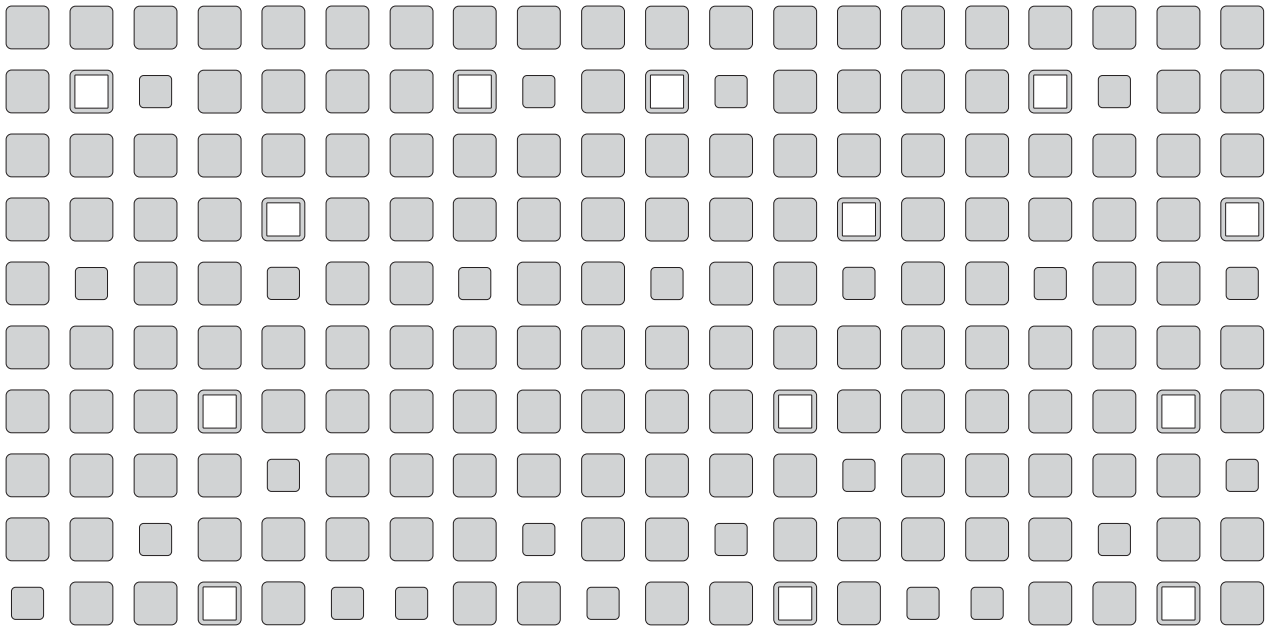


VERSION 1.0

VMware CIM SDK

Programming Guide



VMware, Inc.

3145 Porter Drive
Palo Alto, CA 94304
www.vmware.com

Please note that you will always 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.

Copyright © 1998-2004 VMware, Inc. All rights reserved. Protected by one or more of U.S. Patent Nos. 6,397,242, 6,496,847, 6,704,925, 6,711,672, 6,725,289, 6,735,601, 6,785,886, 6,789,156 and 6,795,966; patents pending. VMware is a registered trademark and the VMware "boxes" logo, GSX Server, ESX Server, Virtual SMP, VMotion and VMware ACE are trademarks of VMware, Inc. Linux is a registered trademark of Linus Torvalds. All other marks and names mentioned herein may be trademarks of their respective companies. Revision 20041202 Version 1.0 Item: SDK-ENG-Q304-008

Table of Contents

Introducing VMware CIM SDK	5
Overview of the Programming Guide	6
Using this Programming Guide	7
Intended Audience	7
Using the Pegasus CIMOM	8
Overview of VMware CIM SDK Components	9
Technical Support Resources	10
Setting Up Your Development Environment	11
Starting the Pegasus CIMOM	12
Setting Up Your Development Environment	13
Tips for Application Testing	13
VMware CIM Extension Schema	15
VMware Virtual Domain Model	16
VMware Storage Subsystems Model	17
VMware Multipathing Model	19
Sample Code	21
Copyright Information	22
Connecting to the Pegasus CIMOM	23
Retrieving Information about ESX Server	26
Retrieving Information about Virtual Environments	27
Listing the Storage Resources Available to the Host	29
Determining the Storage Resources Available to Virtual Machines	32
Glossary	35
Revision History	37

1

CHAPTER

Introducing VMware CIM SDK

The goal of the VMware® CIM SDK is to provide independent software vendors (ISVs) and the enterprise storage management industry a CIM-compliant object model for virtual machines and their related storage devices. The SDK also includes a Pegasus CIMOM installed with VMware ESX Server, as well as sample client code, to allow ISVs to explore virtual machine resources and to incorporate them into their management applications. VMware, Inc. considers the first version of the CIM SDK to be experimental. The interface may change in future releases to align it more closely with evolving standards.

