Developer's Setup Guide

VMware Infrastructure SDK 2.5

Developer's Setup Guide
Revision: 20071217
Item: VI-ENG-Q407-368

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

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

The VMware Web site also provides the latest product updates.

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

docfeedback@vmware.com

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

Contents

About This Book

This book, the *Developer's Setup Guide*, provides information about setting up your development environment for using the VMware® Infrastructure (VI) SDK to develop client applications.

VMware provides several different SDK products, each intended for different developer communities and target platforms. This guide is intended for developers who are creating applications aimed at managing VMware virtual infrastructure through ESX Server and VirtualCenter Server systems.

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
20070816	Initial publication of VI SDK 2.1 Beta 2.
20071002	Updated for the VI SDK 2.5 RC (release candidate). (Note product name change from SDK 2.1 to SDK 2.5.)
20071129	Added or clarified information about new environment variables (WBEMHOME, VMKEYSTORE) and usage. Added information about command-line argument (--ignorecert). Edited instructions throughout to clarify requirements. Paths changed. Directory structure changed. Added information about `vim.jar`, `vim25.jar`, `samples.jar`, `apputils.jar`: all new names and structures. Edited throughout to account for multiple WSDLs (vim, vim25).
20071217	Added information about configuring ESX Server 3.5 and VirtualCenter 2.5 to chapter 2.

To view the most current version of this guide, go to http://www.vmware.com/support/pubs/sdk_pubs.html.

Intended Audience

This book is intended for anyone who wants to develop applications using the VI SDK. VI SDK developers typically include software developers creating virtual infrastructure management applications using Java or C# (in the Microsoft .NET environment) targeting the Web-services based API available on ESX Server and VirtualCenter Server systems.

Terminology Used in this Guide

To simplify the discussion, this guide uses these terms:

- **Target server**—The VirtualCenter Server or ESX Server systems that are the targets of your client-side code.
- **Development workstation**—The Linux or Microsoft Windows machine that is configured with the Web-services client-side libraries, development environment, and the VI SDK sample code and other artifacts found in the download.

Document Feedback

VMware welcomes your suggestions for improving our documentation. If you have comments, send your feedback to:

docfeedback@vmware.com

Technical Support and Education Resources

The following sections describe the technical support resources available to you. You can access the most current versions of other VMware manuals by going to:

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

Online Support

You can submit questions or post comments to the Developer Community: SDKs and APIs forum, which is monitored by VMware technical support and product teams. You can access the forum at:

http://communities.vmware.com/community/developer

Support Offerings

Find out how VMware support offerings can help meet your business needs. Go to http://www.vmware.com/support/services.

VMware Education Services

VMware courses offer extensive hands-on labs, case study examples, and course materials designed to be used as on-the-job reference tools. For more information about VMware Education Services, go to http://mylearn1.vmware.com/mgrreg/index.cfm.

Introducing the VMware Infrastructure SDK 1

The VMware Infrastructure SDK (VI SDK) facilitates development of client applications that target the VI API.

VMware provides several different SDK products, each intended for different developer communities and target platforms. The VI SDK is for developers who want to create client applications that can manage, monitor, and maintain VMware virtual infrastructure, as deployed on ESX Server and VirtualCenter Server systems. This chapter provides an overview of VI SDK packaging, requirements, and functionality. It includes these topics:

What Is the VMware Infrastructure API?

The VMware Infrastructure API (VI API) provides a complete set of language-neutral interfaces to the VMware virtual infrastructure management framework. In much the same way that JMX (Java Management Extension) provides an infrastructure for instrumenting, managing, and monitoring Java applications, this framework provides an infrastructure for basically doing the same for VMware Infrastructure 3 components (for example, virtual machines, host systems) and subsystems (such as performance manager). The virtual infrastructure management framework is accessed by external clients using the VI API.

VI API is implemented as industry-standard Web services, hosted on VirtualCenter Server and ESX Server systems. The VI API complies with the Web Services Interoperability Organization (WS-I) Basic Profile 1.0, which includes XML Schema 1.0, SOAP 1.1, WSDL 1.1. For more information about the WS-I Basic Profile 1.0, visit the Web Services Interoperability Organization (WS-I) Web site at:

http://www.ws-i.org

The Web service provides all the operations necessary, including life-cycle operations, to monitor and manage virtual infrastructure components—compute resources, virtual machines, networks, storage, and the like.

Knowledge Required for Using the VI SDK

To be successful with the VI SDK, you should be familiar with basic Web services concepts, in general, and understand specific programming and debugging processes and techniques. Some concepts you most need to understand are these:

- Web services technology provides operations (same basic concept as "methods" in other programming languages). Using the VI 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 a WSDL (Web Services Description Language) file. The WSDL file is 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 sending SOAP (Simple Object Access Protocol)-formatted messages. (SOAP is an XML format and is programming-language neutral. One of the jobs of the client-side Web services tools is formatting (transparent to you, the developer) the SOAP messages from the programming language that you use.
- 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 the Web server (on ESX Server and VirtualCenter Server) can be configured to support HTTP. (See "Modifying the Server Configuration to Support HTTP" on page 14 for details.)

In addition, you should be familiar with the operation of VMware VirtualCenter and VMware ESX Server—the targets for your coding efforts. See the VMware Infrastructure 3 Documentation page for more information:

http://www.vmware.com/support/pubs/vi_pubs.html

Programming Languages Supported by the VI SDK

Technically speaking, since the VI 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, it's best to use Java or C#, since SOAP toolkits are readily available for these languages, and sample client applications for these two development languages are available as part of the VI SDK package. (See "What's in the VI SDK Package?" on page 9 for additional packaging details, and for some caveats regarding Java samples and specific versions of the JDK and Axis.)

