Product Documentation

OpenPegasus Administrator's Guide
Release 2.4

Product Documentation

OpenPegasus Administrator’s Guide

ISBN: 1-931624-48-8

Document Number: G042

Published by The Open Group, April 2005.

Comments relating to the material contained in this document may be submitted to:

The Open Group
Thames Tower
37-45 Station Road
Reading
Berkshire, RG1 1LX
United Kingdom

or by electronic mail to:

ogedit@opengroup.org

Contents

Preface

The Open Group

The Open Group is a vendor-neutral and technology-neutral consortium, whose vision of Boundaryless Information Flow will enable access to integrated information within and between enterprises based on open standards and global interoperability. The Open Group works with customers, suppliers, consortia, and other standards bodies. Its role is to capture, understand, and address current and emerging requirements, establish policies, and share best practices; to facilitate interoperability, develop consensus, and evolve and integrate specifications and Open Source technologies; to offer a comprehensive set of services to enhance the operational efficiency of consortia; and to operate the industry's premier certification service, including UNIX certification.

Further information on The Open Group is available at www.opengroup.org.

The Open Group has over 15 years' experience in developing and operating certification programs and has extensive experience developing and facilitating industry adoption of test suites used to validate conformance to an open standard or specification.

More information is available at www.opengroup.org/testing.

The Open Group publishes a wide range of technical documentation, the main part of which is focused on development of Technical and Product Standards and Guides, but which also includes white papers, technical studies, branding and testing documentation, and business titles. Full details and a catalog are available at www.opengroup.org/pubs.

This Document

This document is Product Documentation for the OpenPegasus open source implementation of the DMTF WBEM specifications. It is maintained by the OpenPegasus community and published by The Open Group.

The scope of this document is the default build configuration of OpenPegasus. Customized and optional configurations may require additional information.

This Guide is intended for system administrators and describes how to install, configure, and maintain the OpenPegasus CIM Server. The contents are as follows:

- Chapter 1, Overview of OpenPegasus, introduces WBEM Standards and the OpenPegasus Architecture.
- Chapter 2, How Does OpenPegasus Work?, gives an idea of how providers and clients work.

- Chapter 3, OpenPegasus , lists the commands, executables, and daemon processes that are available.
- Chapter 4, Example of a Client Request, shows a client request and the response received, both encoded in XML.
- Chapter 5, Installing and Setting Up OpenPegasus, describes what system administrators should do before they actually use OpenPegasus. It tells how to prepare the system for installation, and how to start OpenPegasus. It lists the OpenPegasus configuration properties that can be set.
- Chapter 6, Security Considerations, describes WBEM Services security, and describes WBEM Services authentication, authorization, and encryption.
- Chapter 7, OpenPegasus Troubleshooting, lists some troubleshooting suggestions . It also lists the messages generated by OpenPegasus.
- Appendix A gives some background into CIM terms used by clients and providers to represent resources in the repository.
- Appendix B lists the operations implemented in WBEM Services.
- Appendix C explains the OpenPegasus Configuration Options Security Disclaimer.

OpenPegasus supports a variety of build-time configuration options. In addition, vendors are free to enhance or change the OpenPegasus source. Refer to the vendor’s documentation for details of their release.

Trademarks

Boundaryless Information Flow™ is a trademark and UNIX® and The Open Group® are registered trademarks of The Open Group in the United States and other countries.

Microsoft® and Windows NT® are registered trademarks of Microsoft Corporation.

The Open Group acknowledges that there may be other brand, company, and product names used in this document that may be covered by trademark protection and advises the reader to verify them independently.

Acknowledgements

The Open Group and the OpenPegasus Community gratefully acknowledges the contribution of the following in the development of this document:

- Hewlett-Packard Company for donating the initial version of this document
- The Distributed Management Task Force (DMTF) for much of the Glossary

Referenced Documents

The following documents are referenced in this Technical Guide:

DMTF CIM Operation over HTTP
: CIM Operation over HTTP Specification
(see www.dmtf.org/standards/documents/WBEM/DSP200.html).

DMTF CIM Specification
: Common Information Model (CIM) Specification, Version 2.2
(see www.dmtf.org/standards/cim).

DMTF CIM Tutorial
: Tutorial for the DMTF Common Interface Model (CIM)
(see www.dmtf.org/education).

DMTF CIM-XML Specification
: Specification for the Representation of CIM in XML
(see www.dmtf.org/standards/documents/WBEM/DSP201.html).

OpenSSL
: OpenSSL Toolkit (see www.openssl.org/docs).

W3C XML
: eXtensible Markup Language (XML), W3C Architecture Domain
(see www.w3.org/XML).

1 Overview of OpenPegasus

This chapter introduces OpenPegasus: what it is, where it comes from, and how you can learn more about it.

Web-Based Enterprise Management (WBEM) is a Distributed Management Task Force (DMTF) standard based on the Common Information Model (CIM) (see www.dmtf.org). WBEM allows customers to manage their systems consistently across multiple platforms and operating systems, enabling the deployment of integrated, multi-vendor solutions. WBEM enables management applications to monitor and control managed resources wherever and whenever required.

OpenPegasus is an Open Source Software (OSS) implementation of the DMTF WBEM standard. The OpenPegasus project is hosted by The Open Group (see www.openpegasus.org).

As an implementation of the DMTF WBEM standard, the OpenPegasus CIM Server acts as an information broker between management applications, called CIM Clients, and management instrumentation, called CIM Providers.

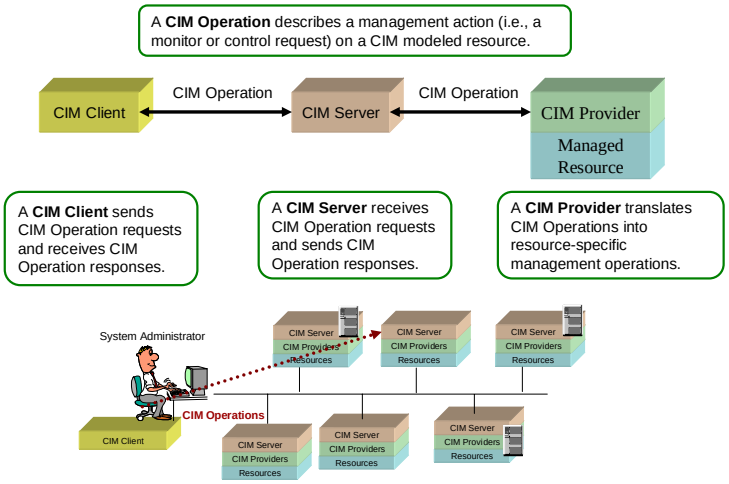

Figure 1: WBEM Solution Architecture Overview

A CIM Client is used to request access to a CIM managed resource. A CIM Client sends CIM Operation requests to the OpenPegasus CIM Server to monitor and control resources.

A CIM Provider is developed to offer access to a CIM-modeled managed resource. A Provider is responsible for translating CIM Operation requests into resource-specific management operations and translating resource-specific responses into CIM Operation responses. A CIM Provider registers with the OpenPegasus CIM Server to manage one or more CIM-modeled resources.

OpenPegasus runs on a wide variety of operating platforms, including Linux, Windows, HP-UX, AIX, OS/400, Solaris, Mac OS/X, and zOS.

The CIM Operations that OpenPegasus supports are listed in Appendix B.

1.1 DMTF WBEM Standards

OpenPegasus implements the DMTF WBEM standard. WBEM is a platform and resource-independent DMTF standard that defines both a common model and protocol for monitoring and controlling resources from diverse sources (e.g., different types of platforms or different types of resources on the same platform).

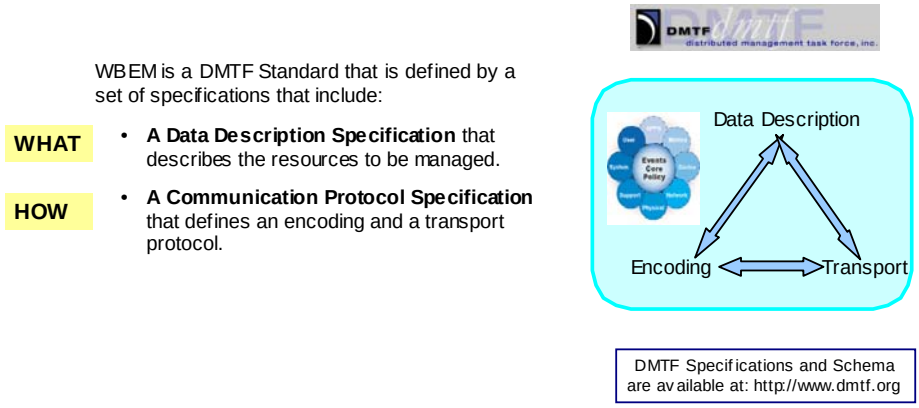

Figure 2: WBEM Specification Overview

OpenPegasus implements the following three DMTF WBEM specifications:

- Common Information Model (CIM) Infrastructure Specification

  This specification defines the language and methodology for describing managed resources. Using CIM, WBEM provides a platform and resource-neutral mechanism for management applications to describe a request to access a managed resource.

  For an overview of CIM objects, see Appendix A.

- Specification for the Representation of CIM in XML

  This specification defines a standard for encoding CIM Operation requests and responses into XML. For an overview of XML, see the W3C Architecture domain's eXtensible Markup Language (XML).

  Chapter 4 provides an example of an XML-encoded CIM Operation request and response.

- Specification for CIM Operations over HTTP

  This specification defines the CIM Operations. In addition, it defines the HyperText Transfer Protocol (HTTP)-based transport protocol used to send XML-encoded CIM Operation requests and responses between the CIM Client and the CIM Server.

For more information about the OpenPegasus HTTP Server, the ports reserved for OpenPegasus, and other transport considerations, see Chapter 6.

For more information, see the following DMTF Specifications:

- CIM Specification[1]
- Specification for the Representation of CIM in XML
- Specification for CIM Operations over HTTP

1.2 OpenPegasus Architecture

The four main components of a WBEM solution are the managed resources, the CIM Providers, the CIM Server, and the CIM Clients.

- A CIM Client issues CIM Operation requests and receives and processes CIM Operation responses.
- A CIM Server receives CIM Operation requests, coordinates the processing of requests and responses among the Providers, and sends CIM Operation responses back to the CIM Client.
- A CIM Provider is responsible for the actual processing of CIM Operations for one or more managed resources. It provides the mapping between the CIM interface and a resource-specific interface.
- A Managed Resource is a manageable entity (e.g., memory, process, system, application, or network) plus the resource-specific instrumentation capable of monitoring and controlling the resource.

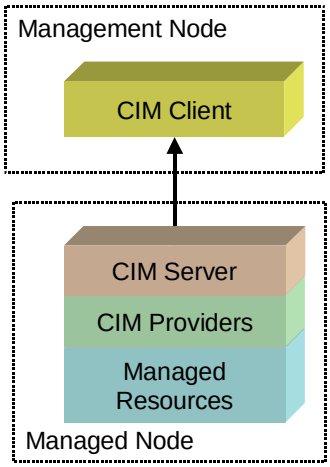

Figure 3: WBEM Solution Components

[1] With Version 2.3, this Specification has been renamed to the CIM Infrastructure Specification.

The OpenPegasus release includes a DMTF-compliant CIM Server implementation as well as a set of CIM Client and CIM Provider applications.

At the most fundamental level, the OpenPegasus CIM Server includes a CIM Object Manager (CIMOM), a CIM-XML Protocol Adapter, and a CIM Repository.

- The CIMOM is responsible for the implementation of the "protocol-independent" semantics of CIM objects and operations.
- A Protocol Adapter is responsible for the implementation of the encoding and transport components of a protocol. The CIM-XML Protocol Adapter implements the DMTF CIM-XML encoding and transport protocol. As a security consideration, it is worth noting that although the OpenPegasus CIM Server includes an embedded HTTP Server, this server has been specifically designed to accept only valid CIM Messages; all other HTTP requests will be rejected.
- The CIM Repository is a persistent store managed by the CIM Server. It contains a description, using CIM, of the resources that can be managed. For information about maintaining and restoring the repository, see Section 5.4 and Section 7.1.

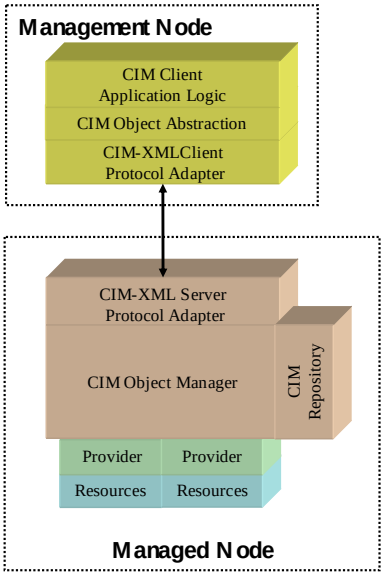

Figure 4: CIM Server Architecture

2 How Does OpenPegasus Work?

This chapter gives an idea of how OpenPegasus provides a management infrastructure so clients and providers can communicate.

It outlines how providers register their resources' properties (attributes or characteristics) and methods (capabilities, operations, or actions) with OpenPegasus.

It gives an overview of how clients use OpenPegasus to make a request about a resource and receive a response. Chapter 4 has an example of an actual request sent by a client, and the response it received.

The OpenPegasus CIM Server can receive requests from clients on many different kinds of systems and platforms, as long as the requests conform to the DMTF CIM-XML standard. The CIM Server processes the clients' requests, and passes them to the appropriate providers. When providers receive requests, they pass information back to the CIM Server. Then the CIM Server sends a response back to the client.

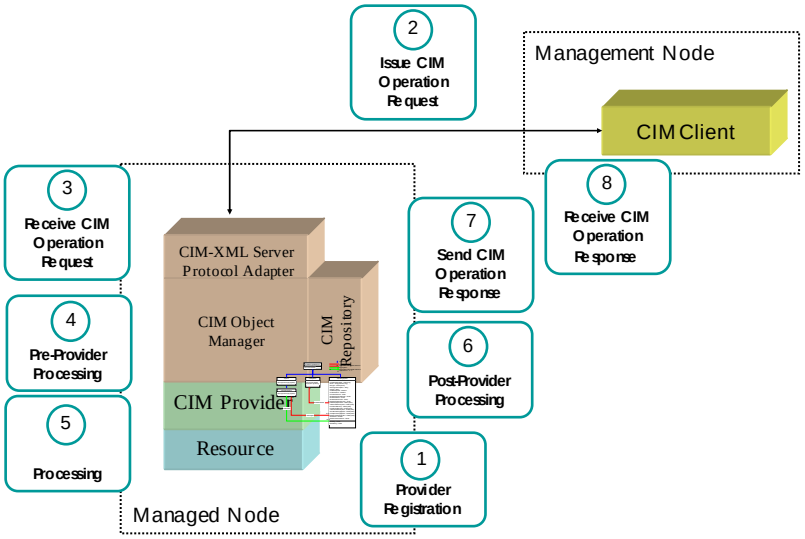

Figure 5: CIM Operation Flow

2.1 OpenPegasus Providers

To allow the OpenPegasus CIM Server to expose management operations for a specific resource, a developer writes software called a CIM Provider. A CIM Provider is responsible for the actual processing of CIM Operations for one or more managed resources. It provides the mapping between the CIM interface and a resource-specific interface. When you install a Provider on your system, it must be registered with the CIM Server.

2.1.1 When a Provider Installs

The following information needs to be supplied as part of Provider registration:

- The CIM definition of the managed resource. See Appendix A.

  Resources are defined largely by characteristics inherited from the most general classes and passed to the more specific subclasses.

  For example, there could be a schema, `Creature`, which contains a class `Human`. `Human` could, in turn, have a subclass `Female`. Class `Female` could, in turn, have several more subclasses until we get to the specific instance of `MyMother`.

  Resources can also be grouped in namespaces. OpenPegasus installs with four namespaces, listed in Appendix A.

- What information the resource provider will expose (make available) about the resource. These are the properties and methods.

  For example, one property of `MyMother` would be her unique `Name` and `SocialSecurityNumber`. Other properties might include `Birthdate` and `PhoneNumber`.

- A shared library to invoke the actions that are offered to manage the resource.

  For example, it would be handy if the method `callMother` would remind me of her `PhoneNumber` when her `Birthdate` approaches.

- Information about the Provider itself: its version, its type, a description of itself, how to invoke it, and the name of its shared libraries.

Providers are enabled automatically when they are registered. After that, they can be disabled with the `cimprovider` command. Once disabled, they can be re-enabled with the `cimprovider` command.

2.1.2 Provider Responsibilities

Developers of a WBEM resource provider are responsible for informing their users (clients) about their provider: how to specify the provider's resources in the CIM Schema, and what properties and methods it offers.

After a Provider has registered, the provider's developers can replace it with a newer version to add, remove, and modify information about the resource, including new classes, properties, and methods.

