Pegasus Response to NIST Survey.

Authors: Karl Schopmeyer, The Open Group; David Dillard; VERITAS Software; Denise Eckstein, Hewlett-Packard; IBM Corp., Ed Boden/Warren Grunbok, IBM Corp.

Version 1.4
Date 15 February 2004, Updated 14 march 2004, Update 24 March 2004

This document is a response prepared by the OpenPegasus Steering Committee to a survey issued by NIST in February 2004 that contains a set of questions concerning status and characteristics of CIM Server implementations. This response defines the OpenPegasus response to these questions.

The responses are provided as colored paragraphs following each of the survey questions in the survey.

Note: Some references are made via hyperlink to OpenPegasus documents. Since many of these documents are OpenPegasus PEPs (Pegasus Enhancement Proposals) in the review process there may be two potential problems with accessing references, 1) they require a password since everything in the review process requires logon to review (we need a name for comments) 2) these references may change rapidly since they are review documents and the reference changes with each new submission of a document version for review. Therefore, we are also providing a copy of the current version of these hyperlinked documents attached to the response.

CIM/WBEM Platform, Product and Technology Survey

Stephen Quirolgico <steveq@nist.gov>
National Institute of Standards and Technology
January 30, 2004

The purpose of this survey is to help NIST gain a better understanding of available CIM/WBEM platforms, products and technologies. Your detailed answers to the following questions are greatly appreciated. If the answer to any question is proprietary (or cannot be fully answered because of IP) please indicate this as part of your answer.

1. General

The following are general questions regarding your WBEM product/platform.

- Product/Platform Name (e.g., Pegasus, WBEM Services, OpenWBEM, *etc.*): OpenPegasus. Note that despite the fact that "Pegasus" is freely used as the name, the full name and web site references are OpenPegasus (http://www.openpegasus.org) .

- Vendor/Organization (e.g., Open Group, WBEMSolutions, Sun, *etc.*): Note: If an organization, please list participating companies. OpenPegasus is organized as an open community project under the auspices of the Open Group (http://www.opengroup.org). The Open Group manages the continuing OpenPegasus project and maintains the resulting source code. OpenPegasus source code and participation in the project is freely available without cost or commitment to the Open Group (see http://www.openpegasus.org). The OpenPegasus Architecture team provides the guidance and control of the OpenPegasus architecture providing a forum for decisions on architecture and implementation of new features. This group meets regularly by Telecon and the meeting and its results are open. The OpenPegasus Steering Committee, which is the only controlled-membership group operated under the auspices of the Open Group OpenPegasus project, is the body which sets the directions and priorities of the project. It maintains the roadmap, and determines the release schedule. The Steering Committee is the "gatekeeper" which governs what functionality is accepted into the project. It reviews proposals and adds activities into the project plans when it is satisfied that they are resourced and will not have an adverse impact on existing developments. In addition it sets the procedures under which the project operates and is the ultimate arbiter in the cases where the technical teams are unable to achieve consensus. Membership in the Steering Committee is open to the Sponsor Companies, who are supporting the project through their financial contributions. The members of the OpenPegasus Steering Committee today are:

1. IBM Corp.
2. Hewlett-Packard
3. EMC Corporation
4. VERITAS Software Corporation

- Vendor/Organization Contact Name & Company: Mr. Martin Kirk (m.kirk@opengroup.org) is the program director for the Open Group and Mr. Karl Schopmeyer (k.schopmeyer@opengroup.org) is the project leader.