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

	Java	**C#**
Development environment or framework	J2SE 5.0 (aka, J2SE 1.5—J2SE 1.5_0_08 or subsequent version recommended)	Microsoft Visual Studio 2005 Microsoft Visual C#
Web-services-client application development toolset ("SOAP toolkit")	Apache Axis 1.4	Microsoft .NET Framework 2.0 (recommended)

Note that Perl scripters can access the VI API through the VI Perl Toolkit. Designed to minimize the client-side development overhead (mapping datatypes and generating client proxy code, for example) associated with distributed-object computing, the VI Perl Toolkit includes a runtime component that handles language bindings, remote instantiation, and other Web-services-client programming infrastructure issues.

You can learn more about the VI Perl Toolkit at:

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

Types of Applications That Can Be Built Using the VI SDK

The types of applications that you can develop using the VI SDK include system administration, provisioning, and monitoring applications for VMware Infrastructure 3 systems. As examples of client applications, note that the VMware Infrastructure Client (VI Client) application and VMware Virtual Infrastructure Web Access interface have both been created using VI SDK components. The VI Client is a traditional Windows client application; Web Access is a browser plug-in that is available via the Web server port on ESX Server and VirtualCenter Server systems.

With the VI SDK, you can go beyond the capabilities of the VI Client and Web Access tool, by automating or otherwise streamlining many administration, provisioning, or monitoring tasks associated with virtual infrastructure management and operations.

For example, you can create a client application that creates multiple virtual machines across several different ESX Server systems automatically, perhaps when initiated from a Web-server submission form.

Here are some examples of the operational tasks that you can automate using the VI API:

- Create, configure, power-cycle, or suspend virtual machines explicitly, or by using cloning 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 host systems (ESX Server) and VirtualCenter Server.
- Capture the state of a virtual machine (to a snapshot) and restore the state of a virtual machine from a snapshot.
- 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) sub-systems.

This list is not comprehensive. Also, note that 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.

What's in the VI SDK Package?

The VI SDK is a zip file that includes:

- Two different sets (vim, vim25) of WSDL files (`vimService.wsdl`, `vim.wsdl`) that define the API available on an ESX Server and VirtualCenter Server Web service. The SDK package also includes pre-compiled client-side libraries (`vim.jar`, `vim25.jar`) available for test purposes) that have been generated from the WSDL.
- Sample code demonstrating common use cases associated with programmatically managing virtual infrastructure. The sample code includes compiled and ready-to-run Java class files, as well as Java and C# source code files. (For C# developers, the Microsoft Visual Studio project files (`.sln`) have been included.)

  NOTE The pre-compiled Java samples (`samples.jar`) have been compiled using JDK 1.5 from stubs generated by Apache Axis 1.4, and will work only with these specific versions of Java and Axis. To use different versions of Java and Axis, you must rebuild the samples (using the build script). Note that JDK 1.5.0_08 (or subsequent version) is recommended, to avoid a performance issue. See Knowledge Base article 2983901, "Performance Issue (for VI SDK 2.0) with JDK Versions Prior to JDK 1.5.0_08," for additional details.

- Batch files and shell scripts (build.bat, build.sh; Build2005.cmd) 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 (VI API Reference Guide) that provides language-neutral descriptive information (object type definitions, properties, and method signatures, for example) about the VMware Infrastructure API (VI API) and the server-side object model.
- Access to technical publications, including:
  - *Developer's Setup Guide* (this book), which helps you setup your development environment and run sample applications using either Java or C#;
  - *Programming Guide*, which provides conceptual and prescriptive information about how to develop client applications using the VI SDK.

NOTE This *Installation Guide*, *Programming Guide*, and other technical publications, including readmes for the Java samples and C# samples, are available online only, as a zip file download that can be unpacked into the SDK directory.

Complete information about setting up the environment, and about generating, compiling, and running applications is included in Chapter 3, "Setting Up for Java Development," on page 19, and in Chapter 4, "Setting Up for Microsoft C# Development," on page 27.

VI SDK Versions and VMware Infrastructure 3 Product Compatibility

VMware has released several VI SDK products to support various versions of the VMware Infrastructure product family. The VI SDK 2.5 can be used with:

- ESX Server 3.5, ESX Server 3i, ESX Server 3.0.x
- VirtualCenter 2.5, VirtualCenter Server 2.0.x

The VMware Infrastructure API is a Web service that runs on ESX Server and VirtualCenter Server. The API exposed is basically the same in both products. However, the VirtualCenter Server provides several capabilities not available through the ESX Server web service, such as:

- Collecting historical performance data
- Optimizing resources (including managing distributed resources)
- Enabling migration from one ESX Server to another via VMotion
- Providing distributed resource management, including recovery, across all ESX Servers under its control.

If you attempt to invoke an operation on an ESX Server that is implemented only on the VirtualCenter Server, a "not implemented" error message will be raised.

Preliminary Server-Certificate Setup 2

Before you can get started with the VI SDK, you must have access to one or more ESX Server or VirtualCenter Server systems. The VMware Infrastructure API is available as a secure Web service running on both of these platforms. "Secure" Web service means that, by default, both ESX Server and VirtualCenter Server are configured for HTTPS and support SSL to encrypt communications.

This chapter includes these topics:

NOTE Java developers can use SunFakeTrustSocketFactory class (available in the Axis client libraries (org.apache.axis.components.net package) to ignore server-certificate verification. The Java samples included with the SDK use this technique via an optional command-line argument (--ignorecert). If you plan to use the --ignorecert option (or use this technique in your own code), you can skip this chapter entirely.

WARNING Use the --ignorecert option for development and testing purposes only. 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.

Alternatively, you can modify the default server configuration so that it supports non-SSL (regular HTTP) communications, in which case you do not need to import the server certificates on the client development workstation.