2.2 Client Requests

A CIM Client issues CIM Operation requests and receives and processes CIM Operation responses. OpenPegasus includes a C++ Client interface that implements the CIM-XML protocol.

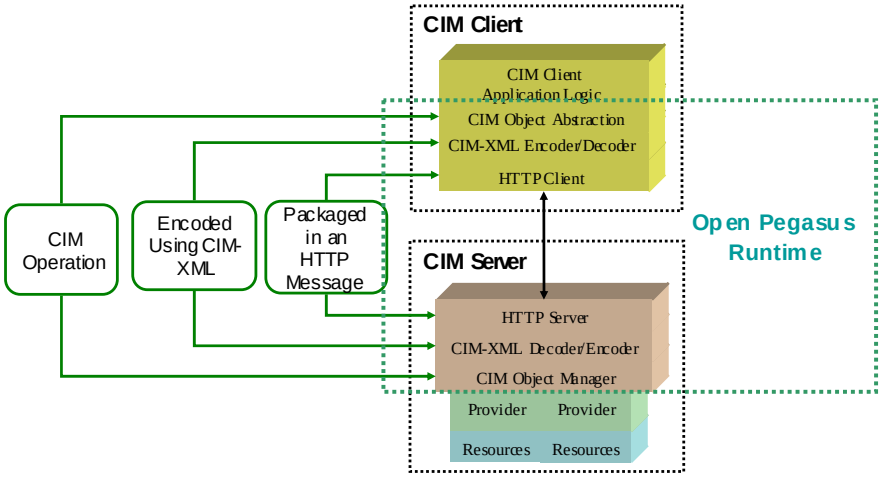

Figure 6: CIM-XML Protocol

At the CIM-XML protocol level, a CIM Client request must include:

- A properly formed HTTP header.

  A remote request must be addressed to the OpenPegasus' HTTP Server on the wbem-http port or wbem-https port. Requests must be written in XML. For information about XML coding for CIM, see the DMTF CIM-XML specification.

- The operation desired and its required parameters.

  For example, the `GetClass` operation requires a class name. The `osinfo` request in Chapter 4 uses the `EnumerateInstances` operation; its only requirement is the class name.

- The namespace.

  For example, the `osinfo` request in Chapter 4 specifies the `PG_OperatingSystem` class in the `root/cimv2` namespace.

It is the responsibility of the resource's provider to document the name of the resource and its properties and methods. Client developers can use the documentation to write client software. System administrators use the documentation to decide whether to install the provider.

A client can use CIM Operations, such as the `EnumerateInstances` operation used in the example in Chapter 4. The client developer uses standard CIM Operations like `GetClass` and

`GetProperty` to gather resource information. The CIM Operations supported by OpenPegasus are listed in Appendix B.

2.3 Processing Requests

A CIM-XML Request is a CIM Operation request encoded in XML (eXtensible Markup Language) and transported over HTTP or HTTPS. The OpenPegasus HTTP Server can be configured to listen for CIM messages on the wbem-http port or the wbem-https port.

1. First, the client connects to OpenPegasus' HTTP Server. A remote client sends a valid system login (name and password) to a system with OpenPegasus that has the appropriate provider installed. For information about login permissions, see Chapter 6.

2. The OpenPegasus CIM Server uses its XML decoder to parse the XML in the request. If there is an error, it returns an error message and stops processing the request. Only a valid CIM Operation is accepted. A request could be rejected by the HTTP Server if it had badly formed HTTP headers or badly formed XML.

   For information about XML coding for CIM, see the DMTF CIM-XML specification.

3. If the request is valid, the CIM Server consults the CIM Repository and checks the following:

   — Does this namespace exist? If not, an error is returned and OpenPegasus stops processing the request. For example, the `osinfo` request used in Chapter 4 has this namespace information:

   ```
   <LOCALNAMESPACEPATH>
   <NAMESPACE NAME =”root”/>
   <NAMESPACE NAME=”cimv2”/>
   </LOCALNAMESPACEPATH>
   ```

   — Does this user have permission in this namespace? If the OpenPegasus property `enableNamespaceAuthorization` is set to true, OpenPegasus will also check to be sure the user is allowed access to this namespace. (See Chapter 6 for more about authorization.)

   — Does this class exist? OpenPegasus looks up the `classname` given in the request. For example, the `osinfo` request used in Chapter 4 has this class information:

   ```
   <IPARAMVALUE NAME=”ClassName”>
   <CLASSNAME NAME=”PG_OperatingSystem”/>
   </IPARAMVALUE>
   ```

   — Does this resource have a registered provider? If there is no provider registered for this resource, OpenPegasus returns an error to the client. For example, the provider for the `osinfo` client request is the Operating System Provider.

4. When OpenPegasus finds the registered provider, it also finds the provider's instructions about how to reach its appropriate shared library.

   OpenPegasus uses this to invoke the appropriate method, and tell the provider which user is making the request. After receiving the request, the provider's developers are

responsible for any additional user authorization it requires for performing the action, and for returning a response to OpenPegasus.

5. OpenPegasus' CIM Server waits for a response from the provider, and conveys the response back to the client. Each request gets one response, even if it contains information from more than one provider.

   For example, a client may ask OpenPegasus for a list of all the printers available to a system. Several providers may respond, one for each type of printer. OpenPegasus waits until all the providers respond and combines the information in one response to the client.

   If no provider can be reached, or none respond, OpenPegasus returns an error (`CIM_ERR_NOT_SUPPORTED`) to the client.

For a list of the standard CIM errors and other messages, see Chapter 7.

2.4 OpenPegasus Indications

You can receive a notification when an Event happens. An Event is the occurrence of a phenomenon of interest. An Event can be defined to indicate, for example, the occurrence of a disk-write error, a failed authentication attempt, or even a mouse click.

2.4.1 OpenPegasus Indication Architecture

The Indication Architecture is made up of the following components:

- Indication Generation
- Indication Subscription
- Indication Processing
- Indication Delivery
- Indication Consumption

The following section provides an overview of the OpenPegasus Indication Architecture.

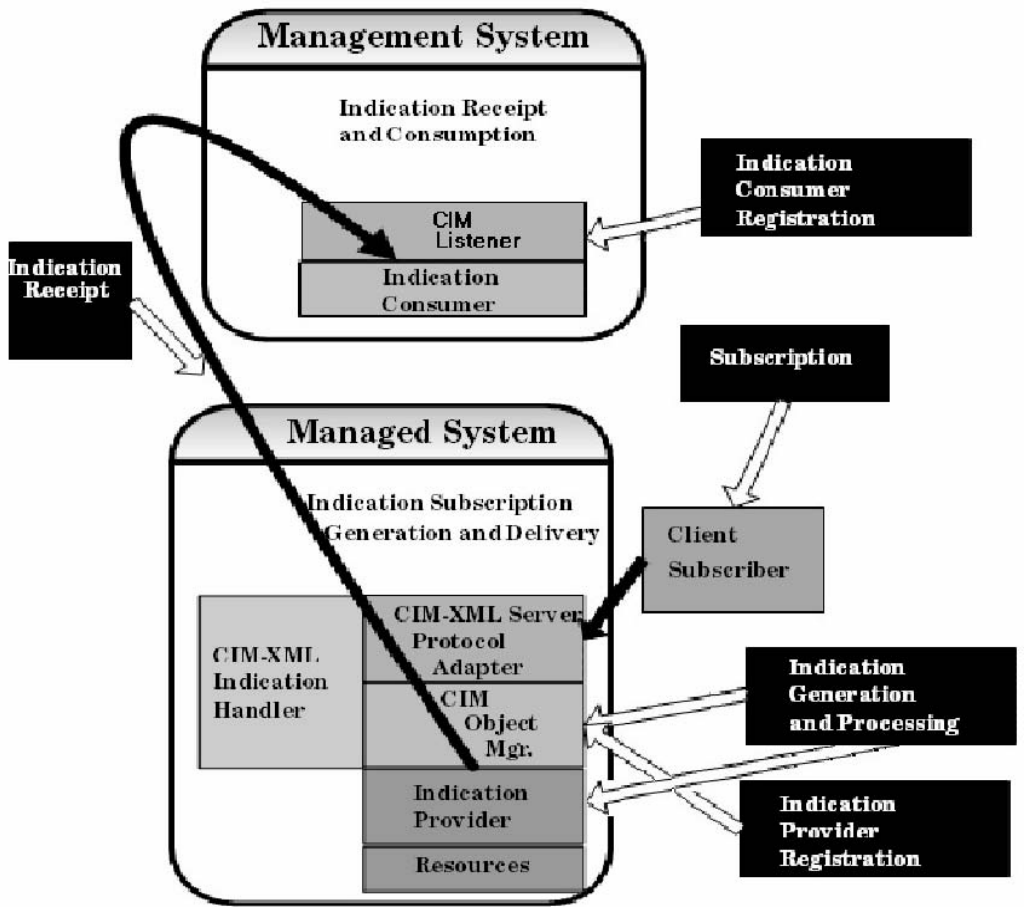

Figure 7: The Big Picture – OpenPegasus Architecture

2.4.1.1 Indication Generation

An Indication is the representation of the occurrence of an Event.

The abstract class `CIM_Indication` serves as the base class for all Indication classes. A CIM Indication Provider registers with the CIM Server to generate indications of one or more classes.

A CIM Indication Provider translates the detection of an Event into a CIM Indication and sends the Indication to the CIM Object Manager for further processing and delivery.

2.4.1.2 Indication Subscription

An Indication Subscriber is a CIM Client that issues CIM Operation requests to create instances of the `CIM_IndicationSubscription` class.

The Indication Filter specifies what indications should be sent (i.e., what Events are of interest), and the Listener Destination specifies where and how (i.e., by what protocol) indications should be sent. The Indication Service component of the CIM Object Manager is responsible for the processing of CIM Operations on the classes in the CIM Subscription Schema, management of subscriptions, and communication with the Indication Providers.

2.4.1.3 *Indication Processing*

The Indication Service component of the CIM Object Manager processes each generated indication to determine to which indication handlers, if any, the indication should be sent.

2.4.1.4 *Indication Delivery*

A CIM Indication Handler receives Indications, performs the mapping between the internal representation of a CIM Indication and the desired format and protocol, and sends the Indication to the designated destination.

— Note: A CIM Server may support multiple indication handler interfaces.

A CIM-XML Indication Handler, functioning as a CIM Client, uses the DMTF CIM-XML protocol to send an Indication to each specified destination.

When the CIM Server acts as a CIM Listener, the CIM Server functions as an HTTP Server to receive Indications as CIM Export Messages.

A CIM Message is a well-defined request or response data packet used to exchange information between CIM Applications. There are two types of CIM Messages: CIM Operation Messages and CIM Export Messages.

A CIM Operation Message is a CIM Message used to invoke an operation on the target CIM namespace.

A CIM Export Message is a CIM Message used to communicate information about a CIM namespace or element that is foreign to the target. A CIM Export Message is informational only and does not define an operation on the target CIM namespace or even imply the existence of a target namespace.

A CIM Listener receives CIM Export requests (e.g., Indications), coordinates the distribution of requests among one or more Consumers, and sends CIM Export responses.

— Note: For stand-alone CIM Listeners, the listener waits at an application-specific port to receive Indications (i.e., CIM Export Messages).

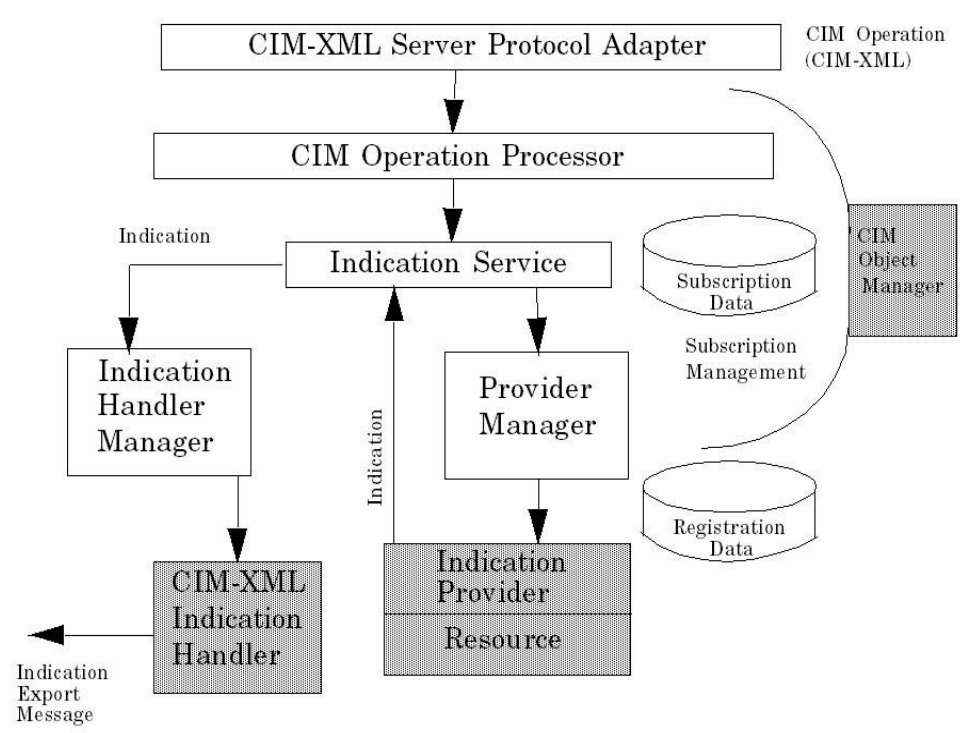

Figure 8: Indication Architecture

2.4.1.5 Indication Consumption

A CIM Indication Consumer "consumes" the CIM data (e.g., an Indication) encapsulated in a CIM Export Message. For example, a Consumer may store the Indication in an Event Database for further processing. An Indication Consumer registers with the CIM Listener to receive Indications.

See Chapter 7 for more information on troubleshooting WBEM Indications.

3 OpenPegasus Command Line Utilities

This chapter lists the commands, executable scripts, and daemon processes that are available with OpenPegasus.

The list is in alphabetical order. Refer to the following manual pages pages, for additional information. Availability of features may vary by platform. Please refer to the platform-specific documentation for additional details.

Name	Type	Reference Page
`cimauth`	Command	Yes
`cimconfig`	Command	Yes
`cimmof`	Command	Yes
`cimmofl`	Command	Yes
`cimprovider`	Command	Yes
`cimserver`	Command	Yes
`init_repository`	Script	No
`osinfo`	Command	Yes
`wbemexec`	Command	Yes

Table 1: Overview of OpenPegasus Commands and Scripts

cimauth

NAME

cimauth - add, modify, remove or list CIM user authorizations

SYNOPSIS

```
cimauth -a -u username -n namespace [ -R ] [ -W ]
cimauth -m -u username -n namespace [ -R ] [ -W ]
cimauth -r -u username [ -n namespace ]
cimauth -l
cimauth -h
cimauth --help
cimauth --version
```

DESCRIPTION

The cimauth command provides a command line interface to manage CIM user authorizations.

The first form of cimauth allows addition of authorizations to a specified user on a specified namespace. This form of the command can only be used to add authorizations to one user on one namespace at a time.

The second form of cimauth allows modification of existing authorizations for the specified user on a specified namespace. This form of the command can only be used to modify authorizations to one user on one namespace at a time.

If there are no authorizations specified with add and modify options, read authorization is assumed by default.

The third form of cimauth allows removal of authorizations for the specified user on a specified namespace. This form of the command can be used to remove authorizations of one user on one namespace or all the namespaces on which the user has authorizations. If no namespace is specified, then authorizations on all the namespaces for the specified user will be removed.

The last form of this command allows listing of the user name, namespace and authorizations.

Specifying no options with the cimauth will show the usage of the command.

OPTIONS

The cimauth command recognizes the following options:

`-a`	Indicates that authorizations to be added for a user on a namespace.
`-m`	Indicates that authorizations to be modified for a user on a namespace.
`-r`	Indicates that authorizations to be removed for a user on a namespace.
`-l`	Displays the authorizations of all the authorized users.
`-u` *username*	Indicates a specific user name.
`-n` *namespace*	Indicates a specific namespace.
`-R`	Indicates read authorization.
`-W`	Indicates write authorization.

491 `-h, --help` Display the help message

492 `--version` Display CIM Server version number

493 **EXTENDED DESCRIPTION**

494 This command does not configure or list CIM user password information (see cimuser).

495 This command is only relevant if the property enableNamespaceAuthorization is set to true,
496 which is not the default. (Set the enableNamespaceAuthorization property with the cimconfig
497 command.)

498 Running cimauth requires root permission.

499 Cimauth can only be used when the CIM Server is running.

500 **EXIT STATUS**

501 cimauth returns one of the following values:

502 0 Successful completion.

