Developer’s Setup Guide

VMware vSphere Web Services SDK 5.0

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-000670-01

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

http://www.vmware.com/support/

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

docfeedback@vmware.com

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

Contents

About This Book 5

1 About the vSphere Web Services SDK 7

Knowledge Required for Using the vSphere Web Services SDK 7

Programming Languages Supported by the vSphere Web Services SDK 8

Types of Applications That You Can Build Using This SDK 8

vSphere Web Services SDK Package Contents 9

SDK Versions and VMware vSphere Product Compatibility 10

2 Setting Up for Java Development 11

Requirements 11

Setup Instructions for Java Development 11

Setting Up for Development Using JAX-WS 12

Setting Up for Development Using Apache Axis 12

Setting Environment Variables for Axis Development 13

Batch Files and Shell Scripts for Building and Running Samples 15

Importing Server-Certificates into the Java Keystore 15

Generating Stubs and Compiling Classes 16

Precompiled JAX-WS Samples 17

Precompiled Axis Samples 17

Using the Included Build Scripts 17

Running the SimpleClient Sample Application to Validate Setup 18

Running a Sample Application Using the Provided Scripts 18

Troubleshooting Setup Issues 20

3 Setting Up for Microsoft C# Development 21

Requirements 21

Environment Variable Settings 21

Setup Instructions for C# Development 22

Building the C# Samples 22

Building for a Chosen Version of Visual Studio 22

Iterating the Build for Both Vim and Vim25 Namespaces 23

Running the Microsoft .NET (C#) Version of SimpleClient 24

Troubleshooting Setup Issues 25

Appendix: vSphere Server Certificates 27

Secure Client-Server Communications 27

Simplified Security Setup for Development Environment 27

Obtaining Server Certificates 28

Obtaining Certificates by Using the vSphere Client 28

Obtaining Certificates by Connecting Directly to Server Systems 29

Modifying Server Configurations to Support HTTP 29

Index 33

About This Book

This book, the *Developer's Setup Guide*, provides information about setting up your development environment to use the VMware® vSphere Web Services SDK 5.0.

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
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 *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 *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.

Technical Support and Education Resources

The following sections describe the technical support resources available to you. To access the current versions of other VMware books, go to http://www.vmware.com/support/pubs.

Online and Telephone Support

To use online support to submit technical support requests, view your product and contract information, and register your products, go to http://www.vmware.com/support.

Support Offerings

To find out how VMware support offerings can help meet your business needs, go to http://www.vmware.com/support/services.

VMware Professional Services

VMware Education Services courses offer extensive hands-on labs, case study examples, and course materials designed to be used as on-the-job reference tools. Courses are available onsite, in the classroom, and live online. For onsite pilot programs and implementation best practices, VMware Consulting Services provides offerings to help you assess, plan, build, and manage your virtual environment. To access information about education classes, certification programs, and consulting services, go to http://www.vmware.com/services.

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 various 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 ESX™, ESXi, and VMware vCenter Server systems.

This *Developer's Setup Guide* provides information about setting up the development environment for using Java and Microsoft .NET (using the C# programming language) to create new applications, and 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" on page 7
- "Programming Languages Supported by the vSphere Web Services SDK" on page 8
- "Types of Applications That You Can Build Using This SDK" on page 8
- "vSphere Web Services SDK Package Contents" on page 9
- "SDK Versions and VMware vSphere Product Compatibility" on page 10

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 the programming language of your choice, 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 WSDL (Web Services Description Language) 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. SOAP (Simple Object Access Protocol) 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 (secure HTTP, which 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 29.)

You should also know about basic ESX, ESXi, and vCenter Server operations. See the VMware vSphere Documentation page at http://www.vmware.com/support/pubs/vi_pubs.html.

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 (proxy code) from Web-services WSDL files. However, VMware recommends using Java or C#, languages for which SOAP toolkits are readily available. Also, the vSphere Web Services SDK package includes sample client applications developed in both Java and C#.

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 Apache Axis 1.4 libraries.

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

Language/Tool Context	Java	C#
Development environment or framework	J2SE 6.0 (aka, J2SE 1.6—J2SE 1.6 b22 or subsequent version recommended)	Microsoft Visual Studio Microsoft Visual C#
Web-services-client application development toolset ("SOAP toolkit")	JAX-WS 2.1 (Java API for XML Web Services) or Apache Axis 1.4	Microsoft .NET Framework

Developers, scripters, and administrators using Microsoft PowerShell or Perl can use the vSphere API through various toolkits provided by VMware. You can learn more at http://communities.vmware.com/community/developer.

Types of Applications That You Can Build Using This SDK

The types of applications that you can develop using the vSphere Web Services SDK include 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. Here are some examples of the operational tasks that you can automate using the vSphere 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 emitted 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 per-host or per-virtual machine operation.

vSphere Web Services SDK Package Contents

The vSphere Web Services SDK is a zip file that includes:

- 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 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`) have been included.)

