTP.)

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 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.

Language/Tool Context	Java	C#
Development environment or framework	J2SE 7 or J2SE 8 (also known as J2SE 1.8). For best results, use J2SE 1.7 or higher.	Microsoft Visual Studio 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 download 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

vSphere includes 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, which includes vCenter Single Sign-On Server, Lookup Service Server, and other features. This configuration combines all the platform services in the same installation with vCenter Server.
- vCenter Server with an external Platform Services Controller, where the platform services are installed on a different host or in a different virtual machine from vCenter Server.

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, and in Chapter 3 Setting Up for Microsoft C# Development.

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 with many previous versions of VMware vSphere servers and its predecessor, VMware Infrastructure, including:

- ESX/ESXi 6.7, 6.5, 6.0, 5.5, 5.1, and 5.0
- vCenter Server 6.7, 6.5, 6.0, 5.5, 5.1, 5.0

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 were 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. 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 the following topics:

Java Development Requirements for the Web Services SDK

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. Java 2, Standard Edition, version 8.0 (J2SE 1.8.x) is also supported.

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 for the Web Services SDK for details.

Set Up for Java Development for the Web Services SDK

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 for the Web Services SDK

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 with the Web Services SDK

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.

Procedure

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 for the Web Services SDK 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 Web Services SDK SimpleClient Sample Application to Validate Setup.

Batch Files and Shell Scripts for Building and Running Samples in the Web Services SDK

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.

Filename	Description	Usage note
`build.bat` `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` `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 for the Web Services SDK

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 Chapter 5 Endpoint Configuration for HTTP. 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.

Procedure

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 for the Web Services SDK

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 in the Web Services SDK

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\JAXWS\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 Web Services SDK SimpleClient Sample Application to Validate Setup.

Use the Included Build Scripts in the Web Services SDK

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.

Procedure

1. Open a command prompt.
2. Navigate to the subdirectory containing the `build.bat` and `build.sh` files.

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

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 the following example. 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.

   Successful Stub Generation and Compilation Using the build.bat Script

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

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 Web Services SDK SimpleClient Sample Application to Validate Setup

Running the Web Services SDK 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 in the Web Services SDK