503 1 Error

504 When an error occurs, an explanatory error message is written to stderr and an error value 1 is
505 returned.

506 **EXAMPLES**

507 The following command adds read and write authorizations to user "guest" on namespace
508 "/root/system".

509 `cimauth -a -u guest -n root/system -R –W`
510

511 The following command adds read authorizations to user "guest" on namespace "/root/cimv2".

512 `cimauth -a -u guest -n root/cimv2`
513

514 The following command modifies authorizations of the user "guest" on namespace
515 "/root/system" to read only.

516 `cimauth -m -u guest -n root/system -R`
517

518 The following command removes the authorizations for user "guest" on namespace
519 "root/system".

520 `cimauth -r -u guest -n root/system`
521

522 The following command displays the list of authorized user names, namespaces and
523 authorizations.

524 `cimauth -l`

525 **SEE ALSO**

526 cimuser (1m)

cimconfig

NAME

cimconfig - get, set, unset or list CIMOM configuration properties.

SYNOPSIS

```
cimconfig -g name [ -c ][ -d ][ -p ]

cimconfig -s name=value [ -c ][ -p ]

cimconfig -u name [ -c ][ -p ]

cimconfig -l [ -c | -p ]

cimconfig -h

cimconfig --help

cimconfig --version
```

DESCRIPTION

The cimconfig command provides a command line interface to manage CIMOM configuration properties.

The first form of cimconfig allows to get current, planned and / or default value of the specified configuration property.

The second form allows to set the current value and / or planned value of the specified configuration property to the specified value.

The third form allows unsetting the current and / or planned values of the specified configuration property to its default value.

The last form of this command allows for all the configuration properties to be listed. Specifying the -c or -p options, will provide a listing of all the current or planned configuration property names and values.

Specifying no options with the cimconfig command will show the usage message.

OPTIONS

The cimconfig command recognizes the following options:

`-g`	Gets the current value of the specified configuration property.
`-g -c`	Gets the current value of the specified configuration property.
`-g -p`	Gets the planned value of the specified configuration property.
`-g -d`	Gets the default value of the specified configuration property. Returns an error when the CIMOM is not running.
`-s`	Indicates that a configuration property is to be added or updated by setting its current value to the specified value. Returns an error when the CIMOM is not running or the specified property is not dynamically updatable.

`-s -c`	Indicates that a configuration property is to be added or updated by setting its current value to the specified value. Returns an error when the CIMOM is not running or the specified property is not dynamically updatable.
`-s -p`	Indicates that a configuration property is to be added or updated by setting its planned value to the specified value.
`-u`	Indicates that the current value of the specified configuration property is to be reset to default. Returns an error when the CIMOM is not running or the specified property is not dynamically updatable.
`-u -c`	Indicates that the current value of the specified configuration property is to be reset to default. Returns an error when the CIMOM is not running or the specified property is not dynamically updatable.
`-u -p`	Indicates that the planned value of the specified configuration property is to be reset to default.
`-l`	Displays the name of all the configuration properties.
`-l -c`	Displays the name and value pair of all the current configuration properties.
`-l -p`	Displays the name and value pair of all the planned configuration properties.
`-h, --help`	Displays the help message
`--version`	Displays CIM Server version number

EXTENDED DESCRIPTION

This version of the cimconfig command does not return classified error codes. When an error occurs, it writes explanatory error message to stderr and returns an error value 1.

An operation using the “current” option changes the value immediately; an operation using the “planned” option takes effect the next time the CIM Server is started with the cimserver command.

For current values, the CIM Server must be running. For planned values, the CIM Server can be running or not.

OpenPegasus properties are listed in Chapter 5.5.

Running cimconfig requires root permission.

EXIT STATUS

cimconfig returns one of the following values:

0 Successful completion.

1 Error

When an error occurs, an explanatory error message is written to stderr and an error value 1 is returned.

EXAMPLES

The following commands get the current value for the configuration property "port".

```
cimconfig -g port
```

```
    cimconfig -g port -c
```

The following command gets the planned value for the configuration property "traceLevel".

```
    cimconfig -g traceLevel -p
```

The following commands set the current value of the property "traceLevel", to the new value "2".

```
cimconfig -s traceLevel=2
cimconfig -s traceLevel=2 –c
```

The following command sets the planned value of the property "traceLevel", to the new value "3".

```
    cimconfig -s traceLevel=3 –p
```

The following commands reset the current value of the property "traceLevel" to the default value.

```
    cimconfig -u traceLevel
    cimconfig -u traceLevel –c
```

The following command resets the planned value of the property "traceLevel" to the default value.

```
    cimconfig -u traceLevel –p
```

The following command displays the list of all the current configuration properties.

```
    cimconfig -l
```

The following command displays the list of all the current configuration properties and values.

```
    cimconfig -l –c
```

The following command displays the list of all the planned configuration properties and values.

```
    cimconfig -l –p
```

cimmof

NAME

cimmof, cimmofl - compile MOF files into the CIM Repository

SYNOPSIS

```
cimmof -h
cimmof [ -w ] [ -I path ] [ -n namespace ] file...file

cimmofl -h
cimmofl [ -w ] [ -I path ] [ -n namespace ] file...file
```

DESCRIPTION

The cimmof command is the command line interface to the Managed Object Format (MOF) Compiler. The MOF Compiler is a utility that compiles MOF files (using the MOF format defined by the DMTF CIM Specification) into CIM classes and instances that are stored in the CIM Repository.

cimmofl is a version of cimmof that does not use the CIM Server. This version the MOF compiler does only limited error checking, can incorrectly handle instance operations, and does not protect against concurrent access to the CIM Repository

Warning: Use of cimmofl can corrupt the CIM Server Repository. cimmofl should only be used under very controlled situations. cimmof is the recommended OpenPegasus MOF compiler.

The cimmof command can be used to compile MOF files at any time after installation. If no input file is specified, stdin is used as the input.

The MOF Compiler requires that the input MOF files be in the current directory or that a fully qualified path be given. To simplify the specification of multiple MOF files in the cimmof command line, the MOF Compiler allows compiling from files containing a list of MOF files using the include pragma (as shown below).

"#pragma include (""application.mof"")"

"#pragma include (""server.mof"")"

MOF files using the include pragma must be in the current directory or in a directory specified by the command line option.

The –n option can be used to specify a namespace in which the CIM classes and instances will be compiled. If this option is not specified, the default namespace is root/cimv2 (with the exception of provider registration schemas).

For provider registration schemas, if the –n option is not specified, the default namespace is root/PG_InterOp. If -n option is specified, the namespace specified must be root/PG_InterOp, otherwise, the error message "The requested operation is not supported." is returned. For provider MOFs, the namespace specified must match one of the namespaces specified in the PG_ProviderCapabilities class schema definition.

OPTIONS

The cimmof command recognizes the following options:

`-h` Display the command usage and the version number of the MOF Compiler. The version number is used to identify the release version of the MOF Compiler.

`-w` Suppress warning messages. When compiling the MOF files, if there are CIM elements (such as classes, instances, properties, or methods) defined in the MOF files which already exist in the CIM Repository, the cimmof command returns warning messages. The –w option can be used to suppress these warning messages.

`-I` *`path`* Specify the path to included MOF files. This path may be relative or absolute. If the input MOF file has include pragmas and the included files do not reside in the current directory, the –I directive must be used to specify a path to them on the cimmof command line.

`-n` *`namespace`* Override the default CIM Repository namespace. The namespacespecified must be a valid CIMnamespace name. For the definition of a valid CIM namespace name, refer to the Administrator's Guide.For provider registration schemas, the namespace specified must be root/PG_InterOp.

EXTENDED DESCRIPTION

Running cimmof requires root permission. Schema can only be loaded as local root, regardless of any authorizations done through cimauth. If namespace authorization is enabled, the user must also have Write authorization in the namespace.

Cimmof can only be used when the CIM Server is running.

EXIT STATUS

The cimmof command returns one of the following values:

0 Successful completion

1 Error

When an error occurs, an error message is written to stderr and an error value of 1 is returned.

DIAGNOSTICS

"Error trying to create Repository in path localhost:5988: Cannot connect to: localhost:5988 Failed to set DefaultNamespacePath."

The CIM Server is not running. Start the CIM Server with the cimserver command and re-run cimmof .

If the MOF Compiler detects an error in the MOF file while parsing the file, a parsing error is returned with the line number of the MOF file containing the error.

"Operation cannot be carried out since the specified superclass does not exist."

The MOF Compiler compiled a MOF file with superclasses that were not in the CIM Repository.

For a list of possible error messages that may be returned, refer to Chapter 7.2.

EXAMPLES

Compile a MOF file into the default namespace in the CIM Repository, issue the cimmof command with no options.

```
cimmof processInfo.mof
```

Compile the MOF files into the 'root/application' namespace.

```
cimmof -nroot/application test1.mof test2.mof
```

Compile the MOF file defined in the directory ./MOF with the name CIMSchema25.mof, and containing include pragmas for other MOF files also in the ./MOF directory.

```
cimmof -w -I./MOF MOF/CIMSchema25.mof
```

List the arguments to the cimmof command and display the version of the MOF Compiler.

```
cimmof -h
```

SEE ALSO

cimserver

cimprovider

NAME

cimprovider - disable, enable, remove or list registered CIM providers or CIM provider modules and module status.

SYNOPSIS

```
cimprovider -d -m module

cimprovider -e -m module

cimprovider -r -m module [ -p provider ]

cimprovider -l [ -s | -m module ]

cimprovider -h

cimprovider --help

cimprovider --version
```

DESCRIPTION

The cimprovider command provides a command line interface to disable, enable, unregister, and list registered CIM providers. If a CIM provider is disabled, the CIM Server rejects any requests to the provider. If a CIM provider is enabled, the CIM Server forwards requests to the provider. And if a CIM provider is unregistered, the CIM Server will no longer have any information about the provider.

In order to use the cimprovider command, cimserver has to be running and the specified provider module (a grouping of providers in the same shared library) or provider has to be registered with WBEM Services.

The first form of cimprovider disables the specified provider module. When a specified provider module is in the disabled state, any new requests to the providers that are contained in the specified provider module will be rejected.

The second form of cimprovider enables the providers that are contained in the specified provider module. The providers that are contained in the specified provider module are now ready to accept new request.

The third form of cimprovider removes (un-registers) the specified provider module and all of its contained providers or the specified provider in the specified provider module. Once removed a provider or provider module, must be re-registered (typically by loading its registration schema via the cimmof command).

The last form of cimprovider lists all the registered provider modules and module status or all the providers in the specified provider module. To list all providers in all modules, issue a cimprovider -l command, followed by cimprovider -l -m for each listed module.

Specifying no options with the cimprovider command displays the command usage.

OPTIONS

The cimprovider command recognizes the following options:

`-d`	Disables the specified CIM provider module. If user(s) try to disable a module that is already disabled, an error message is returned and no action is taken.
`-e`	Enables the specified CIM provider module. If user(s) try to enable a module that is already enabled or try to enable a module that is disabling, an error message is returned and no action is taken.
`-r`	Removes the specified provider module and all of its contained providers. If provider is specified, removes the specified provider in the specified provider module (not affecting any other providers in that module).
`-l`	Displays all the registered provider modules.
`-m` *`module`*	Specifies the provider module for the operation.
`-p` *`provider`*	Specifies the provider for the operation.
`-s`	Displays the status of provider modules.
`-h, --help`	Display this help message.
`--version`	Display CIM Server version number

EXTENDED DESCRIPTION

The -l option for this command can be executed by any user(s). All other options require superuser permissions.

This command disables, enables, or removes one CIM provider module or CIM provider at a time.

The list option can be executed by any user. Using any other options of cimprovider requires root permission.

Cimprovider can only be used when the CIM Server is running

EXIT STATUS

When an error occurs, an error message is written to stderr and an error value 1 is returned. The following return values are returned:

0 Successful completion

1 Error

EXAMPLES

Disable provider module "OperatingSystemProvider" and all of its contained providers (placing them in a stopped state).

```
cimprovider -d -m OperatingSystemProvider
```

Enable provider module "OperatingSystemProvider" and all of its contained providers (placing them in a OK state).

```
cimprovider -e -m OperatingSystemProvider
```

Remove (un-registers) the "OperatingSystemProvider" provider module and all of its contained providers.

```
cimprovider -r -m OperatingSystemProvider
```

Remove (un-registers) the "PG_OperatingSystemProvider" provider that contains in the "OperatingSystemProvider" provider module.

```
cimprovider -r -m OperatingSystemProvider –p
    PG_OperatingSystemProvider
```

List the registered provider modules.

```
cimprovider -l
```

List the registered provider modules and their status (such as OK, Stopping, Stopped).

```
cimprovider -l -s
```

List the registered providers which contained in the "OperatingSystemProvider" provider module.

```
cimprovider -l -m OperatingSystemProvider
```

SEE ALSO

Cimmof, cimserver

cimserver

NAME

cimserver - start or stop the CIM Server; display the version number of the CIM Server

SYNOPSIS

```
cimserver [ configProperty=value ] ...

cimserver -s [ shutdownTimeout=value ]

cimserver -v

cimserver -h
```

DESCRIPTION

The cimserver command provides a command line interface to stop and start the CIM Server, as well as to display the version number of the CIM Server.

After installation, the CIM Server must be started using the cimserver command. If the system is rebooted, the CIM Server will automatically restart, with the exception of the case where the CIM Server was shutdown prior to the reboot. Generally, once the CIM Server is started, it is expected to be always running and ready to serve CIM requests. However, if the CIM Server must be stopped and restarted, the cimserver command can be used to shutdown the CIM Server gracefully and restart it at a later time.

Starting the CIM Server

Issuing the cimserver command without any options starts the CIM Server process.

When starting the CIM Server using the cimserver command, the configProperty=value syntax can be used to set the configuration properties to be used by the CIM Server. It is important to know that the values specified in the .B cimserver command apply only to the current CIM Server process that gets started. The default values for the configuration properties do not change. For a list of the CIM Server configuration properties, see the man page for the cimconfig command or the HP WBEM Services Administrator's Guide.

Shutting down the CIM Server

Issuing the cimserver command with the –s option stops the CIM Server. Optionally, a timeout value can be specified by setting the shutdownTimeout configuration property.

Under normal operation, CIM Server should be able to be shutdown fairly quickly without problem. There are, however, situations that may prevent CIM Server from shutting down within a reasonable amount of time. For example, a provider that is not responding to requests, or a provider that is servicing a long-running CIM request. To handle such situations and to ensure that the CIM Server can be shutdown without having the user wait a long period of time (or indefinitely), a shutdown timeout value is used.

The shutdown timeout value is the maximum amount of time (in seconds) the user is willing to wait for the CIM Server to complete all the outstanding CIM operation requests before shutting down the CIM Server. If the specified shutdown timeout period expires, the CIM Server will be shutdown even if there are CIM operations in progress. The shutdown timeout value is a CIM Server configuration property 9shutdownTimeout) that can be changed using the cimconfig

command. The default shutdown timeout value is 10 seconds. A timeout value (in seconds) can be specified in the cimserver command to shutdown the CIM Server using the shutdownTimeout=value syntax. This overrides the default shutdown timeout value. The minimum timeout value is 2 seconds. While CIM Server is shutting down, a client connection request will result in a connection error (the same as if the CIM Server were not running). For clients who have already established a connection to the CIM Server, new CIM requests will be rejected with a CIM error indicating that the CIM Server is shutting down. When a client receives a response containing a CIM error indicating that the CIM Server is shutting down, it should close the connection and reconnect to CIM Server at a l

OPTIONS

The cimserver command recognizes the following options:

-v Display the version number of the CIM Server.

-h Display the command usage.

-s Stop the CIM Server.

shutdownTimeout= value

Specify the timeout value for shutting down the CIM Server. This can only be used I conjunction with the -s option. The minimum timeout value is 2 seconds. If this is not specified, the default configurable timeout value will be used.

configProperty=value

Set the value for the specified configuration property to be used in starting the CIM Server.

EXTENDED DESCRIPTION

Using cimserver to start or stop the CIM Server requires root permission.

Cimserver can only be used to stop the CIM Server or get a version number when the CIM Server is running.

If the cimserver command is used start the CIM Server when it is already running, no action is taken

EXIT STATUS

The cimserver command returns one of the following values:

0 Success

1 Error

When an error occurs, an error message is written to stderr and an error value of 1 is returned.

DIAGNOSTICS

"unable to connect to CIM Server. CIM Server may not be running."

The cimserver command was issued to stop the CIM Server when CIM Server was not running. An exit status value of 0 is returned.

"Error: Bind failed: Failed to bind socket."

The cimserver command was issued to start the CIM Server and the CIM Server was already running. An exit status value of 0 is returned.

"Unable to start CIMServer. CIMServer is already running."

An aempt was made to start the CIM Server when it was already running.