  NOTE The precompiled Java samples (`samples.jar`) have been compiled using JDK 1.6 b22 from stubs generated by the Java API for XML Web Services (JAX-WS) libraries in J2SE 6.0, 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, you must rebuild the samples (using the build script).

  The vSphere Web Services SDK also contains precompiled Java samples that have been compiled using JDK 1.6 using stubs generated by the Apache Axis 1.4 libraries. These samples work only with these specific versions of Java and Axis. To avoid a performance issue, use JDK 1.5.0_08 or later. See Knowledge Base article 2983901, “Performance Issue (for VI SDK 2.0) with JDK Versions Prior to JDK 1.5.0_08,” for details.

- Batch files and shell scripts (build.bat, build.sh) that automate the build process for Java and C# client applications, respectively.
- Batch files and shell scripts (`run.bat`, `run.sh`) that facilitate running the Java samples from the Windows command prompt.
- API reference documentation (*vSphere API Reference*) that provides language-neutral descriptive information (object type definitions, properties, and method signatures, for example) about the VMware vSphere API and the object model.
- Access to technical publications, including:
  - *vSphere Web Services SDK Developer’s Setup Guide* (this book), which helps you set up your development environment and run sample applications using either Java or C#
  - *vSphere Web Services SDK Programming Guide*, which provides conceptual and prescriptive information about how to develop client applications using the vSphere Web Services SDK.

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 21.

SDK Versions and VMware vSphere Product Compatibility

VMware has released several SDK products to support various versions of the VMware vSphere product family. The VMware vSphere Web Services SDK 5.0 can be used with all versions of VMware vSphere servers and its predecessor, VMware infrastructure, including ESXi 5.0, ESX/ESXi 4.1, ESX/ESXi 4.0, vCenter Server 5.0, vCenter Server 4.1/4.0, ESX Server 3.5/3i/3.0.x, VirtualCenter Server 2.5/2.0.x. All versions are supported by using the appropriate WSDL files, as follows:

- `SDK\vsphere-ws\wsdl\vim25` contains WSDL files for use with ESXi 5.0, ESX/ESXi 4.1, ESX/ESXi 4.0, vCenter Server 4.1/4.0, ESX 3.5, and VirtualCenter 2.5 systems. As of vSphere 4.1, the vSphere API WSDL definitions are divided into several files. This architecture is for future implementations that may import portions of the vSphere API. Backwards compatibility is achieved because both WSDL configurations (`vim25` and `vim` directories) use a top level file with the same name (`vimService.wsdl`).
- `SDK\vsphere-ws\wsdl\vim` contains WSDL files for use with ESX Server 3.0.1 and VirtualCenter 2.0.

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, the 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 the vCenter Server, the server returns a "not implemented" or a "not supported" error message. 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 "not supported" fault.

Setting Up for Java Development 2

This chapter includes these topics:

- "Requirements" on page 11
- "Setup Instructions for Java Development" on page 11
- "Running the SimpleClient Sample Application to Validate Setup" on page 18
- "Troubleshooting Setup Issues" on page 20

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. VMware recommends using Java 2, Standard Edition, version 6.0 (J2SE 1.6.x), specifically JDK 1.6 b22 or later.

The Java Web services development toolset must be a SOAP (Simple Object Access Protocol) implementation that can be deployed to a Tomcat server. VMware recommends that you use one of the following client-side web services libraries:

- Java API for XML Web Services (JAX-WS) version 2.1. The JAX-WS 2.1 libraries are included with the JDK 1.6.
- Apache Axis 1.4. Axis is an open source project of the Apache Web services project.You can obtain Axis from the Apache Web-services project site at http://ws.apache.org/axis/.

NOTE The Apache Axis 1.4 libraries have a known issue that prevents them from working with J2SE 6.0 (JDK 1.6). If you use the Axis 1.4 libraries, you must use J2SE 5.0 (JDK 1.5_0_08). Support for the Apache Axis 1.4 bindings is deprecated in vSphere 5.0.

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 and Axis client libraries were tested with this *Developer's Setup Guide*.

The samples archive (`samples.jar`) includes all vSphere Web Services SDK samples. The samples include client-side proxy code (stub classes) generated using both the JAX-WS and Apache Axis 1.4 libraries. Samples using JAX-WS were generated using JDK 1.6 b22. Samples generated using Apache Axis 1.4 were generated using JDK 1.5_0_08.

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

Setting Up for Development Using JAX-WS

The samples generated using JAX-WS libraries and compiled using Java JDK 1.6 b22 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.

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

1 Create a directory for the JDK (assuming that the JDK is not installed) 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. VMware recommends using JDK 1.6 b22 or later. You can obtain the J2SE from http://java.sun.com/javase/downloads/.