With the VMware CIM SDK, independent software vendors can:

- Explore the virtual machines on the ESX Server machine and view their storage resources using any CIM client.
 - Examine virtual machine storage allocation to determine if availability and utilization policies are being satisfied.
 - Examine the physical storage allocated to a virtual machine.
 - Verify the operational status of virtual machine storage, including all storage devices and paths involved in supplying storage to virtual machines.
-

Overview of the Programming Guide

The purpose of this programming guide is to familiarize you with the logical structure of the VMware CIM SDK, the extended schema present in `VMWARE_ESX.mof`, and the components needed to explore your ESX Server machines using CIM-enabled clients. Developers who use this manual should be familiar with the Common Information Model (CIM), VMware® ESX Server™, and other VMware products.

We discuss the following topics in this reference guide:

- [Setting Up Your Development Environment on page 11](#)
This chapter describes how to configure and start the Pegasus CIMOM installed with ESX Server and how to set up for application development.
- [VMware CIM Extension Schema on page 15](#)
This chapter describes the CIM data model that describes ESX Server, virtual machines, and their storage resources.
- [Sample Code on page 21](#)
This chapter presents code samples that demonstrate basic CIM client functions, such as accessing the CIMOM and querying for data.

Using this Programming Guide

This *VMware CIM SDK Programming Guide* contains information to help you set up a development environment, data models of the VMware CIM Extension Schema, and code samples to demonstrate typical operations. This guide is meant to be used in conjunction with the HTML-based *VMware CIM SDK Reference*, which provides greater detail for each class in the VMware CIM Extension Schema.

Intended Audience

This programming guide is written for programmers that are familiar with CIM concepts and principles. Readers of this manual should be comfortable with developing system administration and system monitoring applications and be familiar with general debugging techniques. In addition, developers who use this manual should be familiar with the operation and management of ESX Server.

Note: In this release, the VMware CIM SDK supports ESX Server 2.5 and CIM 2.8.

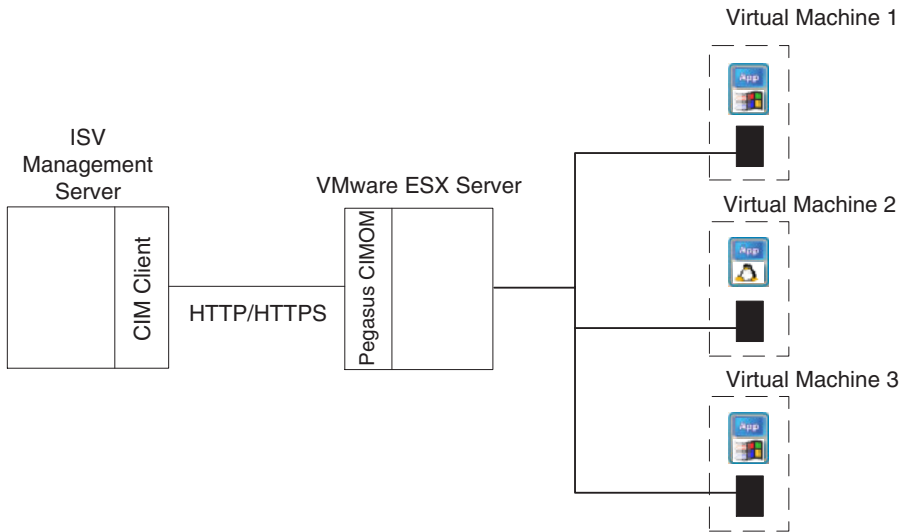
Using the Pegasus CIMOM

VMware includes a Pegasus CIMOM with ESX Server for the following reasons:

- **Accessibility** — With a CIMOM and VMware-specific providers running on the ESX Server machine, ISVs can quickly build custom agents and clients to incorporate VMware servers, virtual machines, and their available resources into enterprise management applications.
- **Industry Support** — ISVs and enterprise storage management vendors are supporting the CIM and SMI-S movements. CIM and SMI-S are designed to be independent of any particular programming language and other implementation-specific semantics. Clients or applications that query the Pegasus CIMOM can be implemented on any platform and in any programming language with an implementation of the DMTF CIM standard.
- **Remote Operations** — With appropriate firewall rules, it is possible to explore VMware environments remotely by querying the Pegasus CIMOM.
- **Low Barrier to Entry** — The concepts behind Pegasus CIMOM are easy to understand, and developers can quickly create and deploy them using many open source tool kits available on the Web.