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.general.SimpleClient --url https://yourFQDNservername/sdk
                    --username username --password password [--ignorecert ignorecert]
```

The following example shows sample output from the SimpleClient sample program.

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, do the following procedure.

Procedure

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 the following topics:

C# Development Requirements for the Web Services SDK

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 Windows Driver Kit (WDK).

Software Downloads for C# Development with the Web Services SDK

The VMware vSphere Web Services SDK is included in the vSphere Management SDK package.

You can obtain the Management SDK from https://developercenter.vmware.com.

Setting Up for C# Development with the Web Services SDK

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# with the Web Services SDK

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.

Procedure

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

   Use Microsoft Visual Studio 2012.

   When you install Visual Studio, select the option **Microsoft Foundation Classes for C++**.
2. Obtain the Microsoft .NET Framework, if it is not included in the Microsoft Visual programming environment.

   Use .NET version 4.5 or later, depending on your Visual Studio version.
3. Install Microsoft Windows Driver Kit (WDK).
4. Download and unzip the VMware vSphere Web Services SDK package from the VMware Web site at https://developercenter.vmware.com.

Building the C# SSO DLL for the Web Services SDK

In the vSphere Web Services SDK, VMware supplies several sample Single Sign-On (SSO) clients for Visual Studio. 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 in the Web Services SDK.

Build the C# SSO DLL with the Web Services SDK

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 with the Web Services SDK.

Procedure

1. Open a Visual Studio Native Tools command prompt.
2. Navigate to the .NET subdirectory for SSO client samples.

   cd *path_to_sdk*\ssoclient\dotnet\cs\samples
3. Generate a test certificate and STSService stubs using the `build.bat` script.

   ```
   build.bat
   ```

Building the C# vSphere DLLs with the Web Services SDK

In the vSphere Web Services SDK, VMware supplies sample vSphere clients for Visual Studio. 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.

Build the C# vSphere DLLs for the Web Services SDK

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

Prerequisites

Prerequisites for Building the C# vSphere DLLs

- Install a C# development environment, as described in Setting Up for C# Development with the Web Services SDK.

Procedure

1. Open a Visual Studio Native Tools command shell as Administrator.

   Use Visual Studio 2012 or 2013. Visual Studio 2015 is not supported for this step.
2. Navigate to the .NET subdirectory for vSphere client samples.

   cd *path_to_sdk*\vsphere-ws\dotnet\bin
3. Run the build script to generate the `VimService*.dll` files from the WSDL.

   ```
   build.bat
   ```

   The script builds the client-side API bindings and compiles them into the Vim25Service DLL.

Building the C# Sample Programs in the Web Services SDK

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.

To build the C# sample programs

Procedure

1. Launch Visual Studio and load the solution file, `Samples2012.sln`.

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

2. 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.

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

Running the Microsoft .NET C# Version of SimpleClient in the Web Services SDK

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

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 with the Web Services SDK.
- The SSO DLL is built. See Building the C# SSO DLL for the Web Services SDK.
- The vSphere DLLs are built. See Building the C# vSphere DLLs with the Web Services SDK.

Procedure

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:

   path_to_sdk`\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.

   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 of the Web Services SDK

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

Proxy Server Connection Problem with the Web Services API

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.

vSphere Server Certificates

4

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 chapter includes the following topics:

Secure Client-Server Communications

To connect to the server using HTTPS, client applications must verify the identity 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.

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 for the Web Services SDK 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.
- 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.

Procedure

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.

What to do next

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 with the Web Services SDK. For Java developers, see Set Up for Java Development for the Web Services SDK.

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

In some configurations, you might need to import certificates into your Active Directory domain.

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.

Procedure

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 an 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.

Creating a Security Certificate for .NET SSO Samples

By default VMware provides you a certificate in the .NET SSO samples directory which is not password protected. All the SSO samples will work with this certificate, but you can replace it with your own certificate.

The procedure you use to replace the security certificate depends on which version of Windows you are using. If you are using Windows 10 or later, use the procedure Create a Security Certificate for .NET SSO Samples Using Windows 10 or Later. If you are using Windows 8 or earlier, use the procedure Create a Security Certificate for .NET SSO Samples Using Windows 8 or Earlier.

Create a Security Certificate for .NET SSO Samples Using Windows 10 or Later

By default VMware provides you a certificate in the .NET SSO samples directory which is not password protected. Use this procedure to replace the certificate if you are running Windows 10 or later.

You choose to replace the default certificate supplied with the SDK, at `SDK/ssoclient/dotnet/cs/samples/certificate`.

Prerequisites

- This procedure applies to Windows 10 or later.
- You must have PowerShell installed, because Makecert is deprecated.

Procedure

1 Open a PowerShell window, running as Administrator.

Type **`powershell`** in the task bar search field, then right-click **Windows PowerShell** and select **Run as Administrator**.

2 Change to the .NET SSO samples directory.

`cd installed_directory/SDK/ssoclient/dotnet/cs/samples`

3 Delete the default certificate provided by VMware.

```
del *.pfx *.cer
```

4 Use the `New-SelfSignedCertificate` command to generate a PowerShell certificate object.

```
$cert = New-SelfSignedCertificate -certstorelocation cert:\localmachine\my -
subject "CN=*.vmware.com, OU=Ecosystem Engineering, O=`"VMware, Inc.`", L=Palo
Alto, ST=California, C=US" -KeySpec KeyExchange -KeyExportPolicy Exportable -
KeyUsage DigitalSignature
```

5 Create a password to protect the certificate.

```
$pwd = ConvertTo-SecureString -String 'password' -Force -AsPlainText
```

6 What on earth is going on here?