CAUTION VMware recommends that secure HTTP (HTTPS; the default configuration) be used for production deployments. Modifying the server configuration to support HTTP access to the VI API is recommended for **test or development environments only, not for production deployments.**

If you don't obtain the certificates (or setup the server for HTTP), you will not be able to connect to the server from client applications, including the sample applications. However, for development environments, Java developers can by-pass the server-certificate verification by using the --ignorecert argument. See "Setup Instructions for Java Development" on page 19 for more information.

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) 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 Server and VirtualCenter Server systems. These default certificates are digitally signed with the name of the host system, so they are sometimes referred to as "self-signed certificates." Because these certificates are self-signed (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 (the "target servers") and store it locally. You must obtain server certificates from each target server.

For example, if you are creating a client application that will run against the VirtualCenter Server and an ESX Server directly (in standalone mode), you must obtain both the VirtualCenter Server certificate and the ESX Server certificate. On the other hand, if your application will run solely against the VirtualCenter Server (that may be managing any number of ESX Server systems), you must obtain the certificate from the VirtualCenter Server only.

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 Internet Explorer browser (from the development workstation) to quickly browse to each ESX Server or VirtualCenter Server and accept the certificate into the local cache (as detailed in "Obtaining Certificates by Using Microsoft Internet Explorer's Certificate Cache," below).
- 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 Server or VirtualCenter Server and copy the certificates directly from the server to the development platform. See "Obtaining Certificates by Connecting Directly to Server Systems" on page 14 for details.

Obtaining Certificates by Using Microsoft Internet Explorer's Certificate Cache

This approach works with Microsoft Internet Explorer browser only. Mozilla Firefox handles certificates differently, so these instructions can not be applied to the Firefox (or other, non-Internet Explorer) browsers, nor to non-Windows platforms.

To obtain server certificates using Microsoft Internet Explorer (for Windows development platform)

1. From the development workstation, open Internet Explorer browser.
2. Navigate to the ESX Server or VirtualCenter Server web server using the HTTPS protocol:

   https://*servername*

   A Security Alert message displays a warning regarding the certificate's certifying authority.

   NOTE Message text will vary, depending on the version of Microsoft Internet Explorer that you use. The warnings are raised because the default certificates are self-signed. If your site has replaced the default certificates, you will likely not see the warning messages. However, you will still need to obtain the certificates and install on the local development workstation as discussed below.

3. Click **View Certificate** to open the Certificate properties page.
4. Click **Install Certificate** to launch the Certificate Import Wizard. Keep the default setting ("Automatically select the certificate store based on the type certificate") and click **Next** to continue.
5. Click **Next** to continue installing the certificate.
6. Click **Finish**. A Security Warning message displays regarding the certificate's certifying authority.

7 Click **Yes** to continue with the certificate installation. A Certificate Import Wizard "success" message displays.

8 Click **OK** to dismiss the success message. The Certificate properties page becomes active again.

9 Click **OK** in the Certificate dialog box to continue to the server. The initial Security Alert message presented in step 2 above becomes active again (returns to focus in the window).

  Click **Yes** in the Security Alert message to continue with the original HTTPS request for the server (back at step 2, where all this began). The server (VMware ESX Server 3, VMware VirtualCenter 2) Welcome page displays.

  The Certificate has now been installed in Internet Explorer's certificate cache.

10 Repeat the process for each ESX Server and VirtualCenter Server that you will be using with the VI SDK.

Once you've obtained all the certificates you need (one for each target server), next steps depend on the programing language you'll be using:

- For **C# (Microsoft .NET) developers**, at this point, preliminary setup tasks are complete: the .NET infrastructure uses the certificates you've imported. You can continue setting up your development workstation by following the instructions in Running the SimpleClient Sample Application to Validate Setup.
- For **Java developers**, you must export the certificates from the Internet Explorer cache to a local directory. Minimize the Internet Explorer browser window (or simply move it aside for a moment), and export the certificates as detailed in the next several steps.

To export the cached certificates to a local directory (for Java development on Windows)

1 Create a directory named `VMware-Certs` (at the root level) for the certificates; several of the VI SDK batch files assume this path as the location of the keystore, and will fail if you don't use this path:

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

2 From the Internet Explorer **Tools** menu, select **Internet Options** to open the Internet Options properties page.

3 Click the Content tab to activate content advisor, certificates store on the properties page.

4 Click **Certificates** to open the Certificate manager.

5 Click the Trusted Root Certificate Authorities tab to display the list of trusted certificates. This list should contain a certificate for each of the target servers selected in steps 2 through 8, above.

6 Scroll through the list of certificates to find the certificates. For ESX Server systems, the certificate name matches the DNS name of the server. For VirtualCenter Server systems, the certificate name is VMware.

7 For each target server:

   a Click the certificate to select it.

   b Click **Export...** to launch the Certificate Export Wizard.

   c Click **Next** to continue. The Export File Format dialog displays.

   d Keep the defaults ("DER encoded binary X.509 (.CER)") and click **Next** to continue. The File To Export dialog displays, enabling you to enter a unique name for the certificate.

   e Choose a filename and enter it, along with the complete path to the directory created in step 9:

   `C:\VMware-Certs\`*`servername`*`.cer`

   NOTE If you don't enter the complete path, the certificate gets stored in your `Documents and Settings` folder. On Windows development workstations, the batch files included with the VI SDK have been defined to use the `c:\VMware-Certs` directory.

   f Click **Next** to continue with the export. A Completing the Certificate Export Wizard page displays, summarizing the information about the certificate.

   g Click **Finish** to complete the export. A Certificate Export Wizard "success" message displays.

   h Click **OK** to dismiss the success message.

8 When you have exported all certificates for the servers you want to target, click **Close** to exit the certificates export dialog.

9 Click **Cancel** to close the Internet Options properties page.

Continue setting up your Java development environment by following the steps in "Requirements" on page 19.