Overview of VMware CIM SDK Components

The following diagram provides an overview of how you can use the VMware CIM SDK to manage your virtual machines:



The CIM SDK is composed of the following:

- Pegasus CIMOM to allow custom clients to explore virtual machines and their allocated resources on ESX Server
- *VMware CIM SDK Programming Guide* to provide developers with an introduction to the VMware CIM Extension Schema and with examples of usage
- *VMware CIM SDK Reference* to provide developers with an in-depth resource about the classes in the VMware CIM Extension Schema

Technical Support Resources

Refer to the following for additional information:

- VMware ESX Server — http://www.vmware.com/products/server/esx_features.html
- Distributed Management Task Force — <http://www.dmtf.org>
- Common Information Model — <http://www.dmtf.org/standards/cim/>
- OpenPegasus — <http://www.openpegasus.org>

Setting Up Your Development Environment

This chapter describes the basics of configuring your local development environment and ESX Server for CIM client application development. The following two sections discuss how to start the Pegasus CIMOM on the ESX Server machine and the basics required to begin application development using a CIM-compliant library, the installed CIMOM, and VMware Extended Schema. See the following chapter entitled [Sample Code](#) for examples of basic usage. Since CIM is a platform-independent specification, your choice of programming language and development environment may vary. Consult your CIM client library documentation for further implementation requirements.

Open source CIM toolkits for client development include:

- OpenPegasus (C++)
 - WBEM Services (Java)
 - OpenWBEM (C++)
 - SBLIM (C)
-

Starting the Pegasus CIMOM

The Pegasus CIMOM of the OpenPegasus project is installed with ESX Server 2.5 and can be accessed by any CIM implementation capable of using HTTP or HTTPS to communicate with it. The Pegasus CIMOM is responsible for receiving CIM operation requests from CIM clients and returning appropriate data (CIM operation responses) from underlying “provider” applications. In the case of ESX Server, the Pegasus CIMOM returns information about the various class objects defined in the VMware Extension Schema (`VMWARE_ESX.mof`) to requesting client applications. The Pegasus CIMOM is not run by default on server startup.

To check the status of the Pegasus CIMOM:

1. Log on to the VMware Service Console using a local or remote connection. Secure Shell (SSH) is supported on the ESX Server.
2. Switch to directory `/etc/init.d`
3. Use the command `./pegasus status` to determine the current operational status of the Pegasus CIMOM on the server.

To start the Pegasus CIMOM:

1. Log on to the VMware Service Console using a local or remote connection. Secure Shell (SSH) is supported on ESX Server.
2. Switch to directory `/etc/init.d`
3. Use the command `./pegasus start` to start the Pegasus CIMOM on the server.

To enable the Pegasus CIMOM to run by default on server startup:

1. Log on to the VMware Service Console using a local or remote connection. Secure Shell (SSH) is supported on ESX Server.
2. Execute the command `/sbin/chkconfig --add pegasus`.

Setting Up Your Development Environment

Due to the platform- and language-independence of the CIM standard, there are a number of CIM client libraries to choose from for application development. For each CIM implementation, such as OpenPegasus or WBEM Services, consult the product documentation for appropriate requirements, limitations, and setup information. Whatever language and platform you choose for the client, it will need HTTP/HTTPS access from the host on which it runs to the appropriate port on the ESX Server machine.

To connect your client applications to the Pegasus CIMOM on ESX Server, you will need the following information:

- Hostname or IP address of the ESX Server machine
- Port number that the Pegasus CIMOM is listening on. This is set to 5988 for HTTP communication and 5989 for HTTPS communication by default as recommended by the DMTF.
- Username and password
- SSL context information (if applicable)
- Namespace for the VMware Schema Extension, which is `/vmware/esx` by default

Tips for Application Testing

The following are tips for testing your CIM clients with the Pegasus CIMOM installed with ESX Server.

1. Use a CIM browser to test the connection to the CIMOM and to check the data returned. The Pegasus CIMOM installed with ESX Server supports the cimXML protocol. To test your connection to the Pegasus CIMOM using a CIM browser, you will need the IP address of your ESX Server machine, appropriate user name and a password information, and the namespace. If no data is returned, you may need to recheck your connection information and/or check to make sure that the Pegasus CIMOM is currently running. Many open source CIM browsers are available for download on the World Wide Web, and some are provided along with the client libraries needed for CIM-based development. Your programming language and development platform will determine which CIM browsers are available to you.
2. Set the following shell variable on the client machine to check the XML transmitted between the client and the Pegasus CIMOM: `set XML_TRACE=1`. Two files will be generated, `xml_out.log` (the request sent to the CIMOM) and `xml_in.log` (the response from the CIMOM) in the local directory of the client machine.