EXAMPLES

Stop the CIM Server with the default timeout value of 10 seconds.

```
cimserver -s
```

Stop the CIM Server with a timeout value of 5 seconds.

```
cimserver -s shutdownTimeout=5
```

Start the CIM Server.

```
cimserver
```

Start the CIM Server with the configuration property enableNamespaceAuthorization set to true.

```
cimserver enableNamespaceAuthorization=true
```

Display the version number of the CIM Server. This version number is used to identify the release version of the CIM Server in the HP WBEM Services product.

```
cimserver -v
```

Display the command usage.

```
cimserver -h
```

SEE ALSO

cimconfig

init_repository

NAME

init_repository – initializes the repository

SYNOPSIS

DESCRIPTION

OPTIONS

EXTENDED DESCRIPTION

If the repository is moved or corrupted, first try to restore it from backup. If that does not work, use the init_repository script to restore it to the state it was in at installation. Everything that was entered after install will be lost, so it will be necessary to re-install any providers that have been added.

Running init_repository requires root permission.

init_repository can only be used when the CIM Server is running.

EXIT STATUS

EXAMPLES

SEE ALSO

osinfo

NAME

osinfo - returns infomation about the operating system running on a host system.

SYNOPSIS

`osinfo [-s] [-h hostname] [-p portnumber] [-u username] [-w password] [-t timeout] [-c]`

DESCRIPTION

The osinfo command displays information about the operating system running on either the local or a designated host system. osinfo is a CIM Client application and requires a CIM Server to be running on the target system.

If a property value is unavailable, the value "UNKNOWN" will be displayed.

By default, information about the operating system running on the local system is displayed.

OPERANDS

osinfo recognizes the following options:

`-s`	Enable the use of the SSL protocol between osinfo and the CIM server. The -s option should be specified if the CIM Server on the specified hostname/portnumber expects clients to connect using HTTPS. This option is ignored if neither hostname nor portnumber is specified.
`-h hostname`	Connect to the CIM Server on the specified hostname. If this option is not specified, osinfo connects to the local host.
`-p portnumber`	Connect to the CIM Server on the specified portnumber. If this option is not specified, osinfo connects to the default port for the wbem-http service, or if the -s option is specified, to the default port of the wbem-https service.
`-u username`	Connect as the specified username. If username is not specified, the current logged in user is used for authentication. This option is ignored if neither hostname nor portnumber is specified.
`-w password`	Authenticate the connection using the specified password. If the password is not specified, the user is prompted for a password. This option is ignored if neither hostname nor portnumber is specified.
`-t timeout`	Defines the number of milliseconds to wait for a response before "timing out" the operation. The default timeout value is 2000. The timeout value must be an integer value greater than 0.
`-c`	Use the CIM formats for DateTime and SystemUpTime values (not the formatting done by default). As specified by the DMTF, the CIMDateTime format is yyyymmddhhmmss.mmmmmmsutc, where yyyy is a 4-digit year, mm is the month, dd is the day, hh is the hour on a 24-hour clock, mm is the minute, ss is the second, mmmmmm is the number of microseconds, s is a "+" or "-" indicating the sign of the UTC (Universal

Time Code) correction field (since the DateTime is returned in the local time zone of the system), and utc is the offset from UTC in minutes.

EXTENDED DESCRIPTION

By default, the information is formatted for display in English with uptime displayed in days, hours, minutes, and seconds.

The command can be used for for troubleshooting, to see whether OpenPegasus can return a simple request about its own system.

Any user can execute the osinfo command; root permission is not required.

osinfo can only be used when the CIM Server is running.

EXIT STATUS

When an error occurs, an error message is written to stderr and an appropriate value is returned. The following values are returned:

0 Success

1 Error

DIAGNOSTICS

osinfo error: Cannot connect to: <hostname >: <portnumber >

There was a failure attempting to connect to the CIM Server running on the target host system.

EXAMPLES

Run the default osinfo command on bryce.

```
osinfo
```

OperatingSystem Information

Host: bryce.xyz.com

Name: Linux

Version: B.11.00

UserLicense: Unlimited user license

Number of Users: 4 users

Number of Processes: 71 processes

OSCapability: 32 bit

LastBootTime: Sep 24, 2001 9:16:6 (-0700)

LocalDateTime: Feb 9, 2003 18:59:43 (-0700)

SystemUpTime: 43497817 seconds = 503 days, 10 hrs, 43 mins, 37 secs

From bryce, display information about the operating system running on bodie.

```
osinfo -s -h bodie -u guest -w guest
```

OperatingSystem Information

Host: bodie.xyz.com

Name: Linux

Version: B.11.00

UserLicense: Unlimited user license

Number of Users: 0 users

Number of Processes: 67 processes

OSCapability: 32 bit

LastBootTime: May 28, 2002 11:10:25 (-0700)

LocalDateTime: Feb 10, 2003 18:18:5 (-0700)

SystemUpTime: 22320460 seconds = 258 days, 8 hrs, 7 mins, 40 secs

Host: bryce.xyz.com

From bryce, display information about the operating system running on bodie, but this time specify a ridiculously short timeout value.

```
osinfo -s -h bodie -u guest -w guest -t1
osinfo error: connection timed out
```

wbemexec

NAME

wbemexec - submit a CIM operation to the CIM Server for execution

SYNOPSIS

```
wbemexec [ -h hostname ] [ -p portnumber ] [ -v httpversion ] [ -m
httpmethod ] [ -t timeout ] [ -u username ] [ -w password ] [ -d
debugoption ] [ -s ] [ inputfile ]
```

DESCRIPTION

The wbemexec command provides a command line interface to the CIM Server. The input to the command consists of a CIM request encoded in XML. The request is submitted to the CIM Server for execution. The result of the operation is returned to stdout, and consists of the CIM response encoded in XML.

By default, the operation is executed on the local host, using the default port (5988), and the request is sent as an HTTP/1.1 request, using the HTTP POST method. By default, wbemexec waits 20000 milliseconds (20 seconds) on sending a request, then times out if a response hasn't been received. The -h option allows the user to specify a different host. The -p option allows the user to specify a different port number. The -v option allows the user to specify a different HTTP version for the request. The -m option allows the user to specify a different HTTP method (i.e. M-POST) for the request. The -t option allows the user to specify, in milliseconds, a different timeout value for the request. The -u and -w options allow the user to specify a username and password to use for authentication of the user and authorization of the operation. By default, stdin is used as the input, if no input file is specified. The -s option enables the SSL protocol between wbemexec and the CIM server. The -d option may be used to specify that debug information be included in the output.

OPTIONS

wbemexec recognizes the following options:

`-h hostname` Use the specified host. A CIM Server must be running on the specified host. If this option is not specified, wbemexec will connect to the local host and authenticate itself.

`-p portnumber` Use the specified port number. The port number must be the port number on which the CIM Server is running on the specified host. If no port number is specified, wbemexec first attempts to connect to the CIM Server on the default port for wbem-http service; if that fails, it tries the default port for wbem-https.

`-v httpversion` Use the specified HTTP version for the request. The version must be "1.0" or "1.1". The 1.0 version may not be specified if the M-POST method is specified. By default, the request is sent as an HTTP/1.1 request, using the HTTP M-POST method.

`-m httpmethod` Use the specified HTTP method for the request. The method must be "POST" or "M-POST". The M-POST method may not be specified if the 1.0 version is specified.

`-t` *`timeout`* Wait the specified number of milliseconds on sending a request, before timing out if no response has been received. The timeout value must be an integer value greater than 0. The default value is 20 seconds

`-u` *`username`* Authorize the operation using the specified username. If username is not specified, the current logged in user will be used for authentication and authorization.

`-w` *`password`* Authorize the operation using the specified password. If the password is not specified and the remote host requests authentication, the user will be prompted for a password.

`-s` Enable the use of the SSL protocol between wbemexec and the CIM server.

`-d` *`debugoption`* Include specified debug information in the output.

1 Include HTTP-encapsulated XML request in output.

2 Include HTTP header of response in output.

OPERANDS

`inputfile` The name of a file containing an XML encoded CIM operation request. If this operand is omitted, input is read from stdn.

EXTENDED DESCRIPTION

wbemexec can only be used when the CIM Server is running.

EXIT STATUS

When an error occurs, an explanatory error message is written to stderr and an appropriate value is returned. The following return values are returned:

0 Success

1 Error

EXAMPLES

Submit an XML request contained in the file cimrequest.xml to the CIM Server running on the local host on the default port (5988), using the username guest and password guest for authentication and authorization:

```
wbemexec -u guest -w guest cimrequest.xml
```

Submit an XML request contained in the file cimrequest.xml to the CIM Server running on the host xyzserver on port 49152, using the username guest and password guest for authentication and authorization:

```
wbemexec -h xyzserver -p 49152 -u guest -w guest cimrequest.xml
```

Submit an XML request contained in the file cimrequest.xml to the CIM Server running on the local host on the default port (5988), including both the HTTP-encapsulated XML request, and the HTTP header portion of the response in the output:

```
wbemexec -d 1 -d 2 cimrequest.xml
```

4 Example of a Client Request

This chapter gives an example of a client request and the response.

The request is for the `EnumerateInstances` operation on the `PG_OperatingSystem` class.

Requests and responses are encoded in XML. For more information about XML, see the DMTF CIM-XML specification.

The following information is in a table format. The first column has line numbers for the actual request and response. The middle column may group several related lines. The right-hand column is a comment on the corresponding middle column.

The request is first; it is 16 lines long. Next is the response; it is actually 172 lines long, but lines 81 to 170 were cut for brevity.

4.1 Example Request

Table 2: EnumerateInstances Request for PG_OperatingSystem Class

1	`<?xml version=”1.0” ?>`	Begin specifying that this is an XML-encoded CIM message (see ending at line 15 and 16).
2	`<CIM CIMVERSION=”2.0” DTDVERSION=”2.0”>`	
3	`  <MESSAGE ID=”51000” PROTOCOLVERSION=”1.0”>`	
4	`    <SIMPLEREQ>`	This is a simple request for the operation: method `EnumerateInstances`.
5	`      <IMETHODCALL NAME=”EnumerateInstances”>`	
6	`        <LOCALNAMESPACEPATH>`	Line 6 begins (and 9 ends) specifying the `/root/cimv2` namespace for the CIM Operation.
7	`          <NAMESPACE NAME=”root”/>`	
8	`          <NAMESPACE NAME=”cimv2”/>`	
9	`        </LOCALNAMESPACEPATH>`	
10	`        <IPARAMVALUE NAME=”ClassName”>`	Line 10 begins (and 12 ends) specifying the class name (required) for `EnumerateInstances`: `PG_OperatingSystem`.
11	`          <CLASSNAME NAME=”PG_OperatingSystem”/>`	
12	`        </IPARAMVALUE>`	
13	`      </IMETHODCALL>`	Ending of the method call and simple request.
14	`    </SIMPLEREQ>`	
15	`  </MESSAGE>`	Ending of the CIM Operation request message.
16	`</CIM>`	

Lines 1 through 3

This is checked when the request comes to the HTTP Server. At this point, several things have to happen in order to continue:

- The client must be able to connect to the system on the authorized port.
- The CIM Server must be running.
- The user/password pair must pass authorization.
- The request must have a properly formed header.
- When the request is parsed, it must not contain XML errors.

Lines 4 and 5

At this point, OpenPegasus considers the operation that is requested. If it is a supported operation, the process continues.

Lines 6 through 9

Two criteria must be met in order to continue:

- This namespace must be valid.
- If the `enableNamespaceAuthorization` property is enabled, this user must be authorized to access this namespace

Lines 10 through 12

The `classname` must exist, and it must have a provider registered. The provider must respond to the request. Here, the OS Provider is registered for the `PG_OperatingSystem` class. Checking the provider documentation, you can see that it supports the `EnumerateInstances` method.

Now it is up to the provider to process the request and send a response. If the resource does not respond, OpenPegasus will send a message to the client. If the resource sends its own error, OpenPegasus will pass this on to the client in its response. Often, these messages will be appended to a standard CIM error.

4.2 Example Response

The table below shows the response to the request to `EnumerateInstances` for `PG_OperatingSystem`.

The return value is a named instance. Named instances include both `INSTANCENAME` (the instance with its key properties) and `INSTANCE` (*all* the properties). Because this instance has so many properties, some of them have been cut from the example text.

1159 **Table 3: EnumerateInstances Response for PG_OperatingSystem Class**

1	`<?xml version =”1.0” encoding=”utf-8”?>`	Lines 1–3 indicate this is an XML-encoded message (see ending at lines 171 and 172).
2	`<CIM CIMVERSION=”2.0” DTDVERSION=”2.0”>`	
3	`  <MESSAGE ID=”51000” PROTOCOLVERSION=”1.0”`	
4	`    <SIMPLERSP>`	This is simple response to the `EnumerateInstances` method.
5	`    <IMETHODRESPONSE NAME=”EnumerateInstances”>`	
6	`      <IRETURNVALUE>`	Return value is named instance (*all* properties).
7	`        <VALUE.NAMEDINSTANCE>`	
8	`          <INSTANCENAME CLASSNAME=”PG_OperatingSystem”>`	Begin *keys* of class name.
9	`            <KEYBINDING NAME=”CreationClassName”>`	One key for this instance. It is `CreationClassName`, a string, and its value is “`CIM_OperatingSystem`”.
10	`            <KEYVALUE VALUETYPE=”string”>`	
11	`            CIM_OperatingSystem`	
12	`            </KEYVALUE>`	
13	`            </KEYBINDING>`	
14	`            <KEYBINDING NAME=”CSCreationClassName”>`	Next key is `CSCreationClassName`, a string, with value “`CIM_UnitaryComputerSystem`”.
15	`            <KEYVALUE VALUETYPE=”string”>`	
16	`            CIM_UnitaryComputerSystem`	
17	`            </KEYVALUE>`	
18	`            </KEYBINDING>`	
19	`            <KEYBINDING NAME=”CSName”>`	The next key is `CSName`, also a string, with value “`mycomputer.xyz.com`”.
20	`            <KEYVALUE VALUETYPE=”string”>`	
21	`            Mycomputer.xyz.com`	
22	`            </KEYVALUE>`	
23	`            </KEYBINDING>`	
24	`            <KEYBINDING NAME=”Name”>`	The next key is `Name`, also a string, with the value of “`Linux`”.
25	`            <KEYVALUE VALUETYPE=”string”>`	
26	`            Linux`	
27	`            </KEYVALUE>`	
28	`            </KEYBINDING>`	
29	`          </INSTANCENAME>`	End of keys for instance.
30	`            <INSTANCE CLASSNAME=“PG_OperatingSystem”>`	Begin all properties of instance.

31	<PROPERTY NAME=”CSCreationClassName” TYPE=”string”>	First key property is CSCreationClassName, a string, with value “CIM_UnitaryComputerSystem”.
32	<VALUE> CIM_UnitaryComputerSystem </VALUE>	
33	</PROPERTY>	
34	<PROPERTY NAME=”CSName” TYPE=”string”>	Next key property.
35	<VALUE> mycomputer.xyz.com </VALUE>	
36	</PROPERTY>	
37	<PROPERTY NAME=”CreationClassName” TYPE=”string”>	Next key property.
38	<VALUE> CIM_OperatingSystem </VALUE>	
39	</PROPERTY>	
40	<PROPERTY NAME=”Name” TYPE=”string”>	Next key property.
41	<VALUE> HPUX </VALUE>	
42	</PROPERTY>	
43	<PROPERTY NAME=”Caption” TYPE=”string”>	Next property.
44	<VALUE> The current operating System </VALUE>	
45	</PROPERTY>	
46	<PROPERTY NAME=”Description” TYPE=”string”>	Next property.
47	<VALUE> This instance reflects the Operating System on which the CIMOM is executing (as distinguished from instances of other installed operating systems that could be run). </VALUE>	
48	</PROPERTY>	
49	<PROPERTY NAME=”Status” TYPE=”string”>	Next property.
50	<VALUE> Unknown </VALUE>	
51	</PROPERTY>	
52	<PROPERTY NAME=”OSType” TYPE=”unint16”>	Next property (unsigned integer, 16-bit) (DMTF specifies that 8 = HP-UX).
53	<VALUE> 8 </VALUE>	
54	</PROPERTY>	
55	<PROPERTY NAME=”LastBootUpTime” TYPE=”datetime”>	Next property

56	<VALUE> 2010924091618.000000-420 </VALUE>	(datetime datatype).
57	</PROPERTY>	
58	<PROPERTY NAME="CurrentTimeZone" TYPE="sint16">	Next property (signed integer, 16-bit).
59	<VALUE> -420 </VALUE>	
60	</PROPERTY>	
61 - 150	... Several properties of the instance were removed from this example. ...	
151	</INSTANCE>	End of this instance's properties.
152	</VALUE.NAMEDINSTANCE>	End of named instance.
153	</IRETURNVALUE>	End return value.
154	</IMETHODRESPONSE>	End method response.
155	</SIMPLERSP>	End simple response.
156	</MESSAGE>	End message.
157	</CIM>	End xmlCIM message.

5 Installing and Setting Up OpenPegasus

This chapter describes the prerequisites and what system administrators should do before they actually use OpenPegasus.

— Note: Although certain OpenPegasus file locations are configurable, changing the product default file location settings is not recommended.