Obtaining Certificates by Connecting Directly to Server Systems

This approach can be used by developers using Linux for development workstation, or anyone who has appropriate privileges to directly connect to the target server.

To obtain server certificates using secure shell client application (for Linux development platform)

These instructions require administrative privileges on the VirtualCenter Server, and assume that you can access the necessary sub-directory (see Table 2-1).

1 From the development workstation, create a directory in which to store certificates of ESX Servers or VirtualCenter Servers that you'll be using during development:

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

2 Connect to the ESX Server using an SSL client from the development workstation. (Remember that remote connections to the ESX Server service console as root are effectively disabled: you must connect as another user with privileges on the server, to obtain the certificate.)

   The server certificate filenames and locations on various versions of the ESX Server and VirtualCenter Server are listed in Table 2-2.

   Table 2-1. Directory Locations and Certificate Filenames

Server	Directory Location for Certificate	Certificate
ESX Server 3.x	`/etc/vmware/ssl/`	rui.crt
VirtualCenter Server 2.x	`C:\Documents and Settings\All Users\Application Data\VMware\VMware VirtualCenter\SSL\`	rui.crt

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

4 Import the server-certificate into the certificate store following the specific instructions for your programming language (Java, C#). The details are covered in the next chapter, "Setting Up for Java Development."

Modifying the Server Configuration to Support HTTP

Both ESX Server and VirtualCenter Server support the VI 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), as in:

https://<yourservername>.fqdn.com/sdk

For a development environment, you may prefer to configure target servers (following the instructions in this section) to support regular (non-SSL) HTTP, to simplify your client-side setup, development, and testing. Depending on the version of ESX Server and VirtualCenter Server, the steps vary.

- Modifying ESX Server 3.5 or VirtualCenter 2.5 Configurations
- Modifying ESX Server 3.0.x and VirtualCenter 2.0.x Configuration

Modifying ESX Server 3.5 or VirtualCenter 2.5 Configurations

Both ESX Server 3.5 and VirtualCenter 2.5 handle connections to Web-based services, including the API and the Managed Object Browser (MOB), through an Http Reverse Proxy service. This service has its own configuration file, `proxy.xml`, that can be modified to support HTTP connections to the ESX Server. See the *ESX Server 3 Configuration Guide* for complete details. Figure 2-1 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 Server 3.5 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`.

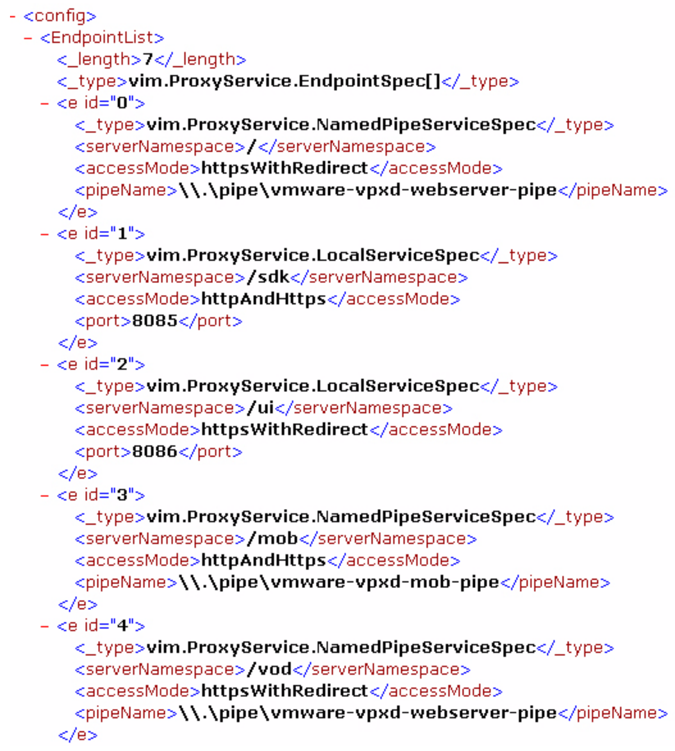
6 Optionally, change the setting for the MOB as well.

7 Save your settings and close the file.

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

```
service mgmt-vmware restart
```

Figure 2-1. 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 VirtualCenter Server 2.5 to support HTTP

1. Log in to the VirtualCenter 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 VirtualCenter Service from a command line or from the Windows Services control panel.

Modifying ESX Server 3.0.x and VirtualCenter 2.0.x Configuration

For ESX Server 3.0.x and VirtualCenter 2.0.x, the configuration files are standard Web server configuration files that you (or your system administrator) can manually edit using any text editor to:

- Add a deployment descriptor tag for the server and its port to the HTTP section of the configuration file
- Remove the redirection tag for the server from the HTTPS section of the configuration file

Table 2-2. Configuration Filenames and Locations

Server Platform	Path	Filename
VirtualCenter Server (2.0.x)	`c:\Documents and Settings\All Users\Application Data\Vmware\VMware VirtualCenter`	`vpxd.cfg`
ESX Server (3.0.x)	`/etc/vmware/hostd/`	`config.xml`

The next subsections provide detailed instructions for making these changes on VirtualCenter and ESX Server, respectively.

Modifying a VirtualCenter Server Configuration File

The VirtualCenter Server configuration file is located several directories deep in the Windows Documents and Settings folder, in:

```
c:\Documents and Settings\All Users\Application Data\Vmware\VMware VirtualCenter\vpxd.cfg
```

These instructions require access to the VirtualCenter Server, and that you can access the file and perform the necessary edit.