  Target Platforms (e.g., Linux, MS Windows, Solaris, *etc.*): OpenPegasus is coded primarily in C++ and was designed for portability with attention paid to a portability layer. Each release of the source code notes which platforms/compilers under which it is known to compile and has been tested (See the release notes with each release). The initial target was for Windows, Linux, and UNIX. Today, as of version 2.3.1, OpenPegasus the port status is as follows:
  - Windows (Microsoft Visual C 6 and 7 compilers) ported and supported.
  - Linux 32 bit systems (multiple distributions) with the GNU 2.9.x and 3.x compilers ported and supported
  - Linux 64 bit systems with the GNU compilers.
  - HPUX (HP has released OpenPegasus on multiple platforms including HP-UX, Linux, OpenVMS, and Tru64 UNIX. Refer to the following web site for details: http://www.hp.com/go/wbem.)
  - OpenPegasus will likely work on the major IBM platforms (AIX, OS/400, z/OS), but these platforms do not currently ship or directly support OpenPegasus today.
  - Solaris OS - ported and supported effective with OpenPegasus 2.4.
  - MacOS – There is a port in discussion today in the Pegasus Architecture Group but the issues of support, etc. have not been resolved.

  Note that the concept of supporting a particular operating system or compiler is a formal part of the OpenPegasus project. Supporting an environment within OpenPegasus means

  1) OpenPegasus is ported to the defined compiler/OS combination,
  2) It is tested on the environment,
  3) There is a commitment from a member of the OpenPegasus community to continue testing and support to assure that the port continues to work as new versions of OpenPegasus are produced.

1. Open source? (yes/no): Yes, OpenPegasus is open source. It is freely available from the Open Group (http://www.openpegasus.org) web site. OpenPegasus development continues today as the DMTF specifications grow and change. Both the source code under development (maintained in CVS) and regular functional releases are available (gzip and zip collections of the source files) for anonymous download. OpenPegasus makes use of other open libraries such as OpenSSL for security services and

ICU for localization but does not require these libraries for operation. These libraries such as OpenSSL are not part of OpenPegasus and are not distributed with OpenPegasus.

- Cost/licensing: OpenPegasus is freely available, there is no cost involved with using the OpenPegasus code. OpenPegasus is maintained under the MIT open source license. This license was chosen at the inception of the project as being the most liberal license commonly available for open source since the goal was to make Pegasus available, not limit its usage. It is an objective of the project that there is a single license for all of the code rather than having components under multiple licenses. Thus, all of the code available is maintained under this license with one exception. The Java Client code in the Pegasus CVS tree is maintained under a separate license (SNIA Public License Version 1.0) because that code was extracted from the SNIA CIMOM code (also an Open Group project) and is maintained under the current license for that code.

- Mailing lists and Product/Project URLs: Web Site: http://www.openpegasus.org. Note that this web site is open and provides open registration for the various mailing lists used by the OpenPegasus community. Membership in the mailing lists may be requested through this web site. OpenPegasus maintains a number of open mailing lists including the following: 1) general information and announcements (Pegasus-l), 2) Developers discussion list (Pegasus-devel), 3 CVS Commit history (Pegasus-commit), 4 Pegasus CVS writers (pegasus-cvs-writers), 5 Discussion on lightweight version of OpenPegasus (pegasus-lite). There is a requirement to register for these aliases and for some of the other functions on the web site (.e.g. documents in review) but that is only to provide a name, and email address. Use of the mail lists and access to the Open Pegasus Architecture Team and its processes and OpenPegasus documents in review does not require membership in The Open Group.

2. Architecture

The following questions concern the architecture of your WBEM product/platform. The general WBEM architecture is shown in Figure 1. Please feel free to include as much detailed information as possible.

- With respect to Item 1 in Figure 1, what programming languages are supported for CIM clients? Are example CIM client programs available? Today, OpenPegasus provides the interface APIs and code for C++ and Java clients. The public Client APIs are frozen to provide for binary compatibility between minor versions of OpenPegasus release. Example, test, and OpenPegasus CIM Server administrative programs are available in C++ and test programs available in Java. The Java code maintained in the OpenPegasus CVS repository is an enhanced version of the SNIA CIMOM (also an Open Group project) to match the level of functionality provided by OpenPegasus and tested for compatibility with OpenPegasus.