Installation instructions will vary by platform; you are advised to consult your product documentation for details. Instructions for installing the OpenPegasus 2.4 Linux RPMs can be found on the OpenPegasus Linux RPM download page. Installation is mostly automatic; you do not need to specify configuration options for the installation to complete successfully.

After installing, you can set some options to define the properties of OpenPegasus itself.

Once OpenPegasus is running with the properties you want, back up the repository directories and configuration files. Consult Appendix D for directory and file locations.

5.1 Certificate and Repository Backup

It is recommended that you back up the appropriate OpenPegasus directory structure on a regular basis. If these files are deleted, moved, or corrupted, you need to restore from the backup.

If you don't have a backup file for the SSL certificate files, you will need to re-install OpenPegasus or re-create certificates using the OpenSSL toolkit. Refer to www.openssl.org/docs for information on adding back the certificates you used since the last time you installed.

If you do not have backup files for the repository, you can only return to the state of the last install. You will lose everything that was added since the last time you installed. You will have to reinstall any providers you added. Any data will be lost.

5.2 Before Starting OpenPegasus

For OpenPegasus to work, the following must be present:

- Configured Ports

  By default, OpenPegasus HTTP Server listens for SSL (Secure Sockets Layer) encrypted communications on the HTTPS (secure) port, 5989. If you are sure your environment is secure, you could set the configuration so the server will listen at the HTTP (not SSL) port, 5988. (See Chapter 6.) Ports 5988 (wbem_http) and 5989 (wbem_https) are specified by the DMTF and are registered with the Internet Assigned Numbers Authority (IANA) at www.iana.org.

When OpenPegasus receives an HTTP request over the configured port, it checks user authentication, parses the request, looks up the resource, and contacts the registered provider if applicable. The provider sends a response to OpenPegasus, and OpenPegasus sends it back to the client through this port.

— Note: On some platforms (e.g., Linux and HP-UX), OpenPegasus supports the use of UNIX domain sockets for local (same-system) connections. This allows connections to be established using a domain socket rather than a TCP port. In high-threat environments, it may be desirable to disable all ports. Support for this option allows the OpenPegasus CIM Server to continue to receive and process requests from local OpenPegasus CIM Clients.

- OpenPegasus Infrastructure

  Installation instructions will vary by platform. You are advised to consult your product documentation for details. Instructions for installing the OpenPegasus 2.4 Linux RPMs can be found on the OpenPegasus Linux RPM download page.

  — Note: If you already have OpenPegasus installed, check your product documentation before uninstalling or re-installing. You could remove all the files associated with OpenPegasus and make all your providers unavailable.

5.2.1 Providers Included with OpenPegasus

The list of Providers packaged with OpenPegasus will vary by vendor. Consult your product documentation for the list of Providers that are automatically installed with OpenPegasus.

The following section details the production-level providers that are included as part of the OpenPegasus source.

— Note: To see a list of provider modules on your system, use the `cimprovider -l` command. To see a provider in a particular module, use `cimprovider -l -m <modulename>`.

- Operating System Provider

  The Operating System Provider makes available operating system information such as operating system type, version, last boot-up time, local date and time, number of users, swap space size, and free physical memory. This provider is for use by clients as part of a basic understanding of the identity of the managed system on which it is running (typically a server or appliance). The Operating System Provider makes use of the `CIM_OperatingSystem` class and `PG_OperatingSystem` subclass.

  — `SystemUpTime` is a convenience property. It provides direct access to this value, *versus* having client/providers calculate the value from `LastBootUpTime` and the `LocalDateTime`.

  — `OperatingSystemCapability` indicates whether the OS itself is 32-bit or 64-bit-capable.

— Note: This provider does not support the `Reboot` and `Shutdown` methods of the `CIM_OperatingSystem` class. The `PG_OperatingSystem` subclass adds the `SystemUpTime` and `OperatingSystemCapability` properties.

- Computer System Provider

  The Computer System Provider makes available basic computer system information such as computer name, status, and administrator contact information.

  This provider is for use by clients as part of a basic understanding of the identity of the Managed System on which it is running (typically a server or appliance).

  This Computer System Provider makes use of the `CIM_ComputerSystem`, `CIM_UnitaryComputerSystem`, and `PG_ComputerSystem` classes.

  The `PG_ComputerSystem` subclass of `CIM_UnitaryComputerSystem` adds the `PrimaryOwnerPager`, `SecondaryOwnerName`, `SecondaryOwnerContact`, `SecondaryOwnerPager`, `SerialNumber`, and `IdentificationNumber`.

  — `PrimaryOwnerPager` is the pager number for the primary system owner.

  — `SecondaryOwnerName`, `SecondaryOwnerContact`, and `SecondaryOwnerPager` are the name, phone number, and pager number of the system's secondary owner respectively.

  — `SerialNumber` is the system's serial number.

  — `IdentificationNumber` is the corporate asset number of the system.

— Note: The `PG_ComputerSystem` follows the industry convention of naming `CIM_UnitaryComputerSystem` subclasses without including "Unitary" in the class name. This practice is an exception to the normal practice used for creating non-DMTF-defined subclasses (simply changing the superclass' prefix from `CIM_` to some organization-specific string).

- Process Providers

  The Process Providers make available basic UNIX process information, such as name of the executable image, process ID, priority, execution state, and various process resource utilization statistics.

  Client applications can use these Providers to give clients an understanding of the processes running on the Managed System within the context of its operating system.

  In addition to implementing the properties of `CIM_Process`, the Process Providers implement the properties of `PG_UnixProcess` and `PG_UnixProcessStatisticalInformation`.

— Note: To see a list of provider modules on your system, use the `cimprovider -l` command. To see a provider in a particular module, use `cimprovider -l -m <modulename>`.

5.2.2 Clients Included with OpenPegasus

The OpenPegasus product includes a simple client you can use to exercise the infrastructure. After installing the infrastructure and the bundled providers, you can run it to check that things are running smoothly.

The `osinfo` command invokes a client request to the included Operating System Provider. If all is well, you will receive a formatted text reply that looks something like the following:

```
Operating System Information
Host: zambezi.cup.hp.com
Name: Red Hat Enterprise Linux AS
    (2.6.9-1.648_ELsmp #1 SMP Tue Oct 26 12:55:36 EDT 2004)
Version: 3.90 (Nathant)
UserLicense: Unlimited user license
Number of Users: 1 user
Number of Processes: 79 processes
OSCapability: 32-bit
LastBootTime: Jan 10, 2005 17:26:19 (-0800)
LocalDateTime: Jan 18, 2005 23:17:54 (-0800)
SystemUpTime: 712295 seconds = 8 days, 5 hrs, 51 mins, 35 secs
```

See Chapter 4 for the request sent by `osinfo`, and the unformatted response.

5.3 Starting and Stopping OpenPegasus

The CIM Server is designed to be always running and ready to serve CIM requests, unless a user command stops it.

`cimserver` is an OpenPegasus daemon process; it is designed to restart automatically when the operating system reboots, and stay running as long as the operating system is running.

The following table describes the OS-specific command for determining whether the CIM Server is running.

Platform	**Command**
Linux	`ps -ef|grep cimserver`
HP-UX	`ps -ef|grep cimserver`

5.3.1 The cimserver Command

If you stop the CIM Server, restart it with the `cimserver` command.

Entering `cimserver` with no options starts the CIM Server on the system where the command is issued. Use the `-s` option to stop the CIM Server, the `-v` option to see the version number of the CIM Server, and the `-h` option for help on the command's syntax.

On startup, you have the option of including parameters to specify configuration property values, but these settings will last only as long as the current process. Use the format `<propertyName>=<value>`.

For a list of properties and their default value, see the reference page for the `cimconfig` command.

One configuration value, `shutdownTimeout`, is only valid with `cimserver -s`, the shutdown form. (And it is the only property that the stop form can use.) That timeout value is only used for that particular shutdown. Specify the amount of time for a graceful shutdown; after timeout passes, the CIM Server will kill all processes, finished or not.

You must be a privileged user (have root permissions on the local system) to use the `cimserver` command.

5.4 Maintaining the Repository

OpenPegasus keeps definitions of the data about managed objects and their providers in its repository.

Four namespaces install with OpenPegasus. Others may be added by clients and providers. The four that are automatically installed are:

`root`	The root namespace exists to conform to the DMTF specifications.
`root/cimv2`	The standard CIM Schemas go here. Also, the schemas for the bundled providers.
`root/PG_Interop`	This is for provider registration. This space is reserved exclusively for providers, and all providers must register here. (See the `cimprovider` reference page.)
`root/PG_Internal`	This space is reserved for use by OpenPegasus only.

The CIM Server should be shutdown prior to performing a backup of the repository files. It is important to schedule backups of the repository directories and files. If the repository is moved or lost or it becomes corrupted, restore the files you backed up.

If you cannot restore the files, the `init_repository` script will restore the files to the way they were when you first installed OpenPegasus. The Providers that installed with OpenPegasus will be intact. However, any managed objects, providers, indication subscriptions, or namespaces that you added since you first installed OpenPegasus will be gone. You will need to re-register (perhaps reinstall) all the added providers.

To run the script, enter the following commands: Refer to Appendix D for the directory locations.

1. Shut down the CIM Server.

   $(SBINDIR)`/cimserver -s`

2. Move the repository directory.

Platform	Command
Linux	`mv` *`$(REPOSITORYDIR) $(BACKUPREPOSITORYDIR)`*

HP-UX	`mv $(REPOSITORYDIR) $(BACKUPREPOSITORYDIR)`

3. Start the CIM Server.
   `$(SBINDIR)/cimserver`

4. Run the `init_repository` script.
   `$(SBINDIR)/init_repository`

5.5 CIM Server Properties

After OpenPegasus is installed, you can configure these properties, using the `cimconfig` command. You must have privileged user (root) capabilities to modify properties.

It is good practice to regularly backup the two property configuration files. Refer to Appendix D for the value of *$(CONFIGDIR).*

- `$(CONFIGDIR)/cimserver_current.conf` contains the current values that are not defaulted.
- `$(CONFIGDIR)/cimserver_planned.conf` contains planned values that are not defaulted. Changes to the planned configuration file will not take effect until the CIM Server is restarted.

— Note: Do not edit configuration files directly. Use the `cimconfig` command to change the property values in the files.

At startup, you can temporarily modify property values, by entering a `propertyname=value` pair on the `cimserver` command line. However, these modifications last only as long as that execution of the CIM Server.

At shutdown, you can temporarily modify just one property value, `shutdownTimeout`, by entering a value on the `cimserver` shutdown command line.

Only certain configuration properties can be changed dynamically. Please refer to the `cimconfig` reference page for details. For properties that are not dynamic, you must use the `-p` parameter to indicate your change, and then you must stop and restart the CIM Server. The `-p` parameter is explained in the `cimconfig` command summary below.

authorizedUserGroups

Set to user group names, group names are separated by a comma. The default is not set to any user group, which means that all users on the system are authorized (if not restricted by setting `enableNamespaceAuthorization`) to access CIM resources.

You can use user group authorization if you need the extra security of restricting access to CIM resources.

A privileged user (user with root permissions on the local system) is always authorized. A privileged user can grant user group authorizations to other users. For more information, see Chapter 6.

enableHttpConnection

Set to true or false. The default is false, which means that OpenPegasus will not listen on port 5988. Setting it to true allows user access through port 5988, using HTTP TCP/IP communication. Use HTTP connections only if you are certain your environment is secure. For more information, see Chapter 6.

enableHttpsConnection

Set to true or false. The default, true, allows user access through port 5989, using HTTPS TCP/IP communication. HTTPS connection has better security than HTTP. For more information, see Chapter 6.

enableNamespaceAuthorization

Set to true or false. The default, false, means that users are authorized across all namespaces.

If `enableNamespaceAuthorization` is set to true, you must authorize each user, namespace by namespace, with the `cimauth` command.

You can use namespace authorization if you need the extra security of restricting access to certain namespaces. Users with root permission on the local system are always privileged users. A privileged user can grant namespace authorizations to others. For more information, see Chapter 6.

enableRemotePrivilegedUserAccess

Set to true or false. The default is true. True means that an authenticated user, with privileged access to the system running the OpenPegasus CIM Server, will be allowed to issue requests to OpenPegasus from a remote system.

shutdownTimeout

Set to a number of seconds. When a `cimserver -s` shutdown command is issued, the timeout is the maximum number of seconds allowed for the CIM Server to complete outstanding CIM Operation requests before shutting down. If the specified timeout period expires, the CIM Server will shut down, even if there are still CIM Operations in progress.

Minimum value is 2 seconds. Default value is 10 seconds.

enableSubscriptionsForNonprivilegedUsers

Set to true or false. When set to the default value, false, only a privileged user (superuser) is allowed to perform operations (create instance, modify instance, delete instance, get instance,

enumerate instances, enumerate instance names) on indication filter, listener destination, and subscription instances. By default, the `enableSubscriptionsForNonprivilegedUsers` configuration property is set to false.

— Note: Restriction of operations on indication filter, listener destination, and subscription instances to only privileged users decreases the risk of a malicious user creating subscriptions that flood a system with indications. Restriction of operations, as well as access restrictions on the repository files, decreases the risk of subscription data (such as email addresses) being read, modified, or deleted by an unauthorized user.

5.6 The cimconfig Command

The `cimconfig` command manages CIM Server configuration properties. The operations are executed on the CIM Server running on the local host.

Use the `cimconfig` command to get, set, or unset CIM Server property values. Use the `-l` (list) option to see all properties and their values.

An operation on a current property (`cimconfig` with a `-c` option) takes effect immediately.

An operation on a planned property (`cimconfig` with a `-p` option) takes affect the next time the CIM Server is started with the `cimserver` command.

Dynamic properties can be set with either current or planned. Non-dynamic properties must be set using the planned option.

Modifications made by `cimconfig` remain in effect until they are changed by another `cimconfig` command. The CIM Server must be running to change the current configuration options.

You can temporarily modify property values when OpenPegasus is down, by entering options at startup in the `cimserver` command line. However, these modifications last only as long as that execution of the CIM Server.

6 Security Considerations

This chapter describes OpenPegasus security.

Security is checked first at the communication path. OpenPegasus has three pathways:

- Local users with requests

  If the user is on the same system as OpenPegasus, OpenPegasus accepts the authentication already done by the system itself. See Local Authentication, below.

- Remote users with requests

  If the user is coming from a remote system, he enters through the OpenPegasus HTTP Server. The embedded HTTP Server receives only valid CIM requests; all other requests are rejected. User information is included in the HTTP message header. The CIM Server checks the user-password information. See Remote Authentication, below.

- Providers

  OpenPegasus interacts with its registered providers through shared libraries.

— Note: CIM providers run as privileged users. Be very careful installing a provider that does not come from a trusted source.

After OpenPegasus passes on a request to a provider, the provider is responsible for checking its own security. The provider sets the rules about which requests it considers, and the conditions for granting or refusing them. If a provider requires authorization beyond that checked by OpenPegasus, the provider supplier is responsible for documenting its own rules.

OpenPegasus uses dedicated ports for CIM-XML traffic. Two ports are specified by DMTF and registered with IANA for CIM-XML communications between remote clients and the CIM Server:

- HTTP TCP/IP communication on port 5988 (wbem_http)
- HTTPS TCP/IP communication on port 5989 (wbem_https)

6.1 User Authentication

When a user request comes through HTTP (HyperText Transport Protocol) or HTTPS (HTTP Secure), the CIM Server determines whether this is a legitimate user on the system. If the request does not pass authentication, the request is rejected without processing.

Local users are users on a system sending requests to OpenPegasus on the same system.

Remote users are users on a system sending requests to OpenPegasus on another system.

6.1.1 Local User Authentication

For local users, the CIM Server uses a local authentication mechanism. The CIM Server uses the existing file system security to authenticate the user. OpenPegasus accepts the authentication already done by the system itself, so local requests include only the users' login names, not their passwords.

6.1.2 Remote User Authentication

Remote users accessing the CIM Server are authenticated with a request/challenge mechanism using HTTP Basic authentication.

A request is received from a management client. The CIM Server challenges the client to send a Base64-encoded username and password in the HTTP Authorization header.

To verify that the encoded user-password pair is authorized on the system, OpenPegasus calls Pluggable Authentication Module (PAM). For information about PAM, see the PAM reference page.

The default value for the configuration parameter `enableRemotePrivilegedAccess` is true. This means that, by default, an authenticated user – with privileged access to the system running WBEM Services – will be allowed to issue requests to OpenPegasus from a remote system.

When OpenPegasus installs, the CIM Server will be configured with a randomly generated, self-signed certificate. If a self-signed server certificate does not give a sufficient level of trust, the system administrator can use a central Certificate Authority (e.g., Verisign) to issue certificates.

When OpenPegasus for Linux is installed, a "wbem" file is installed in the `/etc/pam.d` directory. This file will list the PAM security modules used for OpenPegasus. The System Administrator may change the contents to conform to the security requirements of the managed system.