To edit a VirtualCenter Server configuration file (vpxd.cfg)

1. Log on to the Windows machine running VirtualCenter Server using the same credentials as used to install VirtualCenter Server.
2. Navigate to the directory where `vpxd.cfg` is located:

   ```
   cd c:\Documents and Settings\All Users\Application Data\Vmware\VMware VirtualCenter
   ```
3. Use a text editor, such as WordPad, to open the configuration file. (Note that the file may contain commented instructions about how to modify to support HTTP. If it does, follow the instructions in the file, rather than these.)
4. Search through the `vpxd.cfg` file to locate the `<proxyDatabase>` tag within the `<http>` tag.
5. Within the `<proxyDatabase>` tag, add the following text (or remove the comment code to enable the setting, if the deployment descriptor tags already exist in the file):

   ```
   <server id="1">
       <namespace> /sdk </namespace>
       <host> localhost </host>
   ```

```
    <port> -2 </port>
</server>
```

6 In the <proxyDatabase> tag (within the <http> definition), delete the redirection tag associated with the /sdk, which may look like the following:

```
<redirect id="2"> /sdk </redirect>
```

7 Save changes and close the file.

8 Restart the VMware VirtualCenter Server service using the Services control panel:

 a From the Windows Start menu, select **Settings > Control Panel > Administrative Tools > Services**. The Services control panel opens, displaying the full list of services installed on the server.

 b Scroll through the list of services to find the VMware VirtualCenter Server.

 c Click the VMware VirtualCenter Server service to highlight it, and then click **Restart**. The service restarts and the new server settings take effect.

The server should now be accessible via HTTP connection. To test, open a browser and connect to the VirtualCenter Server using HTTP.

Modifying an ESX Server Configuration File

The ESX Server configuration file is located in:

```
/etc/vmware/hostd/config.xml
```

The instructions below require administrative (root) privileges on the ESX Server system. Note that, by default, remote connections (SSH, for example) to ESX Server service console are disabled. Typically, another user account is used to connect to the server and then **`su - root`** to perform tasks at the service console.

To edit an ESX Server configuration file (config.xml)

1 Connect to the ESX Server and navigate to the `/etc/vmware/hostd` directory.

2 Use a text editor, such as vi, to open the `config.xml` file. You'll see several XML element tags defining the Web server subsystems. The changes you need to make are all in the <proxySvc> element.

3 Find the section of the file in the <proxySvc> element, under the <https> tag. In this section, you'll see tags identifying the <port> and the <proxyDatabase>.

4 Under <proxyDatabase>, locate the tag sequence for the VI API server port (/sdk). The tag sequence may look like this example:

```
<server id="1">
    <namespace> /sdk </namespace>
        <host> localhost </host>
    <port> 8085 </port>
</server>
```

5 Copy the entire set of tags from <server id="*n*"> through its matching closing tag </server>, and paste within the <proxyDatabase> element (in the <http> element section), as shown in this example:

Example 2-1. Excerpt of Server config.xml that Supports HTTP

```
        ...
        <http><port>80</port>
        <proxyDatabase>
            <server id="0"><namespace> / </namespace>
                <host> localhost </host><port> 9080 </port>
            </server>
            <server id="1">
                <namespace> /sdk </namespace>
                <host> localhost </host>
                <port> 8085 </port>
            </server>
                <redirect id="0"> /ui </redirect>
                <redirect id="1"> /mob </redirect>
        </proxyDatabase>
        </http>
        <https><port>443</port>
        <proxyDatabase>
            <server id="0">
                <namespace> / </namespace>
                <host> localhost </host>
                <port> 9080 </port>
            </server>
            <server id="2">
                <namespace> /ui </namespace>
                <host> localhost </host>
                <port> 8080 </port>
            </server>
            <server id="3">
            <namespace>/mob</namespace>
            <host>localhost</host>
            <port>8087</port>
            </server>
        </proxyDatabase>
        </https>
...
```

6 In the <proxyDatabase> tag (within the <http> definition), delete the redirection tag associated with the /sdk, which may look like the following:

```
<redirect id=”2”> /sdk </redirect>
```

7 Return to the <proxyDatabase> tag within the <https> definition and delete the content that you copied in step 4 and 5 above.

8 Save changes and close the file.

9 Restart the Web service:

```
service mgmt-vmware restart
```

The server should now be accessible via HTTP connection.

Setting Up for Java Development 3

This chapter provides detailed setup information for developers using Java. It includes these topics:

Requirements

Developing Java Web-services client applications using the VMware Infrastructure SDK requires the Java SDK and a Java Web services development toolset, specifically:

- **Java 2, Standard Edition, version 5.0** (J2SE 1.5.x) or J2SE 1.4.x. **VMware recommends using J2SE 1.5.0_08 (or later)**, for two reasons:
  - Improved performance: Using Apache Axis client-libraries with versions of the JDK prior to JDK 1.5_0_08 yields slow performance (due to a socket-creation issue).
  - Samples support: Some of the samples use features introduced in JDK 5 (JDK 1.5), and so cannot be used with prior versions of the JDK (1.4.2, for example).
- **Apache Axis 1.4**: Axis is an open source project of the Apache Web services project. It's a SOAP (Simple Object Access Protocol) implementation that can be deployed to a Tomcat server. The client-side components include various XML processing and other libraries needed for Web-services client development. You can obtain Axis from the Apache Web-services project site:

  http://ws.apache.org/axis/

Other client-side tools and libraries, such as IBM's WebSphere and several open source implementations, can be used with the VI SDK. However, only Axis client libraries have been tested with this Developer's Setup Guide.

The samples archive (samples.jar) includes all VI SDK samples, compiled using J2SE 5 (JDK 1.5.0). The client-side proxy code (stub classes) were generated using Apache Axis 1.4 libraries.

Setup Instructions for Java Development

Specific setup instructions will vary, depending on whether your development workstation already meets some or all of the requirements, and whether you plan to use the provided samples. The VI SDK package includes vim.jar, vim25.jar, apputils.jar, and samples.jar files that were generated using Apache Axis 1.4 libraries and compiled using Java JDK 1.5— you can use these libraries, as is, without generating new stubs and re-compiling—if you are using these same versions of the JDK and Axis.