3 Obtain the VMware vSphere Web Services SDK package from the VMware Web site http://www.vmware.com/download/sdk/.

4 Unpack the components into subdirectories created in step 1 above, 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
```

5 Import server-certificates and use the Java keytool utility to create a `vmware.keystore`. See "Importing Server-Certificates into the Java Keystore" on page 15 for details. Optionally, you can ignore server-certificate verification for any of the sample Java applications by passing the `--ignorecert` argument at runtime.

6 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:\jre1.6_0\jre`.

7 Add the precompiled sample files (`vim25.jar` and `samples.jar`) to your system CLASSPATH environment variable. As an alternative, you can use the `run.bat` script. To use the `run.bat` script, the JAVAHOME environment variable must be defined.

If you are not using JDK 1.6 b22, you must use the build.bat (or 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 16 for details.

8 Run the Java version of SimpleClient to test your setup. See "Running the SimpleClient Sample Application to Validate Setup" on page 18 for details.

Setting Up for Development Using Apache Axis

The samples generated using Apache Axis 1.4 libraries and compiled using Java JDK 1.5_0_08 include `vim.jar`, `vim25.jar`, `apputils.jar`, and `samples.jar`. You can use these libraries without generating new stubs and recompiling if you are using these same versions of the JDK and Axis.

To set up a development workstation to use Java and Apache Axis

1. Create directories for each of the components that you want to install (assuming that neither JDK nor Axis client is installed).

   Do not use spaces in the directory names, to avoid issues with some of the included SDK batch and script files. Table 2-1 lists some recommended naming conventions.

   Table 2-1. Recommended Directory Structures

	Windows	Linux
SDK	`c:\devprojects\visdk21`	`~\apps\visdk`
Apache Axis	`c:\apache\axis`	`~\apps\apache\axis`
Java 2, Standard Edition (J2SE)	`c:\java\jdk1.5.0_nn`	`~\apps\java\jdk1.5.0_nn`
	`c:\java\jre1.5.0_nn`	`~\apps\java\jre1.5.0_nn`

2. Install the Java 2 Platform, Standard Edition (J2SE) 5.0. VMware recommends using JDK 1.5_0_08 due to improved performance (for Java sockets). You can obtain the J2SE from http://java.sun.com/javase/downloads/previous.jsp.

3. Obtain the Apache Axis 1.4 client-side Web services libraries from the Apache Web services Web site at http://ws.apache.org/axis/java/releases.html.

4. Obtain the VMware vSphere Web Services SDK package from VMware Web site http://www.vmware.com/download/sdk/.

5. Unpack the various components into subdirectories created in step 1 above, using the provided installer if appropriate. The J2SE uses an installation wizard. The Axis zip file and the SDK unpack 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-WS-SDK-4.1.0-251329.zip
     ```

6. Import server-certificates and use the Java keytool utility to create a `vmware.keystore`. See "Importing Server-Certificates into the Java Keystore" on page 15 for details.

7. (Optional) To ignore server-certificate verification for any of the sample Java applications, you can pass the `--ignorecert` argument at runtime instead of importing server certificates.

8. Configure environment settings as detailed in "Setting Environment Variables for Axis Development" on page 13. To run any of the Java sample applications using the provided run script (`run.bat`, `run.sh`), you must set only AXISHOME, JAVAHOME, VMKEYSTORE, and WBEMHOME.

9. Use the build.bat (or build.sh, on Linux) to generate stubs and compile into `vim.jar`, `vim25.jar`, `apputils.jar`, and `samples.jar`. The build scripts perform all necessary tasks for you, including setting the CLASSPATH and PATH. See "Generating Stubs and Compiling Classes" on page 16 for details.

   NOTE If you are using Axis 1.4 and JDK 1.5, you do not need to use the build script. Instead, add `vim.jar`, `vim25.jar`, `apputils.jar`, and `samples.jar` to your system CLASSPATH. Alternatively, set the AXISHOME, JAVAHOME, VMKEYSTORE, and WBEMHOME and use the `run.bat` (or `run.sh`).

10. Run the Java version of SimpleClient to test your setup. See "Running the SimpleClient Sample Application to Validate Setup" on page 18 for details.

Setting Environment Variables for Axis Development

The environment variable definitions set depend on whether you want to use the provided scripts (`build.bat`, `build.sh`; `run.bat`, `run.sh`). To use the provided scripts, you must set four environment variables: AXISHOME, JAVAHOME, VMKEYSTORE, and WBEMHOME. See Table 2-2 for details.

If you do not want to use the scripts, you must set CLASSPATH and PATH environment variables so that they include the Axis libraries and the J2SE libraries. If these variables do not already exist on your development workstation, you must create them. Table 2-2 lists environment variable names, directories, and filenames that you must specifically set. The SDKHOME environment variable is for convenience.