6.2 HTTPS and HTTP

By default, `enableHttpsConnection` is set to `true`, and OpenPegasus listens on port 5989. You can set the HTTPS connection to false, and set the property `enableHttpConnection` to true to make OpenPegasus listen on port 5988.

Use the `cimconfig` command to reset the property file. To change properties temporarily, for just one session, start CIM Server with the `cimserver` command and use the command line properties option.

If you set both HTTPS and HTTP to true (enabled), OpenPegasus will listen on both ports 5988 and 5989.

If you set both to false (disabled) OpenPegasus will listen only on the domain socket and accept requests from local clients; i.e., connections established using the `connectLocal` method in the `CIMClient` interface.

By default, OpenPegasus uses Secured Socket Layer (SSL) for all communications, with server-side certificates that are trusted by the management application. This gives both spoof protection and confidentiality.

— Note: Basic Authentication requires the client to pass both the username and password, both in Base64 encoding. This encoding is not secure. SSL (`enableHttpsConnection`) should only be disabled in a highly secure environment, where passing clear text passwords is not an issue.

OpenPegasus uses OpenSSL to support HTTPS connections. OpenSSL is a cryptography toolkit that implements the network protocols and related cryptography standards of SSL Version 2/3 and Transport Layer Security (TLS).

On the HTTPS port, CIM Clients are required to use SSL to establish connections with the CIM Server and to send or receive CIM requests.

6.3 User Group Authorization

User group authorization consists of establishing that the already authenticated user is a member of one of the configured groups in the `authorizedUserGroups` configuration property. If the user is not authorized, the client request is rejected without processing and an authorization failure message is sent back.

A user with root permission (uid 0) on the local system can use the `cimconfig` command to set the OpenPegasus `authorizedUserGroups` property to one or more user groups on the local system.

— Note: Unless explicitly rejected by a Provider, a user with root permission (uid 0) on the local system always has authorization to access CIM resources.

When the `authorizedUserGroup` property is set to valid group name(s) on the system and a user who is not a member of the configured group(s) submits a request, the following user error is displayed: `"User <username> is not authorized to access CIM data"`.

For more information about setting authorized user groups, see the reference page for the `cimconfig` command.

6.4 Namespace Authorization

CIM Services gives authenticated users controlled access to the entire CIM Schema. It does not check security for specific resources, such as individual classes and instances.

However, you can choose to control each user's access by requiring authorization for each user on each namespace. A user with root permission (uid 0) on the local system can use the `cimconfig` command to set the OpenPegasus `enableNamespaceAuthorization` property to true, and then use the `cimauth` command to set each user's access authorization on each namespace.

— Note: A user with root permission on the local system (uid 0) always has all permissions on all namespaces.

When namespace authorization is set to true, and a user submits a request for a namespace on which he is not authorized, this user error is displayed: “`Not authorized to run <requesting operation> in the namespace <requesting namespace>`”.

For more information about authorization, see the reference pages for the `cimauth` and `cimconfig` commands.

Authorizations are: Read, Write, or Read and Write. (Note that Write does not automatically include Read.)

The following CIM Operations require Write authorization:

```
CreateClass
CreateInstance
DeleteClass
DeleteInstance
DeleteQualifer
InvokeMethod
ModifyClass
ModifyInstance
SetProperty
SetQualifier
```

The following CIM Operations require Read authorization:

```
EnumerateClasses
EnumerateClassNames
EnumerateInstances
EnnumerateInstanceNames
EnumerateQualifiers
GetClass|
GetInstance|
GetProperty|
GetQualifier
```

A summary of the operations is given in Appendix B.

7 OpenPegasus Troubleshooting

This chapter is for people who are having trouble while trying to use OpenPegasus.

The OpenPegasus messages are listed here.

7.1 Checklist for Troubleshooting OpenPegasus

If you are having trouble with OpenPegasus, try this checklist:

- Is the CIM Server is running? You can use the following platform-specific command to determine whether the CIM Server is running.

Platform	Command
Linux	`ps -ef|grep cimserver`
HP-UX	`ps -ef|grep cimserver`

  If it isn't running, then you must start it using the `cimserver` command.

- Is OpenPegasus installed correctly? Do you have the essential files? Check to see if the repository directory and configuration files exist. If any of these files are missing, restore all the repository directories and files from your backup.

  If you cannot restore the repository directories, you will have to re-initialize the repository. This will return it to the state it was in when you installed OpenPegasus, and you will lose any changes made since then. See Section 5.4.

- Are you trying to process a request when the provider is not registered or enabled? Enter `cimprovider -l -s` to list the name and status of the registered provider modules and `cimprovider -l-m <modulename>` to see the individual providers in that module.

- Exercise the path that requests follow: enter `osinfo`. This invokes a simple request. It should process and display a response to show you it completed.

- Check the system log files. OpenPegasus messages are listed below.

- Are you seeing SSL certificate-related messages on a CIM Client request failure? Make sure the remote CIM Server certificate *`$(CERTIFICATEDIR)/`*`cert.pem` is added to the trust store file on the client system. For information on how to add certificates to the trust store file, refer to the `OpenSSL` documentation (see www.openssl.org/docs).

7.2 OpenPegasus Messages

The OpenPegasus messages are listed in four groups: syslog messages, standard CIM messages, command messages, and SSL errors.

7.2.1 General Syslog Messages

OpenPegasus puts the following messages in syslog:

- If the `logLevel` is set to INFORMATION, the CIM Server logs a message on startup; for example:

  ```
  Jan 19 21:49:24 zambezi cimserver[13661]: Listening on HTTPS port
  5989.
  Jan 19 21:49:24 zambezi cimserver[13661]: Listening on local
  connection socket.
  Jan 19 21:49:25 zambezi cimserver[13661]: Started CIM Server
  version 2.4.
  ```

- If the `logLevel` is set to INFORMATION, the CIM Server logs a message on shutdown; for example:

  ```
  Jan 19 21:50:28 zambezi cimserver[13661]: CIM Server stopped.
  ```

7.2.2 Indication Service Syslog Messages

- Message: `"One or more invalid Subscription instances were ignored"`

  This message may be logged upon CIM Server startup (`IndicationService` initialization), if an invalid `Subscription` instance is found in the repository.

  Invalid instances could exist if instances in the repository have been directly created or modified by circumventing the `IndicationService`.

  In such cases, the `IndicationService` detects such corruption, ignores invalid subscription instances, and continues to make a best effort at processing requests on valid subscription instances. Invalid instances should be removed from the repository, and care should be taken that invalid instances are not introduced into the repository by circumventing the `IndicationService`.

- Message: `"Subscription ($0) has no provider"`

  This message may be logged upon CIM Server startup (`IndicationService` initialization), if there is currently no provider that can serve an existing enabled subscription.

  The substitution data `$0` identifies the subscription, and contains the values of the subscription `Filter` and `Handler Name` properties in the form `"FilterName, HandlerName"`.

  This message may indicate that one or more indication providers has been removed or disabled, and it may be necessary to re-install, re-register, and/or re-enable one or more indication providers to avoid missing indications.

- Message: `"Provider ($0) is now serving subscription ($1)"`

  This message may be logged upon a provider registration change (creation or modification of a `PG_ProviderCapabilities` instance), or when a provider has been enabled (the `"cimprovider -e"` command). The substitution data `$0` identifies the provider, and contains the value of the `Provider Name` property. The substitution data `$1` identifies the subscription, and contains the values of the subscription `Filter` and `Handler Name`

properties in the form `"FilterName, HandlerName"`. This message indicates that additional indications may now be generated by the specified provider for the specified subscription.

- Message: `"Provider ($0) is no longer serving subscription ($1)"`

  This message may be logged upon a provider registration change (deletion or modification of a `PG_ProviderCapabilities` instance), or when a provider has been disabled (the `"cimprovider -d"` command). The substitution data `$0` identifies the provider, and contains the value of the `Provider Name` property.

  The substitution data `$1` identifies the subscription, and contains the values of the subscription `Filter` and `Handler Name` properties in the form `"FilterName, HandlerName"`. This message indicates that no indications will be generated by the specified provider for the specified subscription.

  Other indication providers may still be serving the specified subscription. It may be necessary to re-install, re-register, and/or re-enable one or more indication providers to avoid missing indications.

7.2.3 Standard CIM Messages

- 0 = CIM_ERR_SUCCESS

  The operation completed without error.

- 1 = CIM_ERR_FAILED

  A general error occurred that is not covered by a more specific error code.

- 2 = CIM_ERR_ACCESS_DENIED

  Access to a CIM resource was not available to the client.

- 3 = CIM_ERR_INVALID_NAMESPACE

  The target namespace does not exist.

- 4 = CIM_ERR_INVALID_PARAMETER

  One or more parameter values passed to the method were invalid.

- 5 = CIM_ERR_INVALID_CLASS

  The specified class does not exist.

- 6 = CIM_ERR_NOT_FOUND

  The requested object could not be found.

- 7 = CIM_ERR_NOT_SUPPORTED

  The requested operation is not supported.

- 8 = CIM_ERR_CLASS_HAS_CHILDREN

  Operation cannot be carried out on this class because it has subclasses.

- 9 = CIM_ERR_CLASS_HAS_INSTANCES

Operation cannot be carried out on this class because it has instances.

- 10 = CIM_ERR_INVALID_SUPERCLASS

  Operation cannot be carried out because the specified superclass does not exist.

- 11 = CIM_ERR_ALREADY_EXISTS

  Operation cannot be carried out because an object already exists.

- 12 = CIM_ERR_NO_SUCH_PROPERTY

  The specified property does not exist:

- 13 = CIM_ERR_TYPE_MISMATCH

  The value supplied is not compatible with the type.

- 14 = CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED

  The query language is not recognized or supported.

- 15 = CIM_ERR_INVALID_QUERY

  The query is not valid for the specified query language.

- 16 = CIM_ERR_METHOD_NOT_AVAILABLE

  The extrinsic method could not be executed.

- 17 = CIM_ERR_METHOD_NOT_FOUND

  The specified extrinsic method does not exist.

The following list has the same messages as above; however, it is ordered alphabetically, and without the error number:

- CIM_ERR_ACCESS_DENIED

  Access to a CIM resource was not available to the client.

- CIM_ERR_ALREADY_EXISTS

  Operation cannot be carried out because an object already exists.

- CIM_ERR_CLASS_HAS_CHILDREN

  Operation cannot be carried out on this class because it has subclasses.

- CIM_ERR_CLASS_HAS_INSTANCES

  Operation cannot be carried out on this class because it has instances.

- CIM_ERR_FAILED

  A general error occurred that is not covered by a more specific error code.

- CIM_ERR_INVALID_CLASS

  The specified class does not exist.

- CIM_ERR_INVALID_NAMESPACE:

  The target namespace does not exist.

- CIM_ERR_INVALID_PARAMETER
  One or more parameter values passed to the method were invalid.
- CIM_ERR_METHOD_NOT_AVAILABLE
  The extrinsic method could not be executed.
- CIM_ERR_METHOD_NOT_FOUND
  The specified extrinsic method does not exist.
- CIM_ERR_INVALID_QUERY
  The query is not valid for the specified query language.
- CIM_ERR_INVALID_SUPERCLASS
  Operation cannot be carried out because the specified superclass does not exist.
- CIM_ERR_NO_SUCH_PROPERTY
  The specified property does not exist.
- CIM_ERR_TYPE_MISMATCH
  The value supplied is incompatible with the type.
- CIM_ERR_NOT_FOUND
  The requested object could not be found.
- CIM_ERR_NOT_SUPPORTED
  The requested operation is not supported.

Examples of CIM Responses

For example, consider a client requesting a `createInstance` operation on the `PG_OperatingSystem` class, when this operation is not supported by the Operating System Provider. The requestor will receive the following response (shown below encoded in XML):

```
<?xml version=”1.0” encoding=”utf-8”?>
<CIM CIMVERSION=”2.0” DTDVERSION=”2.0”>
<MESSAGE ID=”53000” PROTOCOLVERSION=”1.0”> <SIMPLERSP>
<IMETHODRESPONSE NAME=”CreateInstance”>
<ERROR CODE=”7” DESCRIPTION=”CIM_ERR_NOT_SUPPORTED:
The requested operation is not supported: “OperatingSystemProvider
does not support createInstance”/>
</IMETHODRESPONSE>
</SIMPLERSP>
</MESSAGE>
</CIM>
```

In the above example, you see the following four components of the response:

1. CIM error code of 7
2. Translation to `CIM_ERR_NOT_SUPPORTED`

3. Expanded text message: `The requested operation is not supported.`

4. The non-standard additional message: `OperatingSystem Provider does not support createInstance`.

As a second example, consider a client that mistakenly provides too few or too many keys to a `GetInstance` operation on the `PG_OperatingSystem` class. The following response is sent (shown below encoded in XML):

```
<?xml version=”1.0” encoding=”utf-8”?>
<CIM CIMVERSION=”2.0” DTDVERSION=”2.0”>
<MESSAGE ID=”35002” PROTOCOLVERSION=”1.0”> <SIMPLERSP>
<IMETHODRESPONSE NAME=”GetInstance”>
<ERROR CODE=”4”
DESCRIPTION=”CIM_ERR_INVALID_PARAMETER: One or more parameter values
passed to the method were invalid: “Wrong number of keys”/>
</IMETHODRESPONSE>
</SIMPLERSP>
</MESSAGE>
</CIM>
```

In the above example, you see the following four components of the response:

1. CIM error code of 4

2. Translation to `CIM_ERR_INVALID_PARAMETER`

3. Expanded text message: `One or more parameter values passed to the method were invalid`.

4. The non-standard additional message: `Wrong number of keys`.

7.2.4 OpenPegasus Command Messages

These messages come from the OpenPegasus commands. They are written to stdout.

cimauth Command Messages

- Message: `You must have superuser privilege to run this command.`

  If you do not have root permissions (uid=0) on the local system, get a logon that does, or have such a privileged user give you permission. (See Chapter 3 and the `cimauth` reference page.)

- Message: `Failed to add authorizations. Please make sure that the authorization schema is loaded on the CIMOM.`

  Essential information is missing from the repository. See Section 5.4.

- Message: `Failed to add authorizations. Specified user authorization already exists.`

  If you want the authorization added, you do not need to do anything; it is already there. To modify, use the `-m` option. To remove, use the `-r` option.

- Message: `Failed to modify authorizations. Specified user authorizations were not found.`

  Enter `cimauth -l` to list all the authorizations. See if the one you want to modify is in the list, and if you are spelling it correctly. If it's not in the list, you need to add it with the `-a` option. Then re-issue the command.

- Message: `Failed to remove authorizations. Specified user authorizations were not found.`

  Enter `cimauth -l` to list all the authorizations. See if the one you want to remove is in the list, and if you are spelling it correctly. If it's not in the list, you need to add it with the `-a` option. Then re-issue the command.

- Message: `CIM Server may not be running.`

  To see whether `cimserver` is running, enter `ps -ef|grep cimserver`. Perhaps an operator stopped it by command, but did not restart it.

  For Linux, enter `/sbin/service pegasus-wbem start`.

cimconfig Command Messages

- Message: `Current value of properties cannot be listed because the CIM Server is not running.`

  Check for `cimserver` using `ps -ef|grep` cimserver. Perhaps it was never started at install, or someone may have stopped it with `/sbin/service pegasus-wbem` stop for Linux. You must restart it.

  For Linux, enter `/sbin/service pegasus-wbem start`.

- Message: `Failed to get property. Please make sure that the config schema is loaded in the CIM Server.`

  Essential information is missing from the repository. See Section 5.4.

- Message: `Failed to set the config property. Please make sure that the config schema is loaded in the CIM Server.`

  Essential information is missing from the repository. See Section 5.4.

- Message: `Failed to unset the config property. Please make sure that the config schema is loaded in the CIM Server.`

  Essential information is missing from the repository. See Section 5.4.

- Message: `Failed to list the config properties. Please make sure that the config schema is loaded in the CIM Server.`

  Essential information is missing from the repository. See Section 5.4.

- Message: `Specified property name was not found.`

  Check the spelling of the property name. Re-issue the command specifying a valid config property. For a list of properties, enter `cimconfig –l`.

- Message: `Specified property value is not valid.`

  See the `cimconfig` reference page for the range of allowed values for the property, and re-issue the command with a valid value.

- Message: `Specified property cannot be modified.`

  You are trying to modify a property that is not dynamic. Dynamic properties can be changed immediately, while the CIM Server is running.

  To modify a non-dynamic property you must modify the planned value, then stop and start the CIM Server. For more information, see the `cimconfig` reference page.

- Message: `Current value cannot be determined because the CIM Server is not running.`

  To see whether `cimserver` is running, enter `ps -ef|grep cimserver`. Perhaps an operator stopped it by command, but did not restart it. To start it, do the following:

  For Linux, enter `/sbin/service pegasus-wbem start`.