To setup a development workstation to use Java

1 Create directories for each of the components that you will need 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 3-1 lists some recommended naming conventions:

Table 3-1. Recommended Directory Structures

	Windows	**Linux**
VI 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 or later 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 Infrastructure SDK (VI SDK) package from VMware Web site:

http://www.vmware.com/download/sdk/

5 Unpack the various components into sub-directories created in step 1 above, using the provided installer if appropriate. (The J2SE uses an installation wizard; the Axis zip file and the SDK simply unpack into a selected target sub-directory. Be sure to unpack with "Use folder names" selected, to maintain the organizational structure.)

6 Import server-certificates (assuming that the target servers are configured for HTTPS) and create a `vmware.keystore` (using the Java keytool utility). See "Importing Server-Certificates into the Java Keystore" on page 22 for details.

a If you want to ignore server-certificate verification for any of the sample Java applications, you can pass the --ignorecert argument at runtime, in which case you can skip this step.

7 Configure environment settings as detailed in "Setting Environment Variables." To run any of the Java sample applications using the provided run script (run.bat, run.sh), you need set only AXISHOME, JAVAHOME, VMKEYSTORE, and WBEMHOME.

8 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 23 for details.

NOTE If you are using Axis 1.4 and JDK 1.5, you can skip this step—simply add `vim.jar`, `vim25.jar`, `apputils.jar`, and `samples.jar` to your system CLASSPATH, or simply set the AXISHOME, JAVAHOME, VMKEYSTORE, and WBEMHOME and use the run.bat (or run.sh).

9 Run the Java version of SimpleClient to test your setup. (See Running the SimpleClient Sample Application to Validate Setup for details).

Batch Files (and Shell Scripts) for Building and Running Samples

The VI SDK includes several batch files (and shell scripts, for Linux) that facilitate building and running the sample applications.

NOTE Unless you are using a JDK other than 1.5 and Axis libraries other than 1.4, you do not need to rebuild the samples.

Note that some of the batch files are for internal use, by other batch files, and are not intended to be used by developers. For example, `build.bat` (and `build.sh`) call the `lcp.bat` (`lcp.sh`) and `clean.bat` (`clean.sh`) scripts. Information about these is included for completeness only: if you modify the batch files for any reason, be aware of the dependencies among them.

Table 3-2. Batch Files and Shell Scripts for Java [..\Axis\java\ Sub-Directory]

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. Result of successful execution of `build.bat` is creation of `vim.jar`, `vim25..jar`, `apputils.jar`, and `samples.jar` files. Use the -w to recompile without regenerating stubs.
`lcp.bat` `lcp.sh`	An internal batch file that sets the local classpath on the workstation. The `lcp.bat` file is called by `build.bat` and by `run.bat`.	Although you can use this script to set local classpath, it exists for internal use by other scripts.
`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 passes as a parameter.	Use this script to run any Java sample applications.
`clean.bat`	An internal batch file called by build.bat, that removes any existing artifacts prior to building the samples. Specifically, deletes all Java class files in the samples packages and samples.jar file.	Deletes all generated source code files. Not intended for direct use (this script is called by other scripts).

Setting Environment Variables

The specifics of the environment variables you must set depends on whether you want to use the provided scripts (build.bat, build.sh; run.bat, run.sh), or not. To use the provided scripts, you need set only four environment variables—AXISHOME, JAVAHOME, VMKEYSTORE, and WBEMHOME. See Table 3-3 for details.

If you don't want to use the scripts—for example, if you're using a Java IDE, such as Eclipse—you should set CLASSPATH and PATH environment variables (see Table 3-3) so that they include the Axis libraries and the J2SE libraries. (If these variables do not exist already on your development workstation, you should create them.)

Table 3-3 lists environment variable names, directories, and filenames be 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, and WBEMHOME, as defined in Table 3-3.

- For Microsoft Windows development workstations, you should set System (rather than User) environment variables.
- For Linux development workstations, you should 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 3-3.

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

Table 3-3. Environment Variable Names and Required JAR and Other Files

Variable Name	Setting Should 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 you've 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	Paths to the binary root directories for both the Java JDK and the Java runtime (JRE). For example:
	`C:\jdk1.5_0_08\bin`
	`C\jre1.5_0_08\bin`

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 should add:
	`%AXISHOME%\bin`
	`%JAVAHOME%\bin`

SDKHOME	Path the top-level directory of the unpacked SDK download. For example: `C:\devprojects\visdk21\SDK`
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 don't need to import the certificate or create the actual keystore, though, if you use the --ignorecert argument).
WBEMHOME	Path to the WBEM (Web Based Enterprise Management) Java archive (`wbem.jar`). `WBEMHOME=%SDKHOME%\SDK\samples\Axis\java\lib\wbem.jar`

Importing Server-Certificates into the Java Keystore

This step is required **only** if:

- The target servers are configured for HTTPS (the default configuration), and
- You will not be using 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 the Server Configuration to Support HTTP" on page 14. However, using HTTP is not recommended for production environments.

These instructions assume that JAVAHOME has been set and that it has been added to the PATH.

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 the subsequent steps below.)

Table 3-4. Java Keystore Location and Filename

Windows	Linux
`C:\VMware-Certs\vmware.keystore`	`~/vmware-certs\vmware.keystore`

3 Navigate to the directory. For example, on Windows:

```
cd %VMKEYSTORE%
```

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
```

You will be prompted to create a password for the keystore you are creating:

```
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 should be 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:

```
Certificate was added to keystore
```

7 Repeat this process for each target server.

Generating Stubs and Compiling Classes

The VI SDK includes vim.jar, vim25.jar, samples.jar, and apputils.jar files created using Axis 1.4 and JDK 1.5. If your development environment includes these same Axis and Java versions, you can use these pre-compiled libraries, as is, without regenerating or compiling. Skip this section and try running the SimpleClient by following the instructions in "Running the SimpleClient Sample Application to Validate Setup" on page 24.