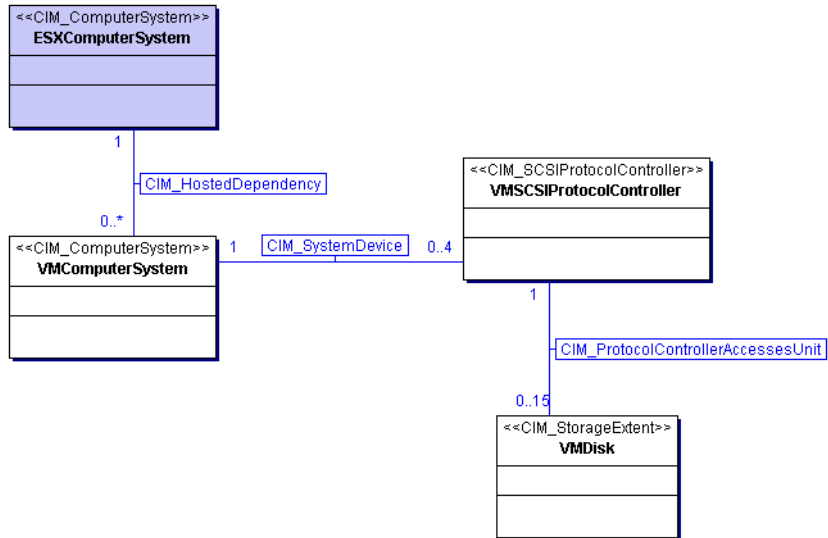
VMware CIM Extension Schema

The following section provides a general overview of the VMware CIM Extension Schema; that is, the logical structure of ESX Server, its virtual machines, and their allocated resources as defined using the Common Information Model. Descriptions of many of the classes are given in the following section as well as UML diagrams that show the relationships between the classes and the hierarchical structure of the VMware Schema Extension. Consult the HTML-based VMware CIM SDK Reference for a complete reference of the classes, properties, and relationships defined in the VMware CIM Extension Schema defined in `VMWARE_ESX.mof`.

The classes presented here are organized in the following major groups:

- VMware Virtual Domain model — Traverses the schema from ESX Server down to the individual storage devices of its virtual machines.
- VMware Storage Subsystems model — Describes the storage components on ESX Server that can be allocated to its virtual machines.
- VMware ESX Server Multipathing model — Describes the virtual machine configuration and sharing of the host machine.

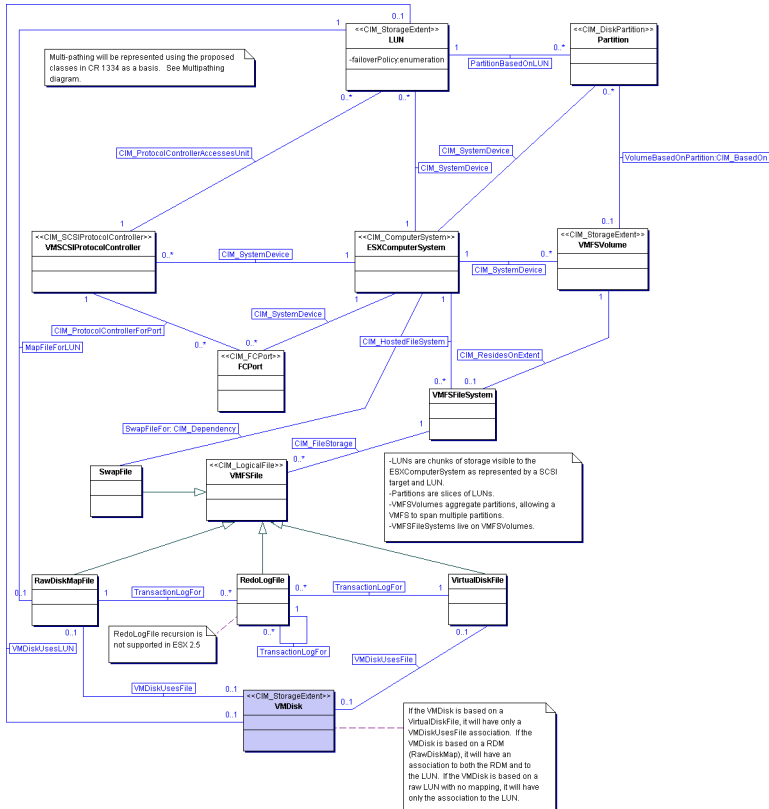
VMware Virtual Domain Model



The preceding figure shows the basic classes in the VMware Virtual Domain:

- ESXComputerSystem — Represents ESX Server.
- VMComputerSystem — Represents the virtual machines running on ESX Server.
- VMSCSIProtocolController — Represents a storage adapter device allocated to a virtual machine.
- VMDisk — Represents a storage device allocated to a virtual machine.