- Message: `Planned value cannot be determined because the CIM Server is not running.`

  To see whether `cimserver` is running, enter `ps -ef|grep cimserver`. Perhaps an operator stopped it by command, but did not restart it. To start it, do the following:

  For Linux, enter `/sbin/service pegasus-wbem start`.

- Message: `CIM Server may not be running.`

  To see whether `cimserver` is running, enter `ps -ef|grep cimserver`. Perhaps an operator stopped it by command, but did not restart it. To start it, do the following:

  For Linux, enter `/sbin/service pegasus-wbem start`.

cimmof Command Messages

- Message: `Warning: class already in repository (OK to ignore).`

  The same class is already loaded, so you do not need to do it again. If you really want to replace this class, first delete it, then load your new MOF file.

- Message: `Cannot connect to: mysystem: 5989. Command failed.`

  The CIM Server is not running. You tried to send a request to system `mysystem`, through port 5989. An operator may have stopped the CIM Server. Restart it, then re-issue the `cimmof` command.

- Message: `Can’t open file <filename>.`

  Check the MOF file that you specified. It could not be opened; it may not exist, the pathname may be incomplete, or there may be a typing error. Re-issue the command specifying a valid MOF file.

- Message: `Could not open include file <filename>.`

  Check the MOF include file that you specified. It could not be opened; it may not exist, the pathname may be incomplete, or there may be a typing error. Re-issue the command specifying a valid MOF file.

- Message: `<filename>:< lineNumber>: parse error before ‘string’`

  There is a parsing error before `string`. If it is your own file, edit it to correct invalid syntax, and then re-issue the command. If you got the file from a provider, contact the provider’s support team.

- Message: `Error adding class <classname> to the repository: CIM_ERR_INVALID_SUPERCLASS: Operation cannot be carried out since the specified superclass does not exist.`

  The file you specified contains schema definition for a class with a superclass, but its superclass is not in the CIM Repository now. You must load the superclass before you load its subclasses.

  If it is your own MOF file, edit it to check the spelling of the class and superclass, and the path and spelling of the MOF file in your command. If you got the MOF file from a provider, contact the provider’s support team.

- Message: `Could not find declaration for Qualifier named <qualifier_name>.`

  OpenPegasus cannot find the qualifier name in the MOF file in the CIM Repository.

  If it is your own MOF file, check the qualifier name in the MOF file you specified. If it is misspelled, correct it. To see all qualifiers, for Linux go to `/var/cache/pegasus/repository/<namespace>/qualifiers`.

  If the qualifier does not exist in the CIM Repository, add it, and then re-issue the command.

  If you got the MOF file from a provider, contact the provider’s support team.

cimprovider Command Messages

- Message: `Required arguments missing.`

  Change the syntax of your command; perhaps check spelling. `cimprovider` does not recognize the options you entered. Enter `cimprovider`, with no options, to see correct usage. Also see the `cimprovider` reference page.

- Message: `Missing required value for flag.`

  Check your syntax for a flag that is missing its value. Enter `cimprovider`, with no options, to see correct usage. Also see the `cimprovider` reference page.

- Message: `The CIM Server may not be running.`

  To see whether `cimserver` is running, enter `ps -ef|grep cimserver`.

  Perhaps an operator stopped it by command, but did not restart it. To start it, do the following:

For Linux, enter `/sbin/service pegasus-wbem start`.

- Message: `Provider module already disabled.`

  You cannot disable a provider that is already disabled. Use `cimprovider -l -m <modulename>` to see status of all the providers in the specified module.

- Message: `You must have superuser privilege to unregister providers.`

  If you do not have root permissions (uid=0) on the local system, get a logon that does, or have such a privileged user give you permission. (See Chapter 3 and the `cimauth` reference page.)

- Message: `You must have superuser privilege to disable or enable providers.`

  If you do not have root permissions (uid=0) on the local system, get a logon that does, or have such a privileged user give you permission. (See Chapter 3 and the `cimauth` reference page)

- Message: `Provider module cannot be enabled since it is disabling.`

  You cannot enable a provider while another client is disabling the module. Enable it later.

- Message: `Specified provider was not registered.`

  You are trying to manage an unregistered provider. (To confirm, use the `cimprovider -l` command.) Register the provider.

cimserver Command Messages

- Message: `Error: Bind failed. Failed to bind to socket.`

  You tried to start the CIM Server, but it is already running.

- Message: `Unrecognized command line option.`

  Re-issue the command specifying a valid option. For help with options, type `cimserver -h` or see the reference page.

- Message: `Duplicate shutdown option specified.`

  The `-s` option was specified more than once. Re-issue the command with a valid option. For help with options, enter `cimserver -h` or see the reference page.

- Message: `Unrecognized config property: <configProperty>`

  Check the spelling of the property. Re-issue the command specifying a valid config property. For a list of properties, enter `cimconfig -l`.

- Message: `Invalid property value: shutdownTimeout=<value>`

  Specify a `shutdownTimeout` value that is a valid integer, 2 or greater.

- Message: `Unable to connect to CIM Server. CIM Server may not be running.`

  To see whether `cimserver` is running, enter `ps -ef|grep cimserver`.

Perhaps an operator stopped it by command, but did not restart it. To start it, do the following:

For Linux, enter `/sbin/service pegasus-wbem start`.

Perhaps someone has disabled both types of connection (HTTPS and HTTP). To start it in that case, enter either:

`cimserver enableHttpsConnection=true` (default)

`cimserver enableHttpConnection=true`

- Message: `Failed to shut down server: CIM_ERR_INVALID_NAMESPACE: The target namespace does not exist "root/PG_Internal"`

  The `cimserver` command cannot stop the CIM Server. The only way to stop the CIM Server is to kill the CIM Server process:

  — Find the process ID (PID) of `cimserver`. Enter `ps -ef|grep cimserver`.

  — Kill the process: `kill -9 <PID>`

  The most likely cause is that the CIM Repository was moved or deleted, or that it is empty or corrupted. Try replacing all the directories and files located in `/var/cache/pegasus/repository` with your backup copy.

  If you cannot replace the repository directories, you can use the `init_repository` script to restore your repository to what it was when you first installed OpenPegasus. You will need to re-install any providers you added since you installed OpenPegasus. (You do not need to re-install the providers that are bundled with OpenPegasus.)

openssl and SSL-Related Messages

- Server-side SSL (Secure Socket Layer) Errors

  1. `Could not get server certificate.`

  The present server certificate file is missing, empty, or not readable. Restore the certificate file (`/etc/opt/hp/sslshare/cert.pem`) from backup, and then start the CIM Server.

  2. `Could not get private key.`

  The present key file is missing, empty, or not readable. Restore the key file (`/etc/opt/hp/sslshare/file.pem`) from backup, and then start the CIM Server.

osinfo Command Messages

- Message: `Cannot get info from OS provider.`

  Verify that the provider in your request is listed. Enter `cimprovider -l -m OperatingSystemModule`.

- Message: `Cannot connect to CIM Server.`

  To see whether the `cimserver` process is running, enter `ps -ef|grep cimserver`.

  To start the process, do the following:

  For Linux, enter `/sbin/service pegasus-wbem start`.

wbemexec Command Messages

- Message: `Invalid input: expected XML request.`

  Check the coding of the request. The input must be a valid CIM request encoded in XML according to the DMTF CIM-XML specification.

- Message: `Invalid XML request.`

  Correct the XML request, and re-issue the command. Refer to the text following the message for more specific information about the invalid XML request. For more information about XML, see the DMTF CIM-XML specification.

- Message: `Timed out waiting for response.`

  You can change the timeout value with a `wbemexec` command option. The request may require more processing time than allowed by the specified or default timeout period. Specify a timeout value longer than the value previously specified or longer than the default.

  Check `syslog` for possible errors or problems with the CIM Server or providers. An error may have occurred in the CIM Server, preventing the CIM Server from responding to requests. (A list of `syslog` messages is in this chapter.)

  If necessary, stop and re-start the CIM Server. Re-issue the `wbemexec` command.

- Message: `wbemexec: Failed to connect to CIM Server.`

  First, read the text that follows this message, for more information about the problem.

  Enter `ps -ef|grep cimserver`. If the `cimserver` process is not running, you must start it. Do the following to start it:

  For Linux, enter `/sbin/service pegasus-wbem start`.

  After this, the log file should record an attempt to start `cimserver` and a confirmation that `cimserver` started.

  On the CIM Server host, enter `uname -a` to be sure you have specified the appropriate host name.

  Enter `cimconfig -l -c` to list current values of properties. See whether the enabled connection is port HTTP or HTTPS. Now see if your request specified the corresponding port. By default, HTTPS (default type) enters port 5989; HTTP enters port 5988.

  You may not be authorized to connect to the CIM Server. See Chapter 3.

- Message: `wbemexec: M-POST method invalid with HTTP Version 1.0.`

  Modify the command line. Either specify HTTP Version 1.1 with the M-POST or POST method, or specify HTTP Version 1.0 with the POST method.

  The `M_POST` method is only valid for HTTP versions 1.1 and later.

- Message: `wbemexec: No input.`

  Be sure that you did not specify an empty file, or redirect input from an empty file.

- Message: `wbemexec: Unable to use requested input file: file cannot be opened.`

Check to be sure there is sufficient memory to open a file, and that you have not reached the open-file limit. `wbemexec` can find the file, and the permissions allow the file to be read, but the file cannot be opened for some other reason.

- Message: `wbemexec: Unable to use requested input file: file does not exist.`

  Check the pathname and spelling of the input file you specified.

- Message: `wbemexec: Unable to use requested input file: file is not readable.`

  Check the permission settings on the specified input file and its directories, modify if necessary, and re-issue the command.

A How Resources are Represented (CIM Schema)

The OpenPegasus repository stores information about managed resources.

To register with OpenPegasus, a provider must define its resource by the classes and subclasses that define it. Then the provider must describe the properties that it will expose, and the methods that it will support.

The properties describe what a class *is*, the methods describe what it can *do*. Properties are attributes or characteristics of the resource. Methods are its actions, capabilities, or behaviors.

To make a request, the client must first identify, by its classes and subclasses, the resource it wants to manage.

The resource descriptions are done using object-oriented modeling. Object-oriented modeling represents real things in an abstract schema. Objects are arranged from most general to most specific. Many attributes of the more general parent are inherited by their more specific children.

Like object-oriented programming languages, the subclasses inherit the definitions of properties and methods from the parent class. Unlike some object-oriented programming, they do not inherit the implementations.

This section briefly defines basic concepts about object representation. As system administrator, you do not need to understand this to install OpenPegasus or maintain it. However, it is the language that is used to explain resources. These are the terms that are used to describe what providers and clients do, and how resources can be managed.

For more information about object representation, refer to the DMTF CIM Tutorial.

The schema is the most general abstraction that represents real things in the WBEM standard. A schema is a collection of classes. Each class in a schema can only belong to that schema. Each class name must be unique within a schema; a schema cannot have two classes with the same name.

The class is the basic modeling unit. It is a collection or set of objects that have similar properties and purposes. Each class defines a certain type of managed object; for example, operating systems or system memory. Objects in the class contain properties (describing what it is) and methods (what it can do). A class can contain other classes (its subclasses). It can also contain instances.

Subclasses are grouped by similarities. Subclasses inherit properties and methods from their parent (their superclass), and can also add their own local properties and methods. Subclasses are themselves classes, and they can have their own subclasses.

`CIM_SoftwareElement`, for example, is a class. It has several subclasses, such as `Linux_SoftwareElement`, `Win32_SoftwareElement`, `HPUX_RPM_SoftwareElement`, and `Linux_Debian_SoftwareElement`.

An instance can be a discrete occurrence of any object, such as your computer's hard drive or the printer on your desk. It is the most specific member of the hierarchy. An instance cannot have any subclasses. All instances in a class share the same properties and methods. Each has a unique name (see key properties, below).

Methods are the behaviors of the class; for example, the `OperatingSystem` class has a `Reboot` method and a printer has an `EnableDevice` method to put it online. Not all classes have methods.

An intrinsic method models a CIM Operation. Standard intrinsic methods (such as `enumerateInstances`, `getInstance`, `modifyInstance`) are relevant to all classes.

An extrinsic method is defined on a CIM Class in some Schema that is unique to that class.

Properties are the attributes of a class; for example, there is a `ParticipatingCS` association between a `CIM_ComputerSystem` and a `CIM_Cluster`. This association has two properties, `RoleOfNode` and `StateOfNode`, to describe attributes of the `ComputerSystem` as a node within the Cluster.

Key properties (one or more properties defined with a "key" qualifier) are identifiers. Keys in classes and subclasses provide a way to uniquely identify the instance that inherits them. All instances inherit a key, or a set of keys, from their superclass. The value that the instance gives to these keys is its own identification. It is the only instance in its namespace that is allowed to have that "name". More than one key property is a compound key.

Consider how to uniquely identify a user account on a UNIX system. You could use two key properties: the value of the user account's `Name` property and the value of the system's `Name` property. Consider also the identifying pair used to route your email to you: `username@domain-name`.

Classes are either concrete or abstract. A *concrete* class (like `CIM_OperatingSystem`) has real instances, particular computer systems. A concrete class must have at least one key property. An *abstract* class like `CIM_ManagedElement` cannot have any instances, and it is not required to have key properties. Its subclasses can have keys as they get more specific.

Associations can be defined between classes. For example, there is a `ParticipatingCS` association between `CIM_ComputerSystem` (the entire computer system) and `CIM_OperatingSystem` (the OS software that exists on that system).

The association itself is a class, so it can have properties and methods. For example, two properties of `ParticipatingCS` are `RoleOfNode` and `StateOfNode`.

Namespaces can give you a logical way to group things, in order to control their scope and visibility. A namespace is not a physical location; it is more like a logical database containing specific classes and instances. Namespace grouping can be used to separate instances and make

sure there are no collisions with others of the same name. Namespaces also can be used to limit access.

OpenPegasus installs with four pre-defined namespaces:

- `root` (in `/root` directory)

  The root namespace exists to conform to the DMTF specifications.

- `root#cimv2` (in `/root/cimv2`)

  The standard CIM Schemas go here, as do the schemas for the bundled providers.

- `root#PG_Interop` (in `/root/PG_Interop`)

  This is for provider registration. This space is reserved exclusively for providers, and all providers must register here. (See the `cimprovider` reference page.)

- `root#PG_Internal` (in `/root/PG_Internal`)

  This is a private space, for use by OpenPegasus only.

B OpenPegasus CIM Operations

OpenPegasus supports a subset of the DMTF-defined CIM Operations. If you are installing a client or provider, be sure these are sufficient operations.

B.1 The InvokeMethod Operation

The following operation is a way of invoking the class of methods called extrinsic methods. (This is the way OpenPegasus supports extrinsic methods.) If a provider has registered with OpenPegasus as a method provider, it will support the use of `InvokeMethod`.

- `InvokeMethod` (Write)

  Takes a method name with input and output parameters, and an instance. The instance is specified by its namespace, classname, and key properties and values. Invokes the specified method on the specified instance.

B.2 Operations Implemented by Providers

The following CIM Operations are implemented by instance providers for the classes they support. The methods are intrinsic. If the provider does not support a particular method, the implementation returns `CIM_ERR_NOT_SUPPORTED`.

- `GetInstance` (Read)

  Takes a namespace, classname, and key properties and values. Returns the instance with all its properties.

- `EnumerateInstances` (Read)

  Takes a namespace and a classname. Returns all instances of the specified class, including all properties. When invoked on a class with subclasses, OpenPegasus will pass the `EnumerateInstance` CIM Operation to providers for all of the subclasses, and combine all the results into a single response.

- `EnumerateInstanceNames` (Read)

  Takes a namespace and a classname. Returns all instances of the specified class. It returns all key properties, but it does not return non-key properties. When invoked on a class with subclasses, OpenPegasus will pass the `EnumerateInstanceNames` CIM Operation to providers for all of the subclasses, and combine all the results into a single response.

- `CreateInstance` (Write)

  Takes a namespace, classname, and key properties and values. Can accept other properties and values. Creates an instance that meets those criteria.

- `DeleteInstance` (Write)

  Takes a namespace, classname, and key properties and values. Can accept other properties and values. Deletes the instance that meets those criteria.

- `ModifyInstance` (Write)

  Takes a namespace, classname, and key properties and values. Can accept other properties and values. Modifies the instance that meets those criteria.

B.3 Operations on Properties

Operations on properties are listed below.

- `GetProperty` (Read)

  Takes a namespace, classname, and key properties and values to specify an instance. Also takes the property desired. Returns the value of the property for the specified instance.

- `SetProperty` (Write)

  Takes a namespace, classname, and key properties and values, to specify a class. Also takes the desired property and value. Sets the desired property of that instance to the specified value.

B.4 Class Manipulation Operations

The class manipulation operations can be used by CIM Clients to explicitly manipulate schema. Schema manipulation can be done implicitly through a MOF file. When the MOF compiler loads a MOF file, the compiler will use a series of `CreateClass` operations to create the classes contained in the file.