```
$path = 'cert:\localmachine\my\' + $cert.thumbprint
```

7 Use the `Export-PfxCertificate` command to generate a certificate file from the certificate object.

```
Export-PfxCertificate -cert $path -FilePath certificate\testssoclient.pfx -
Password $pwd
```

8 Update the certificate password used in the .NET SSO samples.

The password is used in line 553 of the file `vmware.binding.wstrust/samltokenhelper.cs`, in the following statement:

```
signingCertificate.Import(certificateFile, "password",
X509KeyStorageFlags.MachineKeySet);
```

9 Rebuild the SSO solution.

What to do next

Run the .NET SSO samples, using your new certificate.

Create a Security Certificate for .NET SSO Samples Using Windows 8 or Earlier

By default VMware provides you a certificate in the .NET SSO samples directory which is not password protected. Use this procedure to replace the certificate if you are running Windows 8 or earlier.

You choose to replace the default certificate supplied with the SDK, at `SDK/ssoclient/dotnet/cs/samples/certificate`.

Prerequisites

This procedure applies to Windows 8 or earlier.

Procedure

1 Open a Visual Studio Developer command prompt, running as Administrator.

2 Change to the .NET SSO samples directory.

cd *installed_directory*/SDK/ssoclient/dotnet/cs/samples

3 Delete the default certificate provided by VMware.

del *.pfx *.cer

4 Use the `makecert` command to generate a new certificate file.

makecert –r –pe –n "CN=*.vmware.com, OU=Ecosystem Engineering, O=\"VMware, Inc.\", L=Palo Alto, ST=California, C=US" –sky exchange –sv certificate\testssoclient.pvk certificate\testssoclient.cer

5 Convert the certificate file to a `.pfx` format.

pvk2pfx –pvk certificate\testssoclient.pvk –spc certificate\testssoclient.cer –pfx certificate\testssoclient.pfx

6 Delete the `.pvk` and `.cer` files, which are no longer needed.

```
del certificate\testssoclient.pvk
del certificate\testssoclient.cer
```

What to do next

Run the .NET SSO samples, using your new certificate.

Endpoint Configuration for HTTP

5

For development environments, you can simplify the connection process by configuring an HTTP connection rather than an HTTPS connection to the Web service. If your client also connects to a vSphere Automation API endpoint, you can use a similar procedure to configure the vSphere Automation API connection.

The procedure to modify the service configuration differs, depending on the service type. Choose one of the following options that applies to your situation:

- HTTP Configuration for Web Services API Endpoint
- HTTP Configuration for vSphere Automation API Endpoint

This chapter includes the following topics:

Modifying Service Endpoint Configurations to Support HTTP

ESXi, vCenter Server, and vSphere Automation API endpoints run by default on port 443, as secure Web services that can be accessed using SSL or TLS over HTTP (HTTPS). However, for a development environment, you might want to simplify the connection process from a client application by configuring the target services to support HTTP.

Note The HTTP protocol without SSL or TLS is insecure. Your user credentials could be intercepted by wire capture software.

The API service endpoints are handled by a reverse-proxy service, which has a configuration file that can be modified to specify support for HTTP as an accepted protocol. If you configure the service for HTTP, you do not need to import the server certificates on the client development workstation. Modifying the service configuration to support HTTP access is recommended for test or development environments only, not for production deployments. The default protocol, HTTPS, provides better security for production deployments.

HTTP Configuration for Web Services API Endpoint

You can modify the Web Services API endpoint configuration to accept HTTP connections, in addition to or instead of HTTPS connections.

HTTP Configuration for Web Services API Endpoint on ESXi or vCenter Server for Windows

This procedure applies to version 5.5 or later ESXi or vCenter Server for Windows.

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*.

Procedure

1. Log in to a shell window or File Explorer with **root** or **administrator** privileges.
2. Change directories to the location of the endpoint configuration file.

   The location differs, depending on the platform.

   - For ESXi:
     `/etc/vmware/rhttpproxy`
   - For vCenter Server for Windows:
     `C:\Program Data\VMware\vCenterServer\cfg\vmware-rhttpproxy\endpoints.conf.d`
3. Copy the `endpoints.conf` file to a temporary directory for editing.
   - On ESXi:
     `# cp endpoints.conf /tmp/endpoints.conf`
   - On vCenter Server for Windows, use File Explorer to copy the file.
4. Change the permissions on the temporary `endpoints.conf` file to allow editing.
   - On ESXi:
     `# chmod +w /tmp/endpoints.conf`
   - On vCenter Server for Windows, right click the file in File Explorer and select **Properties** to change file permissions.
5. Use a text editor to open the temporary file.
   - On ESXi:
     `# vi /tmp/endpoints.conf`
   - On vCenter Server for Windows, choose any text editor, such as Notepad, from the **Start** menu.

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

   The line looks similar to this:

   ```
   /sdk    local    8085              redirect     allow
   ```

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

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

   ```
   /sdk    local    8085              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 similar to this:

   ```
   /sdk    local    8085              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 similar to this:

   ```
   /mob    local    8085              allow     allow
   ```

10 Save your settings and close the file.

11 Change the permissions on the temporary file to disable editing.

   - On ESXi:
     `# chmod –w /tmp/endpoints.conf`
   - On vCenter Server for Windows, right click the file in File Explorer and select **Properties** to change file permissions.

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

   - On ESXi:
     `# cp endpoints.conf endpoints.conf.old`
   - On vCenter Server for Windows, use File Explorer to copy the file.

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

   - On ESXi:
     `# cp /tmp/endpoints.conf endpoints.conf`
   - On vCenter Server for Windows, use File Explorer to copy the file.

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

   - On ESXi:

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

   - On vCenter Server for Windows:

     From the Windows menu, choose **Control Panel > Administrative Tools > Services**, right click the rhttpproxy service, and choose **Restart**.

Example: 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 Web Services API Endpoint on vCenter Server Appliance

You can modify the Web proxy service to support HTTP on vCenter Server Appliance.

You can modify from a shell window over an SSH connection, using the following procedure. If you do not have SSH enabled, see *vCenter Server Appliance Configuration* or *VMware vCenter Server Appliance Progrmmaing Guide*. This procedure applies to version 6.0 or later vCenter Server Appliance.

Procedure

1. Log in to a shell window with `root` privileges.
2. Change directories to the location of the endpoint configuration file.
   - For vCenter Server Appliance:
     cd /etc/vmware-rhttpproxy/endpoints.conf.d
3. Copy the `endpoints.conf` file to a temporary directory for editing.
   - On vCenter Server Appliance:
     `# cp vpxd-rhttpproxy-endpoint.conf /tmp/vpxd-rhttpproxy-endpoint.conf`
4. Change the permissions on the temporary vpxd-rhttpproxy-endpoint.conf file to allow editing.
   - On vCenter Server Appliance:
     `# chmod +w /tmp/vpxd-rhttpproxy-endpoint.conf`
5. Use a text editor to open the temporary file.
   - On vCenter Server Appliance:
     `# vi /tmp/vpxd-rhttpproxy-endpoint.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     8085                redirect     allow
```

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

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

```
/sdk     local     8085                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 similar to this:

```
/sdk     local     8085                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 similar to this:

```
/mob     local     8085                allow      allow
```

10 Save your settings and close the file.

11 Change the permissions on the temporary file to disable editing.

- On vCenter Server Appliance:
  # chmod –w /tmp/vpxd–rhttpproxy–endpoint.conf

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

- On vCenter Server Appliance:
  # cp vpxd–rhttpproxy–endpoint.conf vpxd–rhttpproxy–endpoint.conf.old

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

- On vCenter Server Appliance:
  # cp /tmp/vpxd–rhttpproxy–endpoint.conf vpxd–rhttpproxy–endpoint.conf

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

- On vCenter Server Appliance:

  /etc/init.d/vmware-rhttpproxy restart

For an example of the contents of a vpxd-rhttpproxy-endpoint.conf file modified to support HTTP connections, see HTTP Configuration for Web Services API Endpoint.