- What CIM operations are currently supported by your CIM/WBEM server? Which CIM operations are not currently supported? As of OpenPegasus version 2.3, all operations except for execQuery are implemented in OpenPegasus. With version 2.4, the code to support execQuery is implemented and is being tested. Today the DMTF defined process indications are supported but lifecycle indications are not yet supported. A core objective of the OpenPegasus project is to adhere to the DMTF specifications and to work to enhance these specifications in areas that they were vague or incomplete as part of development of the OpenPegasus implementation. Thus, when the OpenPegasus process indications support was implemented the OpenPegasus community generated a number of DMTF Change Requests to insure that the specifications were complete and implemented the OpenPegasus to match the specifications defined with these DMTF change requests. Today, the OpenPegasus community feels that both the Query functions and the lifecycle indications functions need further specification to be complete enough for interoperable implementation and we are working to achieve these goals.

- With respect to Item 3 in Figure 1, does your WBEM product/platform include a CIM XML encoder/decoder? Yes, OpenPegasus supports the CIM-XML interfaces defined by the DMTF specifications. CIM Client, CIM Server, and CIM Listener encoders and decoders are included that are compatible with the DMTF specifications. These encoders and decoders have received extensive testing in a number of interoperability tests including the SNIA (Storage Network Industry Association) formal interoperability tests and today the number of interoperability problems is minimal. Note that these encoders and decoders are written in C++ and are hand coded for the CIM-XML syntax rather than using standard parsers to enhance performance.

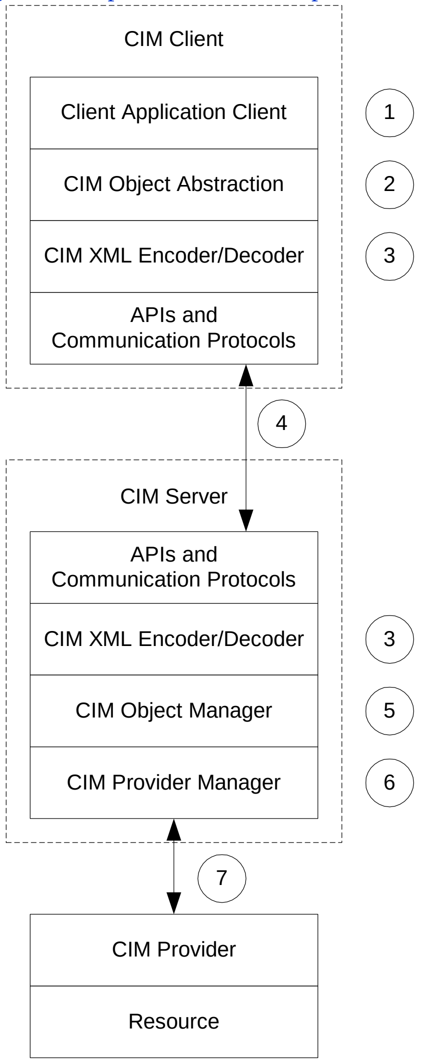

Figure 1. General WBEM Architecture.

- With respect to Item 4 in Figure 1, what communication protocols are currently supported (e.g., HTTP, SOAP, CORBA, RMI, raw sockets, *etc.*)? Can the supported protocols support callbacks (to the client)? If the CIM client resides on the same node as the CIM server, is there an API for directly accessing your CIM server? If so, what programming languages do these APIs support? Today OpenPegasus implements the CIM-XML HTTP protocol as defined in the DMTF specification and a local connection (connectionLocal()) protocol that uses Unix Domain Sockets to establish a local connection to the CIM Server. The DMTF specifications do not provide for callbacks from the CIM Server to the CIM Client. They do provide for indications that are driven from subscriptions and

OpenPegasus implements the CIM-XML indications support protocols (process indications today). If the CIM client interface is required within the CIM Server (e.g. a provider) there is also a binary client interface that provides the same CIM Operations but without the XML mapping.