If you are using versions of Axis and Java other than Axis 1.4 and JDK 1.5, you must regenerate the client-side proxy code (stubs) and recompile into Java archive files. The build.bat (or build.sh) script included with the SDK performs all necessary tasks for you.

These instructions require setting the AXISHOME, JAVAHOME, and WBEMHOME environment variables (see "Setting Environment Variables" on page 21).

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

1 Open a command prompt.

2 Navigate to the sub-directory containing the `build.bat` and `build.sh` files:

```
cd %SDKHOME%\samples\Axis\java
```

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

```
build
```

Shortly, you'll see output at the console, starting with "Generating stubs from wsdl." In a few minutes, the entire process should be complete: the word "Done" displays at the command prompt, as shown in Example 3-1.

Example 3-1. Successful Stub Generation and Compilation Using build.bat Script

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

When the process completes successfully, you will see that the `vim.jar`, `vim25.jar`, `apputils.jar`, and `samples.jar` files reflect the current date and time.

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.

These scripts require the environment variables AXISHOME, JAVAHOME, VMKEYSTORE, and WBEMHOME (see Table 3-3).

To run a sample application using the run.bat (or run.sh) script

To run any of the Java samples using the run.bat (or run.sh) script, simply pass the name of the Java class (the full package name), as in:

```
run com.vmware.samples.simpleclient.SimpleClient https://yourFQDNservername/sdk <username>
<password> [--ignorecert ignorecert]
```

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

2 Navigate to the directory containing the Java samples:

```
cd %SDKHOME%\samples\Axis\java\com\vmware\samples
```

Example 3-2. 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
```

If you have your development environment configured with the proper CLASSPATH and PATH environment variables, you can run the applications directly from the command prompt using the Java runtime.

To run the precompiled SimpleClient from the command prompt

1. Open a Windows command prompt (or shell prompt on Linux).
2. Navigate to the directory containing the class files for the samples:

   ```
   cd %SDKHOME%\samples\Axis\java\com\vmware\vimsample.classes
   ```

3. Invoke the Java runtime, providing the full package name of the SimpleClient, along with server URN, credentials, 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> <server-url> <username> <password> [--ignorecert ignorecert]
   ```

   For example:

   ```
   java -Djavax.net.ssl.trustStore=%VMKEYSTORE% com.vmware.vimsample.simpleclient.SimpleClient
   https://sdkpubslab-02.eng.vmware.com/sdk root password
   ```

   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.vimsample.simpleclient.SimpleClient http://sdkpubslab-02.eng.vmware.com/sdk
   username password

Troubleshooting Setup Issues

If you are unable to successfully run the SimpleClient, check your environment settings and all other setup tasks. Table 3-5 lists some common error messages and provides steps that you can take to resolve them.

Table 3-5. Java Runtime Error Messages

Symptom	Possible Solution
`Exception in thread "main" java.lang.NoClassDefFoundError: com/vmware/vim/ManagedObjectReference at com.vmware.vimsample.simpleclient.SimpleClient .createServiceRef(SimpleClient.java:32) at com.vmware.vimsample.simpleclient.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.read FromSocket(HTTPSender.java:744)`
	■ Verify that the server-certificate has been imported into the local keystore, and then execute the command using HTTPS and provide the credentials (user account, password) for the server.

Setting Up for Microsoft C# Development 4

This chapter provides information for developers using Microsoft C# programming language. It includes these topics:

Requirements

The VI SDK includes C# (`.cs`) source files and Microsoft Visual Studio project files (solutions, or `.sln`) for both Microsoft Visual Studio 2003 and Microsoft Visual Studio 2005. Web services client application development for C# requires:

- Development environment for C#, such as Microsoft Visual C#, Microsoft Visual Studio 2003, or Microsoft Visual Studio 2005.
- Microsoft .NET Framework, specifically Microsoft .NET Framework 2.0 (which is included with Microsoft Visual Studio 2005).
- Microsoft .NET Framework 2.0 Software Development Kit. Depending on the specific version of the Microsoft Visual Studio and Microsoft .NET Framework that you use, you may need to also specifically install the Microsoft .NET Framework 2.0 software development kit, which includes the command-line C# compiler (csc.exe).

CAUTION VMware strongly encourages developers to **use Microsoft Visual Studio 2005 and Microsoft .NET 2.0 only** (not legacy versions, such as Visual Studio 2003 and Microsoft .NET Framework 1.1) with the VI SDK. There is a serious performance issue that occurs when using Visual Studio 2003 and the Microsoft .NET Framework 1.1 with the VI SDK.

Environment Variable Settings

The VI SDK 2.5 includes two versions of the WSDL files:

- WSDL files that support vim25 (for use with ESX Server 3.5 and VirtualCenter 2.5)
- WSDL files that support vim (for use with ESX Server 3.0.1 and VirtualCenter 2.0)

The batch files used to build samples require access to the appropriate WSDL file. If you intend to use any of the provided batch files, you must create an environment variable for WSDLFILE and set to the sub-directory of the WSDL file you intend to use.

The VI SDK download contains batch files for setting up the samples for Microsoft Visual Studio 2003 and Microsoft Visual Studio 2005. It also includes the project files (`.sln`, solution) files).

Table 4-1. Batch Files or Command Scripts for Microsoft C# .NET (..\DotNet)

Filename	Description	Usage Note
`Build2003.cmd`	Batch file that generates client-side stubs for all sample applications for the Microsoft Visual Studio 2003 .NET development environment.	VMware recommends against using Microsoft Visual Studio 2003 (and .NET Framework 1.1).
`Build2005.cmd`	Batch file that generates client-side stubs for all sample applications for the Microsoft Visual Studio 2005 development environment.	Use the Microsoft .NET Framework SDK v2.0 Command Prompt for running Build2005.cmd to ensure correct path settings.