HTTP Configuration for vSphere Automation API Endpoint

You can also modify the vAPI endpoint configuration for vCenter Server to accept HTTP connections in addition to or instead of HTTPS connections. This is useful for clients that interact both with the vSphere Web Services API endpoint and the vSphere Automation API endpoint in a development environment. Use the following procedure.

Procedure

1. Log in to a shell window or File Explorer with **root** or **administrator** privileges.
2. Change directories to the location of the endpoint configuration file.

   The location differs, depending on the platform.

   - For vCenter Server Appliance:
     `/etc/vmware-rhttpproxy/endpoints.conf.d`
   - For vCenter Server for Windows:
     `C:\Program Data\VMware\vCenterServer\cfg\vmware-rhttpproxy\endpoints.conf.d`
3. Copy the vapi-`endpoint.conf` file to a temporary directory for editing.

   - On vCenter Server Appliance:
     `# cp vapi-endpoint.conf /tmp/vapi-endpoint.conf`
   - On vCenter Server for Windows, use File Explorer to copy the file.
4. Change the permissions on the temporary file to allow editing.

   - On vCenter Server Appliance:
     `# chmod +w /tmp/vapi-endpoint.conf`
   - On vCenter Server for Windows, right click the file in File Explorer and select **Properties** to change file permissions.
5. Use a text editor to open the temporary file.

   - On vCenter Server Appliance:
     `# vi /tmp/vapi-endpoint.conf`
   - On vCenter Server for Windows, choose any text editor, such as Notepad, from the **Start** menu.
6. Navigate to the line that specifies the endpoint for REST connections, which begins with `/rest`.

   The line looks similar to this:

   ```
   /rest    local    12346              redirect     allow
   ```
7. To enable HTTP connections for REST clients, change the word `redirect` to `allow`.

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

   ```
   /rest    local    12346              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 `/rest` line looks similar to this:

   ```
   /rest    local    12346              allow     reject
   ```

9 Navigate to the line that specifies the endpoint for clients that use language bindings rather than REST, which begins with `/api`.

   The line looks similar to this:

   ```
   /api    local    12346              redirect     allow
   ```

10 To enable HTTP connections for clients using language bindings rather than REST, change the word `redirect` to `allow`.

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

   ```
   /api    local    12346              allow     allow
   ```

11 Save your settings and close the file.

12 Change the permissions on the temporary file to disable editing.

   - On vCenter Server Appliance:
     `# chmod –w /tmp/vapi-endpoint.conf`
   - On vCenter Server for Windows, right click the file in File Explorer and select **Properties** to change file permissions.

13 Copy the original vapi-`endpoint.conf` file to a backup file.

   - On vCenter Server Appliance:
     `# cp vapi-endpoint.conf vapi-endpoint.conf.old`
   - On vCenter Server for Windows, use File Explorer to copy the file.

14 Copy the temporary file back, replacing the original vapi-`endpoint.conf` file.

   - On vCenter Server Appliance:
     `# cp /tmp/vapi-endpoint.conf vapi-endpoint.conf`
   - On vCenter Server for Windows, use File Explorer to copy the file.

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

   - On vCenter Server Appliance:

     /etc/init.d/vmware-rhttpproxy restart

   - On vCenter Server for Windows:

     From the Windows menu, choose **Control Panel > Administrative Tools > Services**, right click the rhttpproxy service, and choose **Restart**.

Example: A `vapi-endpoint.conf` File Modified To Support HTTP connections for API and REST

```
/vapiendpoint/health            local          12346       allow          allow
/vapiendpoint/resourcebundle    local          12346       allow          allow
/rest                           local          12346       allow          allow
/site/api                       local          12346       redirect       allow
/site/rest                      local          12346       redirect       allow
/api/                           local          12346       allow          allow
```