- With respect to Item 5 in Figure 1, how are client queries processed by your CIM object manager? Include details on how queries are mapped to repository queries as well as how the CIM object manager interacts with the CIM provider manager. Also include details on APIs or standardized technologies used in your CIM object manager. Note that this is a complex question that requires more than a paragraph response. We have attached other documentation (presentations, etc.) describing the overall architecture as response to the architecture question. Generally OpenPegasus is a message-based server. CIM Operations received by the server are mapped into messages and move through the server functions via queues for each processing function. These messages are only converted to direct APIs at the interface to the repository and the providers. The major server functions (communication, encoding/decoding, dispatching, repository, provider management and interface) are each separate internal functions and process operation messages from input queues to output queues.
  The major operation-processing components of the OpenPegasus CIM Server are:
  - Communications – HTTP communication components handles the HTTP headers and TCP communications.
  - Authentication – Handles the authentication of requests.
  - XML encoding and decoding modules – Provide mapping between XML form of operations and responses and OpenPegasus internal operation objects.
  - Operation Dispatcher and Operation aggregator – Dispatches operations to providers based on the provider registration and the repository and aggregates responses from multiple providers into a combined response. This handles all of the defined operation types.
  - Provider Managers – Provides the direct interface between the APIs defined for a particular provider type (.e.g. enumerateInstances as a C or C++ call) and the operations messages in the CIMOM. The provider managers provide multithreaded support for the providers so that multiple requests can be handled simultaneously
  - Repository – Maintains class definitions and static instance information. OpenPegasus provides both a class and instance repository. Generally today, the instance repository is used as the default access of no specific provider is defined for an instance or association operation. The class repository is the only destination for class and qualifier operations because the concept of the class provider is not implemented in OpenPegasus
  - Providers in C++ or in C using the CMPI interfaces (a part of the IBM SBLIM project).

  Additional functionality provided by the OpenPegasus CIM Server environment includes:
  - CIM Server startup and shutdown processes.
  - Indication service – Routing of indications both to indication services and internally within the CIM Server to providers. Note that there is also a function to accept indications from other environments (CIM Listener) integrated into the OpenPegasus CIM Server.
  - Indication Subscription services – Accept indication subscriptions and maintain persistent indication subscriptions.
  - Indication Handler services that allow multiple types of indication handlers (today there are handlers for SNMP and CIM-XML)
  - Provider Management including registration of providers, provider load and unload, management of indication filters for providers, etc.
  - Functionality to manage configuration parameters, users, etc. of the CIM Server.
  - Threading and message management services that provide the intercommunication between the components of the OpenPegasus server.
  - Facilities for logging, tracing the flow of operations through the server, statistic gathering and other server support functions.
  - MOF Compiler that operates either in standalone mode (directly writes to the repository) or using the CIM Client interface so that it can install MOF directly into the operating OpenPegasus CIM Server.

- With respect to Item 6 in Figure 1, how are CIM providers managed by your CIM Provider manager. Include details on how registrations are handled. The OpenPegasus provider manager provides a number of functions including:
    1. Provider loading,
    2. Call for each of the provider operations (.e.g. getInstance, enumerateinstances, etc.),
    3. Thread management so that the providers can operate on multiple operations simultaneously,
    4. Management of indication support so that providers are informed of requirements for indication generation,
    5. Provider unload and reload.

  The relationship between classes and providers (provider registration) is controlled through a provider registration mechanism based on a set of provider registration classes. Providers are registered in the OpenPegasus system by creating instances of the Provider Registration classes (ProviderModule, ProviderCapability, etc.) and installing these instances into the repository (either through the compiler or directly through the create instance functions). Note that there is a utility program to provide status of provider registration and to allow de-registering providers. Providers may be installed in a set of directories defined in the OpenPegasus configuration.

  OpenPegasus version 2.3 provides a single, fixed provider manager that manages providers as separate libraries using C++ as the interface language.