VMware Storage Subsystems Model

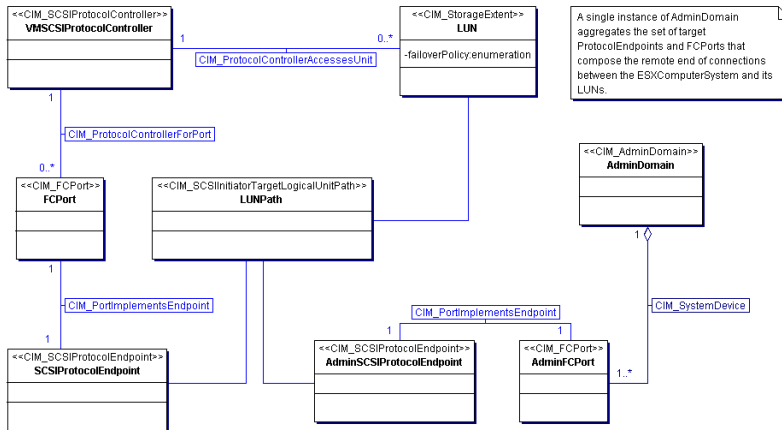


The preceding figure shows the storage subsystems classes defined in the VMware CIM Extension Schema. These classes define the storage components available on ESX Server:

- ESXComputerSystem — Represents ESX Server.
- LUN — Represents a logical or physical storage device available to the server, as identified by a SCSI target and an identifier.
- Partition — Represents a partition on a physical storage device. A partition is a slice of a LUN.
- VMFSFilesystem — Represents a file system available to a virtual machine.
- VMFSVolume — Represents a logical storage volume available to a virtual machine.

- VMFSFile — Represents a file on a VMFSVolume.
- RedoLogFile — Represents a file that journals changes to a VirtualDiskFile or a RawDiskMapFile.
- VirtualDiskFile — Represents a file that constitutes a VMDisk.
- VMDisk — Represents a storage device allocated to a virtual machine.
- RawDiskMapFile — Represents the mapping of a VMFSFile to a raw storage device available to the ESX Server.
- SwapFile — Represents a swap file used for virtual memory on ESX Server.
- .FCPort — Represents a Fibre Channel port.
- VMSCSIProtocolController — Represents a storage adapter device allocated to a virtual machine.

VMware Multipathing Model



The preceding figure shows the multipathing CIM data model for ESX Server.

- LUN — Represents a logical or physical storage device available to the server, as identified by a SCSI target and an identifier.
- AdminDomain — Represents an aggregate of the SCSIProtocolEndPoints and FCPorts that are the remote ends of the connections between ESX Server and its LUNs.
- AdminFCPort — Represents an aggregate of the Fibre Channel ports that link ESX Server to its LUNs.
- AdminSCSIProtocolEndPoint — Represents an aggregate of the SCSI ports that link ESX Server to its LUNs.
- VMSCSIProtocolController — Represents a storage adapter device, such as a host bus adapter (HBA), allocated to a virtual machine.
- FCPort — Represents a Fibre Channel port.
- LUNPath — Represents a path to a LUN.
- SCSIProtocolEndPoint — Represents a SCSI port that links ESX Server to a LUN.

CHAPTER 4

Sample Code

The following examples are included as starting points to help you understand the basic programming concepts required to connect to the Pegasus CIMOM on ESX Server and to send and receive CIM operation requests and responses. Each example represents a small sample of code in a large application and is designed only to show the basic logic and programming constructs needed to use the VMware CIM SDK effectively. All examples use the C++ programming language and the OpenPegasus CIM client library. Your choice of language and library may vary.

All sample code that is provided as part of the VMware SDK kit is subject to the following copyright.

Copyright Information

Each sample program included with the VMware SDK includes a copyright. However, for brevity, we do not include this copyright in its entirety with each sample program. Instead, we include the first line of the copyright followed by ellipses, to indicate its placement. The complete copyright is as follows:

```
Copyright (c) 1998-2004 VMware, Inc.
```

```
Permission is hereby granted, free of charge, to any person obtaining a
copy of the software in this file (the "Software"), to deal in the
Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

```
The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.
```

```
The names "VMware" and "VMware, Inc." must not be used to endorse or
promote products derived from the Software without the prior written
permission of VMware, Inc.
```

```
Products derived from the Software may not be called "VMware", nor may
"VMware" appear in their name, without the prior written permission of
VMware, Inc.
```

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
VMWARE, INC. BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

Connecting to the Pegasus CIMOM

The following example shows programming code that establishes a connection between the CIM client and the CIMOM.