To set environment variables required by the build and run scripts

1 Create environment variables for AXISHOME, JAVAHOME, SDKHOME, VMKEYSTORE, and WBEMHOME, as defined in Table 2-2.

- For Microsoft Windows development workstations, you must set System (rather than User) environment variables.
- For Linux development workstations, you must export the variables in your user profile.

2 To the System CLASSPATH, add the complete path to all Axis client-side library JARs and additional libraries provided with the SDK specified in Table 2-2.

3 To the System PATH environment variable, add the path to the Axis and Java runtime binaries, as specified in Table 2-2.

Table 2-2. Environment Variable Names and Required JAR and Other Files

Variable Name	Setting must Include...
AXISHOME	Complete path to the Apache Axis installation top-level directory. For example: `C:\apache\axis1.4`

CLASSPATH	Complete paths to specific JAR files and other libraries required by Java and Axis tools. Add these specific JAR files to the CLASSPATH (assumes that you have setup AXISHOME, JAVAHOME, and SDKHOME).
	`%AXISHOME%\lib\axis.jar`
	`%AXISHOME%\lib\axis-ant.jar`
	`%AXISHOME%\lib\commons-discovery-0.2.jar`
	`%AXISHOME%\lib\commons-logging-1.0.4.jar`
	`%AXISHOME%\lib\jaxrpc.jar`
	`%AXISHOME%\lib\log4j-1.2.8.jar`
	`%AXISHOME%\lib\saaj.jar`
	`%AXISHOME%\lib\wsdl4j-1.5.1.jar`
	`%JAVAHOME%\lib\tools.jar`
	`%SDKHOME%\samples\Axis\java\vim.jar`
	`%SDKHOME%\samples\Axis\java\vim25.jar`
	`%SDKHOME%\samples\Axis\java\apputils.jar`
	`%SDKHOME%\samples\Axis\java\samples.jar`
	`%SDKHOME%\samples\Axis\java\lib\activation.jar`
	`%SDKHOME%\samples\Axis\java\lib\mailapi.jar`
	`%SDKHOME%\samples\Axis\java\lib\wbem.jar`

JAVAHOME	Path to the Java runtime (JRE) directory. For example: `C:\jre1.5_0_08\jre`

PATH	Add the path to the Java and the Axis binary client tools to the system path. Assuming you setup the AXISHOME and JAVAHOME variables, you must add:
	`%AXISHOME%\bin`
	`%JAVAHOME%\bin`

SDKHOME	Path the top-level directory of the unpacked SDK download. For example: `C:\devprojects\visdk21\SDK\vsphere-ws\`

Table 2-2. Environment Variable Names and Required JAR and Other Files

Variable Name	Setting must Include...
VMKEYSTORE	Path to Java keystore. The VMKEYSTORE environment variable is used by the `run.bat` and `run.sh` batch files. Sample paths: `VMKEYSTORE=C:\VMware-Certs\vmware.keystore` [Windows] `VMKEYSTORE=/root/vmware-certs/vmware.keystore` [Linux] If you use the --ignorecert argument to run any Java samples (using the `run.bat` or `run.sh` script), you must still set the VMKEYSTORE environment variable. Set to any location (you do not need to import the certificate or create the actual keystore, though, if you use the --ignorecert argument).
WBEMHOME	Path to the Axis `java` directory. The `run.sh` script uses this variable to reference the WBEM (Web Based Enterprise Management) Java archive (`wbem.jar`). `WBEMHOME=%SDKHOME%\java\Axis\`

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 1.6 b22 (or the Axis 1.4 libraries with JDK 1.5_0_08), 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 2-3. Batch Files and Shell Scripts for Java [..\Axis\java\ Subdirectory]

Filename	Description	Usage note
`build.bat` `build.sh`	Checks for environment variables (AXISHOME, JAVAHOME, VMKEYSTORE, and WBEMHOME) and sets local classpath (by calling `lcp.bat`) using the variables as defined. The build.bat file then cleans up existing Java files (by calling `clean.bat`).	Use this script to generate client stubs and rebuild all sample applications. Successful execution of `build.bat` creates the `vim.jar`, `vim25.jar`, `apputils.jar`, and `samples.jar` files. Use the -w flag to recompile without regenerating stubs.
`lcp.bat` `lcp.sh`	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 prior to 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.

Importing Server-Certificates into the Java Keystore

This step is required `only` if both of the following conditions apply to your environment.

- The target servers use the default protocol, HTTPS.
- You do not plan to use the `--ignorecert` command-line argument.

Alternatively, 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 29. However, using HTTP is not recommended for production environments.

These instructions require that the JAVAHOME environment variable is set and that it is added to the PATH environment variable. These instructions also require that the certificate for each target server is located in the `C:\VMware-Certs` subdirectory. See "Obtaining Server Certificates" on page 28.

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, in subsequent steps.

Table 2-4. Keystore Path

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 -file certificate-filename -alias server-name -keystore vmware.keystore
```