  Effective with OpenPegasus version 2.4 (now in development) multiple, pluggable provider managers are supported (this code has already been completed and is in test now). This allows providers to be defined in other languages, possibly the addition of remote providers, providers with different privileges, etc. In effect it creates two levels of routing of operations to providers 1) the provider manager, 2) the provider within a provider manager. Currently the first test case for the multiple pluggable provider manager is the CMPI C language provider functionality which is being tested in parallel with the pluggable provider manager extension.

- With respect to Item 7 in Figure 1, how do CIM providers find and register with a CIM server. Include details on discovery protocols used, configuration of CIM providers, and APIs used by CIM providers to register with the CIM server. In OpenPegasus providers are built as separate shared libraries (.e.g. dlls in Windows). Providers register with a CIM Server by installing instances of the Provider registration classes (OpenPegasus does not support the provider qualifier). These classes (providermodule, providercapability, etc.) define the provider type, location, classes supported, namespaces supported, etc. so that the providers can be loaded in response to operation requests or indication subscriptions. Thus the process of creating a provider might be
    - 1) write the provider,
    - 2) define create the instances of the provider registration classes,
    - 3) place the provider runtime in the required directory,
    - 4) install the registration class instances using the compiler,
    - 5) test the provider.

  Note that a provider could be installed registered, unregistered, and removed without stopping the OpenPegasus server. Although it is possible to define a utility to install providers in place of using the compiler, to date the OpenPegasus community has not found this necessary. Further, since today providers are shared libraries, not separate executables, they cannot self-register. The OpenPegasus provider registration classes are based on prototypes of the DMTF Interop provider registration classes which today are experimental class definitions in the DMTF. Experimental classes appear in preliminary CIM releases but not in the final releases. Thus, they appeared in CIM 2.8 preliminary but not in CIM 2.8 final.

  Today the DMTF specifications do not provide any provider discovery protocol. However, a CIM Server discovery mechanism was recently approved that uses SLP (Service Location Protocol) and is included in OpenPegasus now both for version 2.3 and version 2.4. This protocol will allow the discovery of CIMOMs and starting with CIM 2.8 of registered profiles of management functionality.

Once your WBEM product/platform is installed, how do previously-installed CIM providers detect its existence? OpenPegasus today does not include an automatic way to detect previously-installed providers on a NEW CIMOM installation since the provider registration is maintained in the repository. However, once provider registration is completed the registrations remain in the repository so that upgrades to the CIMOM, etc. do not change the registrations. Today the providers defined in the OpenPegasus distribution are registered as part of the repository creation, a process that builds the base repository with the CIM classes, installs special OpenPegasus classes, and also installs the registration for providers and other functions normally used by OpenPegasus.

- Does your CIM server support SNMP/MIB? Include details on how your CIM server acquires MIB information and how it maps this information to CIM.
  To date, the work in the Pegasus community has concentrated on:
  - implementing the CIM Server, CIM Client, CIM Listener and common provider functions,
  - creating a production quality open-source product,
  - satisfying the needs of the current Pegasus users

  The demand for either a specific or general SNMP provider has not yet risen to the level where the community is ready to create or contribute such a provider.
  However, OpenPegasus does support SNMP notification for indications with an SNMP indication handler that generates SNMP traps.

- Does your WBEM product/platform handle additional management information models? OpenPegasus was written as a DMTF compliant management system implementation. It was written around the DMTF CIM models and the CIM Operations over HTTP specification. The architecture supports the addition of interfaces with other management protocols in a number of ways 1) Indication handlers that transmit indications to other systems are an extendable feature (CIMXML and SNMP indication handlers are supported today) and providers can be added to access information from other management protocols (.e.g. SNMP). To date nobody has written a standard SNMP provider for OpenPegasus. OpenPegasus does provide a mapper to WMI that provides mapping between the Microsoft WMI information that is maintained in older versions of the CIM model and with the Microsoft COM protocols may be mapped into standard current CIM-XML. This WMI mapper may be used with OpenPegasus or completely separately.