In addition, if your Microsoft software setup varies from the default paths, you must create a VSINSTALLDIR environment variable, as defined in Table 4-2.

1 Create an environment variable for WSDLFILE, as defined in Table 4-2.

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 4-2.

Table 4-2. Environment Variable Names Required for Build Scripts

Variable Name	Description	Usage Note

VSINSTALLDIR	Location of Microsoft's ...`\Common7` and ...`\SDK` sub-directories.	Required only if your development setup varies from Microsoft default installation paths.
		Use quotation marks around sub-directory pathnames that include spaces. For example:
		`"C:\devstuff\Microsoft Visual Studio 8\Common7"`
		`"C:\devstuff\Microsoft Visual Studio 8\SDK"`

WSDLFILE	Identifies the location of the WSDL appropriate for the target server.	Set to `%SDKHOME%\wsdl\vim` to use VI API 2.0 (targeting ESX Server 3.0.x, VirtualCenter 2.0.x). Set to `%SDKHOME%\wsdl\vim25` for VI API 2.5 (targeting ESX Server 3.5, VirtualCenter 2.5).

Setup Instructions for C# Development

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

NOTE The batch files (`build2003.cmd`, `build2005.cmd`) assume that your setup uses default paths to Microsoft installed products. See "Environment Variable Settings" on page 27 for information about setting an additional environment variable to handle discrepancies in your setup, if necessary.

To set up a development workstation to use C#

1 Install the Microsoft Visual programming environment, such as Microsoft Visual Studio 2005, Microsoft Visual Studio .NET 2003, or Microsoft Visual C#.

VMware recommends using Microsoft Visual Studio 2005, which includes the required .NET Framework 2.0 and improved versions of Web-services-client tools. For more information, visit:

http://msdn.microsoft.com

2 Obtain the Microsoft .NET Framework 2.0 or Microsoft .NET Framework 1.1. If you have been using Microsoft development tools for any length of time, it's likely you already have what's needed. If not, you can obtain Microsoft .NET Framework from Microsoft, at:

http://msdn.microsoft.com

3 Obtain the VMware Infrastructure SDK (VI SDK) package from VMware Web site:

http://www.vmware.com/download/sdk/

4 Unpack the various components into appropriate sub-directories. 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, it's best to execute the command script from the Microsoft .NET 2.0 SDK command prompt, available from the .NET SDK menu.

To build the C# samples

1 Open the Microsoft .NET Framework 2.0 SDK command prompt (from the Windows Start menu, select **Programs > Microsoft .NET Framework SDK v2.0 > SDK Command Prompt**. The command prompt launches.

2 Navigate to the sub-directory containing the `Build2005.cmd` and other files:

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

3 Run the Build2005.cmd to generate new stubs and compile into an assembly. Simply enter the name of the script at the command prompt:

```
build2005
```

4 Test your development workstation. See "Running the Microsoft .NET (C#) Version of SimpleClient" on page 29.

Example 4-1. Example of a Successful Build Session Using Build2005.cmd

```
C:\devprojects\visdk21\SDK\samples\DotNet>build2005
Visual Studio C# 2005 Express is installed
Setting Path
Checking and Creating stage
A subdirectory or file stage already exists.
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 2.0.50727.42]
Copyright (C) Microsoft Corporation. All rights reserved.
Writing file 'stage\VimObjects.cs'.
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.42
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

Microsoft (R) Xml Serialization support utility
[Microsoft (R) .NET Framework, Version 2.0.50727.42]
Copyright (C) Microsoft Corporation. All rights reserved.
Serialization Assembly Name: VimService2005.XmlSerializers, Version=0.0.0.0, Culture=neutral,
               PublicKeyToken=null.
Generated serialization assembly for assembly
               C:\Data\Development\visdk21_GA\SDK\samples\DotNet\vimservice2005.dll -->
               '.\VimService2005.XmlSerializers.dll
'.
Optimizing generated stubs...
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.42
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

Stub generation Done.
ECHO is off.
Building Samples in Debug mode
Done Building optimized Stubs and all Samples

C:\devprojects\visdk21\SDK\samples\DotNet>
```

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 28 of this chapter. For all intents and purposes, using Visual Studio 2003 is discouraged.

To run the SimpleClient application

1 Navigate to the sub-directory where the compiled object code is located. From the top-level directory of the SDK download, the directory is as follows:

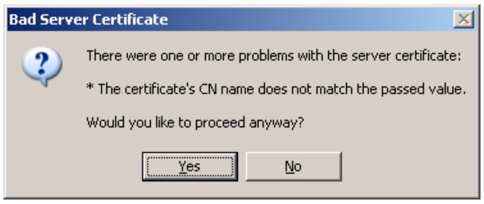
```
%SDKHOME%\samples\DotNet\cs\SimpleClient\bin\Debug
```

2 Run the application, passing to the command line the server URL and logon credentials. Note that, as coded, the SimpleClient application (SimpleClient.cs) requires that credentials (user account and password) be passed to the executable—even if the server has been configured to support HTTP. You can change the source code yourself and recompile.

If you don't provide all three arguments as shown here—server address, account name, and password—the executable generates an error message and comes to a halt.

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

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



There were one or more problems with the server certificate:

* The certificate's CN name does not match the passed value.

Would you like to proceed anyway?

3 Click **Yes** to dismiss this message and proceed to the server. Soon, the output from server should display in the console (command prompt), as shown in Example 4-2:

Example 4-2. 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 4-3 lists some common error messages and provides steps that you can take to resolve.

Table 4-3. 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.

Index