```

/*
 * Goal: Connect to the ESX Server
 */

#include <iostream>
#include <vector>
#include <algorithm>
#include <string>

#include "Pegasus/Client/CIMClient.h"

using namespace std;
using namespace Pegasus;

bool connect( Pegasus::CIMClient& theClient,
             const std::string& theHost,
             unsigned long thePort,
             const std::string& theSSLCertificatePath,
             const std::string& theUserName,
             const std::string& thePassword);

void disconnect( Pegasus::CIMClient& theClient);

int main(int argc, char* argv[])
{
    CIMClient myClient;

    // ensure that the correct command line arguments exist
    if (argc != 6)
    {
        cout << "Usage: " << argv[0]
              << " Host Port User Password msTimeout"
              << endl;
        return 1;
    }

    // the host name or IP address of CIM server
    string myHost(argv[1]);

    // the port number for the server and client to use

```

```

int myPort = atoi(argv[2]);

// the name of the user that the client is connecting as
string myUsername(argv[3]);

// the password of the user the client is connecting as
string myPassword(argv[4]);

// the timeout in milliseconds for the CIMClient
unsigned long myTimeout = atol(argv[5]);

    cout << "Connecting to " << myHost << ":" << myPort << endl;
    if( !connect( myClient,
                myHost,
                myPort,
                "",
                myUsername,
                myPassword ) )
    {
        cout << "Could not connect." << endl;
        return 1;
    }

    myClient.setTimeout(myTimeout);
    cout << "Connected. " << endl << "Timeout was set to " <<
        myClient.getTimeout() << " milliseconds" << endl;

    cout << endl << "Disconnecting..." << endl;

    disconnect(myClient);
    return 0;
}

/*
 * Create an HTTP connection with the CIM server defined by the host and
 * portNumber.
 *
 * Return Value: true - connection succesfully created.
 *              false - could not connect.
 */
bool connect( CIMClient& theClient,
             const string& theHost,
             unsigned long thePort,
             const string& theSSLCertificatePath,
             const string& theUserName,
             const string& thePassword)
{

```

```

// number of times to retry if the initial connection attempt fails
int nRetry=5;

while (nRetry-->0)
{
    try
    {
        if(theSSLCertificatePath.empty())
        {
            theClient.connect( theHost.c_str(),
                               thePort,
                               theUserName.c_str(),
                               thePassword.c_str() );
        }
        else
        {
            theClient.connect( theHost.c_str(),
                               thePort,
                               SSLContext( theSSLCertificatePath.c_str(),
                                           NULL),
                               theUserName.c_str(),
                               thePassword.c_str() );
        }

        // successfully connected if here
        return true;
    }
    catch(const CannotConnectException& e)
    {
        cout << "\nCAUGHT A CANNOT CONNECT EXCEPTION: " << e.getMessage()
              << "\n" << endl;
        cout << "Attempting again: Retries left = " << nRetry << endl;
    }
    catch(const Exception& e)
    {
        cout << "\nCAUGHT AN EXCEPTION: " << e.getMessage() << "\n"
              << endl;
        cout << "Attempting again: Retries left = " << nRetry << endl;
    }
}
return false;
}

void disconnect( Pegasus::CIMClient& theClient )
{
    theClient.disconnect();
    cout << "Disconnected" << endl;
}

```

Retrieving Information about ESX Server

The following example shows how to query the CIMOM for information about the ESX Server machine.

```

/*
 * Goal: Retrieve Information about ESX Server
 */

const CIMNamespaceName VMWareNS("vmware/esx");
const CIMName ESXComputerName("VMWARE_ESXComputerSystem");

Array< CIMInstance > ESXCompInstanceArray=myClient.enumerateInstances(
    VMWareNS, ESXComputerName);

for (int i=0; i<ESXCompInstanceArray.size(); i++) {

    CIMInstance currESXInstance=ESXCompInstanceArray[i];

    // get ElementName property
    Uint32 nElemNamePropIndex=currESXInstance.findProperty("ElementName");

    const CIMValue
        apValue=currESXInstance.getProperty(nElemNamePropIndex).getValue();

    String sESXElemName;
    apValue.get(sESXElemName);

    // get Description property
    Uint32 nDescPropIndex=currESXInstance.findProperty("Description");
    const CIMValue DescPropValue=currESXInstance.getProperty(nDescPropIndex
        ).getValue();
    String sESXDesc;
    DescPropValue.get(sESXDesc);

    // get NumberOfCPUs property
    Uint32 nCPUPropIndex=currESXInstance.findProperty("NumberOfCPUs");
    const CIMValue CPUPropValue=currESXInstance.getProperty(nCPUPropIndex
        ).getValue();
    Uint8 nCPUCount=0;
    CPUPropValue.get(nCPUCount);

    printf("\t%s: %s has %d CPUs.\n", sESXDesc.getCString(),
        sESXElemName.getCString(), nCPUCount);
}

```

Retrieving Information about Virtual Environments

The following example shows how to query the CIMOM for information about the virtual environments on ESX Server.