3. Additional Information

The following questions are intended to provide additional information about your WBEM product/platform.

- How closely does your WBEM product/platform conform to the current CIM specifications? Providing a DMTF CIM/WBEM implementation that is both functionally complete and compliant with the DMTF specifications including CIM Client, CIM Server, and CIM Listener is a key goal of the OpenPegasus project. The OpenPegasus community is heavily involved with the DMTF specification process and both works with the current specifications pushes to improve the specifications. However, we recognize that we are not 100% compliant with any given OpenPegasus release and try to list the deviations and level of compliance with each release. As noted in another section of this document we have delayed implementation of some functions specifically because we feel that the level of specification is not sufficient to insure effective, interoperable implementation (.e.g. Query because the query language was not supported by an approved specification and lifecycle indications).

- How extensive is the available documentation? Does the documentation include detailed programming examples? Does it include a detailed specification of your WBEM product/platform architecture? While not systematic, OpenPegasus includes extensive documentation in a number of areas including the following:

  - 1) Public APIs are documented in the header files using a standard form (DOC++) and a document is in process today using this documentation. NOTE: The OpenPegasus release process insures that public APIs are frozen and provides for backward compatibility.
  - 2) HOWTO documentation is available on most of the utilities, tools, and test programs.
  - 3) Documentation from a continuing series of technical workshops is available on the OpenPegasus web site (the last workshop was in July 2003).

  There is an extensive set of working documents on the OpenPegasus development process, the design decisions made and other current issues on the OpenPegasus web site in the form of PEPs . These documents represent many parts of the OpenPegasus project and design from the definition of each release to the definition of particular components of the design.

- Is your WBEM product/platform designed for a specific operating system? No, OpenPegasus was designed from its inception to be portable. It has been ported to a wide variety of different systems to date and typically we have found that porting is not a major effort although it does depend on knowledge of the OpenPegasus code base. There is a single set of files that represent the majority of system dependent functionality and the OpenPegasus community continues to work to insure that OS and compiler specific issues are handled through these platform/system interfaces. Further, the build system was designed specifically to allow system build and maintenance on multiple different platform types with a minimum of dependence on extra tools for each platform. Thus, despite the fact that Linux and Unix are major targets, the same build system works on Windows with only two tools (GNU Make and an OpenPegasus tool to bring Windows functions in line with Linux).

- Can your WBEM product/platform be used on MS Windows platforms? If so, does it work in conjunction with WMI, or is it separate from WMI? Yes OpenPegasu can be used on Windows and compiles with the Microsoft compilers. A WMI mapper (a separate OpenPegasus CIM Server) is part of the OpenPegasus source tree and has already been used by multiple users as a bridge to WMI management information.

- Does your WBEM product/platform have a way of advertising itself to CIM providers or managed resources? The DMTF specifications include a discovery mechanism today (the Service Location Protocol with a DMTF defined discovery template). This allows discovery of CIM Servers that implement the SLP service agent including both the characteristics of the CIM Server and the profiles of entities managed (registered profile). OpenPegasus implements these protocols today so that CIM Servers are discoverable with OpenPegasus version 2.3.2 and registered profiles of management functionality with OpenPegasus version 2.4.

- How is/has your WBEM product/platform been tested? Has it been tested using commercially-available managed resources, or vendor-specified CIM providers? OpenPegasus includes a test suite with the source code including an extensive and growing set of both unit and end-end tests. Part of the process of producing new functionality in OpenPegasus is to provide the test capabilities for that new functionality so the test suite grows with the platform. A number of the major OpenPegasus users also employ additional testing with commercial tools for code coverage, memory leak, and other standard testing but in the spirit of open source, the project tries to define open source testing mechanisms as much as possible for the environment and to insure that testing can be done on any platform. The test suite is designed to operate on ANY environment although today there are variations between platforms.