For example:

```
C:\VMware-Certs>keytool -import -file rui.crt -alias sdkpubs01 -keystore vmware.keystore
```

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 this process for each target server.

Generating Stubs and Compiling Classes

The vSphere Web Services SDK includes Java archive files for the sample programs. There are two sets of sample `.jar` files: one set created using JDK 1.6 and the JAX-WS web services libraries, and one set created using JDK 1.5_0_08 and the Apache Axis 1.4 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 1.6. These files are located in the following directory:

```
%SDKHOME%\java\JAX-WS\lib
```

If your development environment is using the JAX-WS libraries and JDK 1.6 b22, 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 18.

Precompiled Axis Samples

The Axis samples include the `vim.jar`, `vim25.jar`, `samples.jar`, and `apputils.jar` files that were created using Axis 1.4 and JDK 1.5_0_08. These files are located in the following directory:

```
%SDKHOME%\java\Axis\
```

If your development environment includes these same Axis and Java versions, 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 18.

Using 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 proxy code (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 under the following conditions:

- You are using the JAX-WS libraries and you are using a Java version other than JDK 1.6 b22. To use the included build scripts, the JAVAHOME environment variable must be set.
- You are using the Axis libraries and you are using Java version other than JDK 1.5_0_08, or an Axis version other than Axis 1.4. To use the included build scripts, you must set the AXISHOME, JAVAHOME, and WBEMHOME environment variables (see "Setting Environment Variables for Axis Development" on page 13).

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. For JAX-WS, this directory is:

   ```
   %SDKHOME%\java\JAX-WS\
   ```

   For Axis, this directory is:

   ```
   %SDKHOME%\java\Axis\
   ```

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 completes. 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 `\vim` and `\vim25` subdirectories).

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

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

When the process completes successfully, 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 the next section, "Running the SimpleClient Sample Application to Validate Setup."

Running the SimpleClient Sample Application to Validate Setup

You can quickly 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. If you are using stubs generated by Axis 1.4, the scripts require you to set the AXISHOME, JAVAHOME, VMKEYSTORE, and WBEMHOME environment variables (see Table 2-2).

Running a Sample Application Using the Provided Scripts

You can use the run.bat or run.sh script to run any of the Java samples. 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 appropriate Java samples subdirectory for JAX-WS (`%SDKHOME%\java\JAXWS\com\vmware\samples\general\SimpleClient`) or Axis (`%SDKHOME%\java\Axis\com\vmware\samples\general\SimpleClient`):

```
run.bat com.vmware.samples.general.SimpleAgent --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 (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. For JAX-WS samples, this subdirectory is:

```
cd %SDKHOME%\java\JAXWS\samples
```

For Axis samples, this subdirectory is:

```
cd %SDKHOME%\java\Axis\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.samples.general.SimpleClient
               --url https://example.com/sdk --username pubs --password ***
               --ignorecert ignorecert
```

NOTE If error messages are raised due to system heap or other memory issues, you can give the Java VM more memory, as follows:

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

Troubleshooting Setup Issues

If you are unable to successfully run the SimpleClient, check your environment settings and all other setup tasks. Table 2-5 lists some common error messages and provides steps that you can take to resolve them.

Table 2-5. Java Runtime Error Messages

Symptom	Possible Solution
`Exception in thread "main" java.lang.NoClassDefFoundError: com/vmware/vim/ManagedObjectReference at com.vmware.samples.general.SimpleClient.createServiceRef(SimpleClient.java:32) at com.vmware.samples.general.SimpleClient.main(SimpleClient.java:214)`	■ Verify that the CLASSPATH includes all Axis libraries, directly or through use of the AXISHOME environment variable. ■ Verify that the AXISHOME variable has been set correctly.

`Caught Exception :  Name : org.apache.axis.AxisFault Message : (301)Moved Permanently Trace :`
`AxisFault faultCode: {http://xml.apache.org/axis/}HTTP`
` faultSubcode:`
` faultString: (301)Moved Permanently`
` faultActor:`
` faultNode:`
` faultDetail:`
`        {}:return code:  301`
 	
`{http://xml.apache.org/axis/}HttpErrorCode:301`
`(301)Moved Permanently at org.apache.axis.transport.http.HTTPSender.readFromSocket(HTTPSender.java:744)`
	■ After verifying that the server-certificate has been imported into the local keystore, run the command using HTTPS and provide the credentials (user account, password) for the server.

Setting Up for Microsoft C# Development 3

This chapter includes these topics:

- "Requirements" on page 21
- "Setup Instructions for C# Development" on page 22
- "Running the Microsoft .NET (C#) Version of SimpleClient" on page 24
- "Troubleshooting Setup Issues" on page 25

Requirements

The vSphere Web Services SDK includes C# (`.cs`) source files and Microsoft Visual Studio project files (solutions, or `.sln`) for Microsoft Visual Studio. 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 .NET Framework Software Development Kit. You must use a version of the .NET SDK that includes the command line C# compiler (csc.exe).

Environment Variable Settings

The vSphere Web Services SDK 5.0 includes two versions of the WSDL files:

- WSDL files that support vim25 for use with ESXi 5.0, ESX/ESXi 4.1, ESX/ESXi 4.0, vCenter Server 4.1 and 4.0, ESX 3.5, and VirtualCenter 2.5. See the contents of the directory `sdk\wsdl\vim25`.
- WSDL files that support vim for use with ESX Server 3.0.1 and VirtualCenter 2.0. See the contents of the directory `sdk\wsdl\vim`.

The batch files used to build samples require access to the WSDL files. If you intend to use any of the provided batch files, you must create the environment variable WSDLFILE and set it to the WSDL directory (`%SDKHOME%\wsdl`).

The vSphere Web Services SDK download contains batch files for setting up the samples for Microsoft Visual Studio. The download also includes the solution files (`.sln`) files).

If your Microsoft software setup varies from the default paths, you must create a VSINSTALLDIR environment variable as defined in Table 3-1.

Table 3-1. Environment Variable Names Required for Build Scripts

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. Use quotation marks around subdirectory pathnames that include spaces. For example: "C:\devstuff\Microsoft Visual Studio 8\Common7" "C:\devstuff\Microsoft Visual Studio 8\SDK"
WSDLFILE	Location of the WSDL files.	Set to %SDKHOME%\wsdl for use with build scripts.

To set environment variables required by the build script

1 Create the environment variable WSDLFILE, as defined in Table 3-1. Set System rather than User environment variables.

2 If your Microsoft development and .NET software has not been installed using default paths, create and set the VSINSTALLDIR environment variable as defined in Table 3-1.

Setup Instructions for C# Development

Specific setup instructions depend on whether your development workstation already meets some or all of the requirements and whether you plan to use the provided samples.

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.

VMware recommends using Microsoft Visual Studio 2005 or later, which includes the required .NET Framework (2.0 or later) and improved versions of Web-services-client tools. For more information, go to http://msdn.microsoft.com.

2 Obtain the Microsoft .NET Framework (if you do not already have it) from Microsoft, at http://msdn.microsoft.com.

3 Obtain the VMware vSphere Web Services SDK package from VMware Web site at http://www.vmware.com/download/sdk/.

4 Unpack the various components into appropriate subdirectories. Use Microsoft defaults for Microsoft Visual Studio, Microsoft .NET Framework, and Microsoft .NET Framework SDK.

To ensure that all paths to all tools (`wsdl.exe`, `csc.exe`) are set correctly, execute the command script from the Microsoft .NET SDK command prompt, which is available from the .NET SDK menu.

Building the C# Samples

In the vSphere Web Services SDK, VMware supplies sample clients for Visual Studio 2005, 2008, and 2010. The SDK includes a solution (`.sln`) file for each version of Visual Studio, such as `Samples2008.sln`. Each solution file refers to DLLs whose filenames also include the version of Visual Studio. For example, `Samples2008.sln` refers to `VimService2008.dll` and `VimService2008.XmlSerializers.dll`.

Building for a Chosen Version of Visual Studio

This procedure shows how to build the DLLs for your choice of Visual Studio version. The filenames you build will contain the actual version, whereas the procedure shows filenames that contain the place-holder string *20xx*. Substitute your Visual Studio version wherever you see *20xx* in a filename.

For example, if you plan to run the DLLs in the Visual Studio 2010 environment, use the switch `/out:VimService2010.dll` in step 4 and `/p VimService2010.dll` in step 5. These filenames are referenced in the `Samples2010.sln` solution file.

Iterating the Build for Both Vim and Vim25 Namespaces

In addition, you must do the sequence of step 3 through step 8 twice, substituting the actual values `Vim` and `Vim25` on successive passes wherever you see the place-holder string *`Vimyy`* in a filename. This is needed because the solution files reference both the old and the new API namespaces.

For example, if you plan to run the DLLs in the Visual Studio 2010 environment, use the switch `/n:VimApi` in step 3 and `/out:VimService2010.dll` in step 4 the first time through. Continue to step 8, making the same substitutions in the other steps. Return to step 3 and repeat until step 8, using the switch `/n:Vim25Api` in step 3, `/out:Vim25Service2010.dll` in step 4, and so on.

To build the C# samples

1 Open the Microsoft .NET Framework SDK command prompt (from the Windows Start menu, select **Programs > Microsoft .NET Framework SDK > SDK Command Prompt**. The command prompt launches.

2 Navigate to the DotNet samples subdirectory.

```
cd %SDKHOME%\samples\DotNet
```

3 Generate the `VimService.cs` file from the WSDL, using `wsdl.exe`:

```
wsdl.exe /n:VimyyApi vim.wsdl vimService.wsdl
```

This command generates `VimService.cs` (the default output file) in the current directory, using the `Vim` or the `Vim25` namespace.

4 Compile `VimService.cs` into a library.