```

/*
 * Goal: Retrieve the name, OS type and memory available for each
 * virtual machine.
 */

const CIMNamespaceName VMWareNS("vmware/esx");
const CIMName ESXComputerName( "VMWARE_ESXComputerSystem" );

Array< CIMObjectPath > ESXObjPathArray=myClient.enumerateInstanceNames (
VMWareNS, ESXComputerName);

const CIMName VM_ComputerName( "VMWARE_VMComputerSystem" );

for ( int i=0; i<ESXObjPathArray.size(); i++)
{
    Array< CIMObject > assocObjArray=myClient.associators( VMWareNS,
        ESXObjPathArray[i], "", VM_ComputerName);

    for (int j=0; j<assocObjArray.size(); j++)
    {
        CIMObject VMInstance=assocObjArray[j];

        // get the virtual machine element name
        UInt32 nElemNamePropIndex=VMInstance.findProperty("ElementName");
        const CIMValue elemNamePropValue=VMInstance.getProperty(
            nElemNamePropIndex ).getValue();
        String sElemName;
        elemNamePropValue.get( sElemName);

        // get the virtual machine operating system
        UInt32 nOSPropIndex=VMInstance.findProperty("GuestOS");
        const CIMValue OSPropValue=VMInstance.getProperty( nOSPropIndex
            ).getValue();
        String sOSName;
        OSPropValue.get( sOSName);

        // get the total memory on the virtual machine
    }
}

```

```
        Uint32 nMemPropIndex=VMInstance.findProperty("TotalMemory");
        const CIMValue nMemPropValue=VMInstance.getProperty(
            nMemPropIndex ).getValue();
        Uint64 nTotalMemoryKB=0;
        nMemPropValue.get( nTotalMemoryKB);

        printf("\tVirtual Machine %d: Total Mem: %I64d KB \tName: %s [OS: %s]\n",
            j+1, nTotalMemoryKB, sElemName.getCString(), sOSName.getCString());
    } // end for j

} // end
```

Listing the Storage Resources Available to the Host

The following example shows how to query the CIMOM for information about the storage resources available to ESX Server.

```

/*
 * Goal: Retrieve the LUN data and partition data for each LUN of ESX Server
 */

const CIMNamespaceName VMWareNS("vmware/esx");
const CIMName ESXComputerName( "VMWARE_ESXComputerSystem" );
const String antecedentRole("Antecedent");

Array< CIMObjectPath > ESXObjPathArray=myClient.enumerateInstanceNames (
    VMWareNS, ESXComputerName);

Array< CIMInstance > ESXCompInstanceArray=myClient.enumerateInstances(
    VMWareNS, ESXComputerName);

const CIMName VMWARE_LUN("VMWARE_LUN");
const CIMName VMWARE_Partition("VMWARE_Partition");
String sESXElemName;
String sESXDesc;
String sDevID;
String sVendor;
String sConsoleDeviceName;
String sElemName;
Uint32 nCylCount=0;

for ( int i=0; i<ESXObjPathArray.size(); i++)
{
    Array< CIMObject > assocLUNObjArray=myClient.associators( VMWareNS,
        ESXObjPathArray[i], "", VMWARE_LUN);
    CIMInstance ESXInstance=ESXCompInstanceArray[i];

    // get ElementName property
    const CIMValue apValue=ESXInstance.getProperty(
        ESXInstance.findProperty("ElementName") ).getValue();
    apValue.get( sESXElemName);

    // get Description property

```

```

const CIMValue DescPropValue=ESXInstance.getProperty(
    ESXInstance.findProperty("Description") ).getValue();
DescPropValue.get( sESXDesc);

printf("%s: %s \n", sESXDesc.getCString(), sESXElemName.getCString());

// for each LUN on the server.
for (int j=0; j<assocLUNObjArray.size(); j++)
{
    CIMObject LUN=assocLUNObjArray[j];

    // get the Device ID
    const CIMValue devIDPropValue=LUN.getProperty(
        LUN.findProperty("DeviceID") ).getValue();
    devIDPropValue.get( sDevID );

    // get the Vendor name
    const CIMValue vendorPropValue=LUN.getProperty(
        LUN.findProperty("Vendor") ).getValue();
    vendorPropValue.get( sVendor);