Class manipulation operations are listed below:

- `GetClass` (Read)

  Takes a namespace and classname. Returns the class definition with all properties and methods.

- `EnumerateClasses` (Read)

  Takes a namespace and, optionally, a classname. Returns a list of all the classes and subclasses of that namespace (and classname if you specified it), including the definitions of all properties and methods.

- `EmumerateClassNames` (Read)

  Takes a namespace and classname. Returns a list of all subclasses of that namespace and class, including definitions of all key properties. Does not return non-key properties or methods.

- `CreateClass` (Write)

  Takes a namespace and class definition. Creates the specified class.

- `ModifyClass` (Write)

  Takes a namespace and a new class specification. Replaces the existing class specification to the new (modified) one.

- `DeleteClass` (Write)

  Takes a namespace and classname. Removes the class from the namespace. If the class has subclasses, you must remove the subclasses first.

B.5 Qualifier Operations

Qualifier declaration operations are listed below:

- `GetQualifier` (Read)

  Takes a namespace and a qualifier name. Returns the information on that qualifier, such as scope, flavor, and default value. (A qualifier is a modifier containing information that describes a class, an instance, a property, or a method.)

- `EnumerateQualifiers` (Read)

  Takes a namespace. Returns all qualifiers defined in the specified namespace. (A qualifier is a modifier containing information that describes a class, an instance, a property, or a method.)

- `SetQualifier` (Write)

  Takes a namespace and qualifier name. Also takes a qualifier declaration. Replaces the existing qualifier declaration with the specified declaration. (A qualifier is a modifier containing information that describes a class, an instance, a property, or a method.)

- `DeleteQualifier` (Write)

  Takes a namespace and a qualifier name. Deletes the specified qualifier from the specified namespace.

C OpenPegasus Configuration Operations Security Disclaimer

As a security best practice, it is recommended that any network daemons that are not in use should be disabled. Any daemons that are in use should be configured securely according to the threat environment in which they are located. This is a functionality *versus* security risk trade-off. The optimal configuration will vary depending on local threats and functionality requirements.

C.1 Default Security

For ease-of-manageability, OpenPegasus defaults to "functional" out-of-the-box, but provides several configuration options so that security risks may be minimized.

- The OpenPegasus CIM Server can be configured to only accept connections from local UNIX domain sockets. This is appropriate if you have untrusted users on your network and you do not plan to use OpenPegasus for remote management.

- OpenPegasus can be configured to only allow access from a trusted subset of system users (e.g., "root") and application users (e.g., "oracle") using a UNIX group. Setting up this user group is recommended if you intend to use WBEM in an environment where local users are untrusted, or as a second line of defense against break-ins.

— Note: If an application fails to authenticate after creating this group, you may need to add its application or associated system users.

- OpenPegasus supports the use of other protective measures for high-threat environments. For example, IPSEC or hardware solutions may be used to create a VPN to increase security. A VPN is recommended if you intend to use WBEM for management across an untrusted network, such as an exposed DMZ or the public Internet.

D Directory and File Locations

This Appendix describes the default locations for key OpenPegasus files and directories. However, these locations are configurable; consult your product-specific documentation for exact locations.

Platform		
Linux	$LOGDIR	/var/opt/tog-pegasus/log
	$PROVIDERDIRS	/opt/tog-pegasus/providers/lib
	$REPOSITORYDIR	/var/opt/tog-pegasus/repository
	$CERTIFICATEDIR	/etc/opt/tog-pegasus
	$LOCALAUTHDIR	/var/opt/tog-pegasus/cache/localauth
	$TRACEDIR	/var/opt/tog-pegasus/cache
	$CONFIGDIR	/var/opt/tog-pegasus
	$PIDFILE	/var/run/cimserver.pid
	$RANDOMDIR	/etc/opt/tog-pegasus
	$SOCKETDIR	/var/run/tog-pegasus/socket
	$SBINDIR	/opt/tog-pegasus/sbin
	$BINDIR	/opt/tog-pegasus/bin
	$SAMPLEDIR	/opt/tog-pegasus/samples
HP-UX	$LOGDIR	/var/opt/wbem
	$PROVIDERDIRS	/opt/wbem/providers/lib
	$REPOSITORYDIR	/var/opt/wbem/repository
	$CERTIFICATEDIR	/etc/opt/hp/sslshare/
	$LOCALAUTHDIR	/var/opt/wbem
	$TRACEDIR	/var/opt/wbem
	$CONFIGDIR	/var/opt/wbem/
	$PIDFILE	/etc/opt/wbem/cimserver_start.conf
	$RANDOMDIR	/var/opt/wbem
	$SOCKETDIR	/var/opt/wbem/socket
	$SBINDIR	/opt/wbem/sbin
	$BINDIR	/opt/wbem/bin
	$SAMPLEDIR	/opt/wbem.samples

Glossary

Much of this information was gathered from the DMTF CIM Tutorial, and much more information is available there.

CIM (Common Information Model)
: A hierarchical object-based model developed by the DMTF that defines a large number of concepts common to most computer systems. See Common Information Model. The CIM Specification can be found at www.dmtf.org.

CIM Client
: A client application that issues CIM Operation requests to a CIM Server over HTTP and processes the responses.

CIM Object Manager (CIMOM)
: Manages CIM objects in an OpenPegasus-enabled system. CIMOM receives and processes CIM Operation requests and issues responses.

CIM Object Manager Repository
: Manages CIM objects in an OpenPegasus-enabled system. CIMOM receives and processes CIM Operation requests and issues responses.

CIM Operations
: A set of operations, specified by the DMTF, that can be requested of a CIM Server to be performed on CIM objects. The specification can be found on the DMTF web site.

CIM Schema
: A collection of class definitions used to represent managed objects that exist in every management environment. See also Core Model, Common Model, and Extension Schema.

CIM Server
: A CIM Server is a server that processes requests for operations on CIM objects. It can have an HTTP Server component that communicates with clients using the HTTP protocol, transferring messages encoded in xmlCIM. It can also have a CIM Server that distributes requests to providers for translation to platform-specific operations.

Cipher
: A key-selected transformation between plain text and cipher text. With a good cipher, the secret information inside the cipher remains hidden, even when the cipher text is stored or transmitted.

Class
: A collection of instances, all of which support a common type; that is, a set of properties and methods. The common properties and methods are defined as features of the class. For example, the class called `Modem` represents all the modems present in a system.

Common Information Model (CIM)
: A common data model of an implementation-neutral schema for describing overall management information in a network/enterprise environment.

  CIM is comprised of a Specification and a Schema. The Specification defines the details for integration with other management models defined by the DMTF, such as SNMP's MIBs or the DMI's MIFs. The Schema provides the actual model descriptions.

Common Information Model Object Manager (CIM Object Manager)
: A component in the CIM management infrastructure that handles the interaction between management applications and clients.

Common Model
: The second layer of the CIM Schema, which includes a series of domain-specific but platform-independent classes. The domains are systems, networks, applications, and other management-related data. The common model is derived from the core model. See also Extension Schema.

Core Model
: The first layer of the CIM Schema, which includes the top-level classes and their properties and associations. The core model establishes the conceptual framework for the schema of the rest of the managed environment. Systems, applications, networks, and related information are modeled as extensions to the core model.

  The core model is both domain and platform-independent. See also Common Model and Extension Schema.

Desktop Management Interface (DMI)
: An initiative by the DMTF. The DMI allows desktop computers, hardware and software products, and peripherals (whether they are standalone systems or linked into networks) to be manageable and intelligent. It allows them to communicate their system resource requirements and to coexist in a manageable PC system. The DMI is independent of operating system and processor, enabling the development of manageable PC products and applications across platforms.

Distributed Management Task Force (DMTF)
: An industry-wide consortium committed to making computing resources and environments easier to use, understand, configure, and manage (www.dmtf.org).

Domain
: The class to which a property or method belongs. For example, if status is a property of Logical Device, it is said to belong to the Logical Device domain.

Event
: The occurrence of a phenomenon of interest. For example, an Event can denote the occurrence of a disk write error, a failed authentication attempt, or even a mouse click.

eXtensible Markup Manguage (XML)
: A simplified subset of SGML that offers powerful and extensible data modeling capabilities. An XML Document is a collection of data represented in XML. An XML Schema is a grammar that describes the structure of an XML Document.

Extension Schema
: The third layer of the CIM Schema, which includes platform-specific extensions of the CIM Schema such as Microsoft Windows NT, UNIX, and Microsoft Exchange Server. See also Common Model and Core Model.

Extrinsic Method
: A method defined on a CIM Class in some Schema that is unique to that class (*versus* intrinsic methods which apply across all classes). See also Intrinsic Method.

HTTP (Hypertext Transfer Protocol)
: An application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic stateless protocol that can be used for many tasks through extensions of its request methods, error codes, and headers.

HTTP Server
: OpenPegasus uses a small footprint special-services “lightweight” server that processes HTTP requests and returns standard HTTP responses. The server is not intended as a replacement for a Web Server. The server does not serve up HTML web pages and does not run CGI applications.

Indication
: The representation of the occurrence of an Event.

Inheritance
: The relationship that describes how classes and instances are derived from parent classes, or superclasses. A class can spawn a new subclass, also called a child class. A subclass contains all the methods and properties of its parent class.

  Inheritance is one of the features that allows the CIM classes to function as templates for actual managed objects in the CIM environment.

Instance
: A representation of a real-world managed object that belongs to a particular class, or a particular occurrence of an event. Instances contain actual data.

Instance Provider
: A type of provider that supports instances of system and property-specific classes. Instances providers can support data retrieval, modification, deletion, enumeration, or query processing. Instance providers can also invoke methods See also Class Provider and Property Provider.

Intrinsic Method
: A method defined for the purpose of modeling a CIM Operation. Standard intrinsic methods (such as `enumerateInstances`, `getInstance`, `modifyInstance`) are relevant to all classes. See also Extrinsic Method.

Kerberos
: A security mechanism that provides authentication, data integrity, data privacy, and mutual authentication.

Key
: A property that is used to provide a unique identifier for an instance of a class. Key properties are marked with the `Key` qualifier. A compound key has more than one property, with a `Key` qualifier.

Key Qualifier
: A qualifier that must be attached to every property in a class that serves as part of the key for that class.

Lightweight HTTP Server
: A small footprint server that processes HTTP requests and returns standard HTTP responses. The server is not intended as a replacement for a Web Server. The server does not serve up HTML web pages and does not run CGI applications.

Local Property
: A non-system property defined for a class but not inherited from a superclass.

Managed Object
: A hardware or software system component that is represented as an instance of the CIM class. Information about managed objects is supplied by data and event providers, as well as by the CIM Object Manager. Examples include a network interface card, an operating system kernel parameter, a system user, a print spooler.

Managed Object Format (MOF)
: A compiled language for defining classes and instances. A MOF compiler takes information from a .mof formatted text file and adds the data to the CIM Object Manager repository. MOF eliminates the need to write code, thus providing a simple and fast technique for modifying the CIM Object Manager repository. The DMTF makes their schemas available as MOF files.

Management Application
: An application or service that uses information or request operations originating from one or more managed objects in a managed environment. Management applications retrieve this information and request operations through calls to the CIM Object Manager from the CIM Object Manager.

Management Information Base (MIB)
: A database of managed objects, written in text.

Management Information Format (MIF) Database
: Part of DMI that stores and manages information and passes it to management applications on request. MIFs define the standard manageable attributes of PC products in categories including PC systems, servers, printers, LAN adapters, modems, and software applications.

Management Interface (MI)
: The MI allows DMI-enabled applications to access, manage, and control desktop systems, components, and peripherals.

Metamodel
: A CIM component that describes the entities and relationships representing managed objects. For example, classes, instances, and associations are included in the metamodel.

Metaschema
: The metaschema is a formal definition of the model. It defines the terms used to express the model and its usage and semantics.

Method 1. A function describing the behavior of a class. Including a method in a class does not guarantee an implementation of the method. The `Implemented` qualifier is attached to the method to indicate that an implementation is available for the class.

2. A function included in a CIM Object Manager API interface.

MOF File A text file that contains definitions of classes and instances using the Managed Object Format (MOF) language. Such files can be used to load descriptions of schemas supported by providers (via the `cimmof` command).

Multiple Inheritance
The ability of a subclass to derive from more than one superclass.

Multiple Operation
A CIM request that requires the invocation of more than one method.

Named Element
An entity that can be expressed as an object in the metaschema.

Namespace A unit for grouping classes and instances to control their scope and visibility. Namespaces are not physical locations; they are more like logical databases containing specific classes and instances. Objects located within a namespace must have unique names (specified by one or more key values) within that namespace. Objects in a different namespaces can be unique even if they have the same keys, because the two objects reside in separate namespaces.

namespace A directory-like structure that can contain classes, instances, and other `namespace`s. Objects are located within a `namespace`, and have unique names (specified by one or more key values) within that `namespace`. Objects in different `namespace`s can have the same keys, yet are unique since they reside in separate `namespace`s. Access to a `namespace` can be restricted to an authorized set of users.

Object Path A formatted string used to access namespaces, classes, and instances. Each object on the system has a unique path that identifies it locally or over the network. Object paths are conceptually similar to Universal Resource Locators (URLs).

Open Database Connectivity (ODBC)
A specification for an API that defines a standard set of routines with which an application can access data in a data source.

Operational Semantics
The formalization of real objects by putting them into a common language.

Override Indicates that the property, method, or reference in the derived class overrides the similar construct in the parent class in the inheritance tree or in the specified parent class.

Pluggable Authentication Model (PAM)
A product that coordinates user authentication tools for system security.

Property A name/value pair that describes a unit of data for a class. Property names cannot begin with a digit and cannot contain white space. Property values must have a valid Managed Object Format (MOF) data type.

Property Provider
A type of provider that supports the retrieval and modification of the CIM properties.

Provider An executable that can return and/or set information, execute methods, generate indications, or respond to other requests regarding a given managed object.

Provider Data Sheet (PDS)
Provides basic provider information to software professionals who will design, implement, enhance, and/or support client applications that will use this provider. It contains information about what this provider does, what interfaces it uses, how to install it, and what platforms and operating systems are supported.

Provider Registration
A provider needs to register with the CIMOM so that the CIMOM will know what properties and methods are supported. (Dev G) A provider must be registered with the CIM Server so that the CIM Server will know what properties and methods it supports. A special object is created during registration to relate the information about the provider to the classes in the CIM Schema that the provider supports.

Qualifier A modifier containing information that describes a class, an instance, a property, a method, or a parameter.

Reference A special string property type that is marked with the reference qualifier, indicating that it is a pointer to other instances.

Repository This repository contains the definitions of classes and instances that represent managed objects and the relationships among them. The OpenPegasus repository is not available for use by clients or providers for static or persistent data storage. See also CIM Object Manager Repository.

Required Property
A property that must have a value.

SAN-Aware
Storage Area Network-aware.

Schema A collection of class definitions that describe managed objects in a particular environment.

Simple Network Management Protocol (SNMP)
A protocol of the Internet reference model used for network management.

Simple Operation
A CIM request that requires the invocation of a single method.

SOHO (Small Office/Home Office) A class of products which are typically more powerful and more expensive than those sold to consumers, but smaller than those generally used by large institutions.

Standard Schema
A common conceptual framework for organizing and relating the various classes representing the current operational state of a system, network, or application. The standard schema is defined by the Distributed Management Task Force (DMTF) in the Common Information Model (CIM).

Subclass A class that is derived from a superclass. The subclass inherits all features of its superclass, but can add new features or redefine existing ones.

Subschema A part of a schema owned by a particular organization. The Win32 schema is an example of a subschema.

Superclass The class from which a subclass inherits.

Unified Modeling Language (UML)
More information can be found on the Object Management Group web site (www.omg.org). A UML diagram is used to visualize an object-oriented concept such as the hierarchy of CIM classes, where each box represents a class of object. The arrows may be thought of as meaning “is-a-kind-of”, so that a Human is a kind of Primate. Humans, being Primates, have all of the Attributes (or Properties) or Primates, but have additional properties, such as Nationality. By convention, in a UML diagram of CIM classes, only properties specific to a class, and not those inherited from the parent class and those above, are shown. But the inherited properties are also understood to be present.

Web-Based Enterprise Management (WBEM)
A standard developed by the DMTF that defines network protocols for the communication of CIM objects and operations. WBEM is a set of management and Internet standard technologies developed to unify the management of enterprise computing environments.

Web Server Full-service Web Servers act as HTTP Servers. In addition, they have many other capabilities, such as running CGI scripts. Understanding the distinction between a limited-service HTTP Server and a full-service Web Server is critical to understanding security on OpenPegasus. OpenPegasus uses its own embedded HTTP Server (a lightweight server), not a Web Server.

xmlCIM A specification for the CIM document type, a specialization of the eXtensible Markup Language (XML) that describes how CIM objects and operations should be encoded using XML for communication over a network. The full xmlCIM specification can be found on the DMTF web site.