```
csc /t:library /out:VimyyService20xx.dll VimService.cs
```

This command generates a serializer assembly (a DLL).

5 Use the `sgen` tool to pre-generate and compile the XML serializers.

```
sgen /p VimyyService20xx.dll
```

This command outputs the *`Vimyy`*`Service`*`20xx`*`.XmlSerializers.dll` file in the current directory. This DLL file contains pre-generated XML serializer code.

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. To get some visibility into the tool, use the `/k` flag, which causes `sgen` to keep all its temporary generated files, including the source files and command-line option files.

6 Using the "find and replace" capability in Visual Studio or your favorite editor, find the following string in the `VimService.cs` file that you generated in step 3.

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

Replace it 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 `VimService` class. Refer to the MSDN documentation page for more details on this attribute:

http://msdn2.microsoft.com/en-us/library/system.xml.serialization.xmlincludeattribute.aspx

If you are using some other .NET language, modify the replaced string to comment it out according to the syntax of that language.

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

```
[System.Xml.Serialization.XmlSerializerAssemblyAttribute(AssemblyName =
               "VimyyService20xx.XmlSerializers")]
```

You should end up with something like this:

```
// ... Some code here ...
[System.Xml.Serialization.XmlSerializerAssemblyAttribute(AssemblyName =
               "VimyyService20xx.XmlSerializers")]
public partial class VimService : System.Web.Services.Protocols.SoapHttpClientProtocol {
// ... More code here.
```

The Assembly name should be the name of the Serializer DLL generated by the `sgen` tool in step 5.

8. Regenerate the *`Vimyy`*`Service`*`20xx`*`.dll` library with the following command:

   ```
   csc /t:library /out:VimyyService20xx.dll VimService.cs
   ```

9. Ensure your application references the new *`Vimyy`*`Service`*`20xx`*`.dll` library.

Running the Microsoft .NET (C#) Version of SimpleClient

These instructions assume that you have followed all setup instructions detailed in “Setup Instructions for C# Development” on page 22.

To run the SimpleClient application

1. 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:

   ```
   %SDKHOME%\samples\DotNet\cs\SimpleClient\bin\Debug
   ```

2 Run the application, passing to the command line --url, --username, and -password switches along with the values for each. As coded, the SimpleClient application (`SimpleClient.cs`) requires that credentials (user account and password) be passed to the executable, even if the server is configured to support HTTP. You can change the source code yourself and recompile.

If you do not provide all three arguments—server address, account name, and password—the executable generates an error message and comes to a halt. You must pass all three arguments using the command line switches as shown here:

```
simpleclient --url https://sdkpubslab-01.eng.vmware.com/sdk --username useraccount --
               password password
```

The application connects to the server. A "Bad Server Certificate" message displays:

Bad Server Certificate

There were one or more

* The certificate's CN na

3 Click **Yes** to dismiss this message and proceed to the server. After the client connects to the server, the output from server displays in the console (command prompt), as shown in Example 3-1:

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 Setup Issues

If you are unable to successfully run the SimpleClient, check your environment settings and all other setup tasks. Table 3-2 lists some common error messages and provides steps that you can take to resolve.

Table 3-2. Microsoft C# Runtime Error Messages

Symptom	Possible Solution

`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:`
`...`
	Cannot connect to the web service from Microsoft .NET client sample through proxy server. Try a different server on the same subnet as the client.

Appendix: vSphere Server Certificates

This appendix includes these topics:

- "Secure Client-Server Communications" on page 27
- "Simplified Security Setup for Development Environment" on page 27
- "Obtaining Server Certificates" on page 28
- "Modifying Server Configurations to Support HTTP" on page 29

Secure Client-Server Communications

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.

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 28 for more information.

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 29 for more information. Modifying the server configuration to support HTTP access to the vSphere API is recommended for test or development environments only, not for production deployments. VMware recommends using the default protocol, HTTPS, for production deployments.

Simplified Security Setup for Development Environment

Java developers can use the SunFakeTrustSocketFactory class to ignore the server-certificate verification process during the SSL handshake. The Java samples included with the SDK use this technique by accepting an optional command-line argument, `--ignorecert`. The `SunFakeTrustSocketFactory` class is available in the Axis `org.apache.axis.components.net` client package.

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 "Setup Instructions for Java Development" on page 11 for more information.

Use the `--ignorecert` option only for development and testing purposes. Do not use outside a firewall. Be aware that by not verifying the server-certificate 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 (secure sockets layer) 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 server certificates are created automatically during the process of installing VMware products, including ESX, ESXi, and vCenter Server systems. 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 directly (in standalone mode), you must obtain both the vCenter Server certificate and the ESX certificate. On the other hand, 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 two general 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 "Obtaining Certificates by Using the vSphere Client" on page 28.
- 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. See "Obtaining Certificates by Connecting Directly to Server Systems" on page 29 for details.

Obtaining Certificates by Using the vSphere Client

This approach requires you to install the vSphere Client on your development machine. The vSphere Client leverages the native Microsoft credential-handling mechanisms to allow you to accept the certificate and export it as a local file.

To obtain server certificates using vSphere Client

1. Create a directory named `VMware-Certs` (at the root level) for the certificates. Several of the vSphere Web Services SDK batch files assume this path as the location of the keystore and fail if you do not use this path.

   ```
   C:\VMware-Certs
   ```

2. Install the vSphere Client on the development workstation if necessary.
3. Launch the vSphere Client and then navigate to the ESX, ESXi, or vCenter Server web server.

   A Security Warning message box displays regarding the certifying authority for the certificate.

4. Click **View Certificate** to display the Certificate properties page.
5. Click the **Details** tab.
6. Click **Copy to File...** to launch the Certificate Export wizard.
7. Select **DER encoded binary X.509** (the default) and click **Next**.
8. Click **Browse...** and navigate to the `C:\VMware-Certs` subdirectory.
9. Enter a name for the certificate that identifies the server to which it belongs.

   ```
   C:\VMware-Certs\<servername>.cer
   ```

After obtaining the certificate from each target server, follow the other setup steps appropriate for your programing language. For C# developers, see "Setup Instructions for C# Development" on page 22. For Java developers, see "Setup Instructions for Java Development" on page 11.

Obtaining Certificates by Connecting Directly to Server Systems

This approach can be used by developers who have appropriate privileges to directly connect to the target server. These instructions require administrative privileges on the ESX or vCenter Server, and assume that you can access the necessary subdirectory.

To obtain server certificates using secure shell client application

1. From the development workstation, create a directory in which to store certificates of servers that you want to target during development:

   ```
   ~\vmware-certs\
   ```

2. Connect to the ESX system using an SSL client from the development workstation. Remote connections to the ESX service console as root are effectively disabled, so you must connect as another user with privileges on the server to obtain the certificate.

   The server certificate filenames and locations of ESX and vCenter Server are listed in the following table.

Server	Directory Location for Certificate	Certificate
ESX 5.0	/etc/vmware/ssl/	rui.crt
vCenter Server 5.0	C:\Documents and Settings\All Users\Application Data\VMware\VMware VirtualCenter\SSL\	rui.crt

3. Copy the certificates from the server to the certificate subdirectory of the development workstation, using a unique filename for the certificate (assuming you are copying multiple default certificates from multiple ESX systems, for example).

4. Import the server-certificate into the certificate store following the specific instructions for your programming language (Java, C#). See "Setting Up for Java Development" on page 11.

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 a secure Web service that can be accessed using SSL over HTTP (HTTPS). However, for a development environment, you may want to simplify the connection process from a client application by configuring the target servers to support regular (non-SSL) HTTP.

Connections to the Web services port are handled by an Http Reverse Proxy service. The reverse-proxy service handles requests to the API (through the /sdk path) and to the Managed Object Browser (path is /mob). The reverse-proxy service has a configuration file, `proxy.xml`, which can be modified to specify support for HTTP as the protocol for the Web service. (Figure A-1 shows a `proxy.xml` file that has been modified to support HTTP for both the MOB and the SDK.)

To modify the Web proxy service on ESX to support HTTP

1. Log in to the service console 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 may 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`. Alternatively, to completely disable HTTPS, you can set 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-1. Edited proxy.xml File

```
<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>
```

To modify the Web proxy service on ESXi to support HTTP

Because ESXi has no Service Console, you cannot use the same process described in "To modify the Web proxy service on ESX to support HTTP" on page 29. Instead, you must use the appropriate vSphere CLI command to obtain the proxy.xml file from the server, modify the file to support HTTP, and then move the file back to the ESXi system. See the *vSphere CLI Installation and Reference Guide* for more information.

To modify the Web proxy service on vCenter Server 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.

Index

A

Apache Axis, generating stubs **16**
API **7**
AXISHOME **14**

C

C# requirements **21**
CLASSPATH **14**

E

environment variables
  C# **22**
  Java **13**, **14**, **15**
error message, not implemented **10**

H

http protocol, configuring **29**

J

JAVAHOME **14**

M

Microsoft .NET requirements **21**
Microsoft Visual Studio 2005 **22**

N

non-SSL, configuring server for HTTP **29**

O

operations **7**

P

PATH **14**

R

Requirements
  C# development **21**
  Java development **11**

S

samples.jar **9**
SDKHOME **14**
SOAP toolkits **8**
SOAP, defined **7**

T

technical support resources **6**

V

VMKEYSTORE **15**
VSINSTALLDIR **22**
vSphere SDK Package **9**
vSphere SDK package, WSDL files **9**

W

WBEMHOME **15**
WSDL (web services description language), defined **7**
WSDL files **9**
WSDLFILE **22**