    // get the number of cylinders on the LUN
    const CIMValue CylValue=LUN.getProperty(
        LUN.findProperty("Cylinders") ).getValue();
    CylValue.get( nCylCount);

    printf("\n\tLUN %d: %s    Cylinder count %d: \tVendor Name: %s
        \n", j+1, sDevID.getCString(), nCylCount, sVendor.getCString()
        );

    // get Partitions associated to this LUN
    Array< CIMObject > LUNPartitionArray=myClient.associators(
        VMWareNS, LUN.getPath(), "", VMWARE_Partition, antecedentRole);
    for (int k=0; k<LUNPartitionArray.size(); k++)
    {
        // a slice of a LUN
        CIMObject LUNPartition=LUNPartitionArray[k];
        // get ConsoleDevice property of this partition
        Uint32 nConsDevPropIndex=LUNPartition.findProperty(
            "ConsoleDevice");
        if (nConsDevPropIndex != PEG_NOT_FOUND)
        {
            const CIMValue consDevPropValue=LUNPartition.getProperty(
                nConsDevPropIndex ).getValue();
            consDevPropValue.get( sConsoleDeviceName);
        }
    }
}

```

```
        // get ElementName property
        const CIMValue elemNameValue=LUNPartition.getProperty(
            LUNPartition.findProperty("ElementName") ).getValue();
        elemNameValue.get( sElemName);

        printf("\t\t\tPartition %s\t Name: %s \n",
            sConsoleDeviceName.getCString(), sElemName.getCString()
            );

    } // end for n

} // end for j
} // end for i
```

Determining the Storage Resources Available to Virtual Machines

The following example shows how to query the CIMOM for information about the storage resources available to individual virtual machines.

```

/*
 * Goal: Retrieve the system devices available to each VM.
 *       For virtual disks, compute the available space.
 */

#define ONE_MB          (1024 * 1024)
const CIMNamespaceName  VMWareNS("vmware/esx");
const CIMName  ESXComputerName( "VMWARE_ESXComputerSystem" );
const String  antecedentRole("Antecedent");
const CIMName  VM_ComputerSystem( "VMWARE_VMComputerSystem" );
const CIMName  VMWARE_VMDisk("VMWARE_VMDisk");
const CIMName
    VMWare_VMSCSIProtocolController("VMWare_VMSCSIProtocolController");
const CIMName  VMWARE_SystemDevice("VMWARE_SystemDevice");
String  sVMElemName;
String  sVMDesc;
String  sElemName;
String  sName;
String  sDesc;

Array< CIMInstance > VMInstanceArray=myClient.enumerateInstances( VMWareNS,
    VM_ComputerSystem);

// for each VM found
for ( int i=0; i<VMInstanceArray.size(); i++)
    {
        CIMInstance  VMInstance=VMInstanceArray[i];

        // get ElementName property
        const CIMValue  elemName=VMInstance.getProperty(
            VMInstance.findProperty("ElementName") ).getValue();
        elemName.get( sVMElemName);

        // get Description property
        const CIMValue  DescPropValue=VMInstance.getProperty(
            VMInstance.findProperty("Description") ).getValue();
        DescPropValue.get( sVMDesc);
    }

```


Glossary

CIM— Common Information Model (CIM). The CIM conceptual information model for describing management that is not bound to a particular implementation.

CIMOM— Common Information Model Object Manager. A CIMOM stores class definitions and populates requests for them with information returned from specific data providers.

ISV— Independent software vendor — Systems management vendors, enterprise management frameworks, imaging and provisioning vendors, storage management vendors, and so on.

MOF— Managed Object Format. CIM file format for describing model classes.

Pegasus— OpenPegasus is an open-source implementation of the DMTF CIM and WBEM standards.

Server— A server is a system capable of managing and executing virtual machines.

Virtual Machine— A virtual machine contains system and configuration states for a machine and may execute on a host machine.

Revision History

The following table lists the revision history for the *VMWare CIM SDK Programming Guide*.

Date	Description
December 2, 2004	Version 1.0 release

Index

A

AdminDomain 19
AdminFCPort 19
AdminSCSIProtocolEndPoint 19

E

ESXComputerSystem 16, 17

F

FCPort 18, 19

L

LUN 17, 19
LUNPath 19

O

overview 9

P

Partition 17
Pegasus CIMOM
 checking status 12
 starting 12

R

RawDiskMapFile 18
RedoLogFile 18

S

SCSIProtocolEndPoint 19
SwapFile 18

T

Technical Support 10
tips 13
toolkits 11

V

VirtualDiskFile 18
VMComputerSystem 16
VMDisk 16, 18
VMFSFile 18
VMFSFileSystem 17
VMFSVolume 17
VMSCSIProtocolController 16, 18, 19
VMware Storage Subsystems Model 17
VMware Virtual Domain Model 16