  Please include details on specific resources that have been managed using your WBEM product/platform. There are a number of projects in process that utilize the Pegasus implementation including:
  - IBM pSeries (AIX) Pegasus delivered as part of Linux toolbox CD - Linux SBLIM providers available from SBLIM web site - Instrumentation links can be found at http://www-

124.ibm.com/sblim/instrumentation.html IBM developed CMPI, a standardized provider interface supported by multiple available CIMOMs.
  - There are a number of other projects in process in which OpenPegasus is utilized but the companies involved consider this proprietary information today.

- Describe how your WBEM product/platform is distinguished from other products/platforms. Include in your description the advantages of using this WBEM product/platform over other product/platforms.
  OpenPegasus defined several broad objectives at the inception of the project including:
    - **Portability** – based on C++ as the primary coding language but portable over a wide variety of platforms
    - **Lightweight** – lightweight as possible with the limitations of the language and requirements
    - **Standards compatible** – a true implementation of the DMTF CIM and WEBM specifications.
    - **Production Quality** – high quality and speed that it can serve in high availability and high throughput environments
    - **Continuing project** – a project that continues to grow, supporting the existing version and improving the CIM/WBEM functionality, flexibility, stability, and throughput with new versions and feeding back to the DMTF and Open Group specifications for CIM/WEBM environments.
    - **OpenPegasus is an open source project** and an open development environment that is freely available under a license that minimizes limitations on its usage.
    - **Modularity –** It was intended from the beginning the OpenPegasus be modular so that there would be a small core and many of the components of a CIM server could be attached without modifying this core.

  While we have not yet met all of those goals, the OpenPegasus community continues to grow the product with these key goals in mind.
  Continuing community-based Process/Project - There is a community based process for OpenPegasus, a user and developer community with documented procedures and processes. Through the PEPs, the community has evolved a process for making decisions, defining versions, documenting, and continuing to work together towards a common goal.
  OpenPegasus is an open project with open source. The community in general contributes to each new version of OpenPegasus, not a particular supplier or user. OpenPegasus continues to grow and improve through the group decisions of a number of users, some of them major IT suppliers. There is no private code in OpenPegasus and the license used is the most liberal that existed at the time the project started.
  The project and usage continues to grow in users, both the core users represented by the OpenPegasus Steering Committee and the other users who freely comment on, use, and occasionally contribute back to OpenPegasus.
  There is a continuing effort to improve stability. Stability and performance are major objectives with OpenPegasus and there are defined quality objectives for each release version. There are definite criteria that must be met for a OpenPegasus release (see the PEP for each release for details) including bug levels, code coverage in testing, and stress testing over a defined time limit. While some of the releases have not met all of those goals, in each version the goals are defined and it is a community decision if we choose to make a release without meeting all of these goals. Note that the OpenPegasus code base includes not only the CIM Server and CIM Client code but 1) a number of client utilities and test tools, 2) a growing set of providers including sample providers, test providers and some of the core providers for system management. OpenPegasus also includes the complete test suite used in the continuing unit test and end-end test of OpenPegasus.
  Portability - We have already demonstrated that OpenPegasus is portable to a wide variety of platforms based on the current list of tested and running ports. It is also clear that most of these ports do not take a significant amount of time. Therefore, we feel that Pegasus is portable to even more platform and compiler types, e.g. VxWorks, with a modest effort.

  The OpenPegasus community feels that the key strengths of OpenPegasus are:
  - **OpenPegasus is a community project** – It is not the output of a single company nor under the control of a single company. The OpenPegasus community provides a broad base for

requirements development, for implementers, and for users. It helps assure that it is not simply the whim or directions of a single company. The community process operates under a set of rules to assure that the decisions made are community decisions. This community process is also open (documentation, discussion, bugs, proposals, plans, etc.) to anybody, users or developers. The Open Group protects the open source nature of the result and promotes coordination between the community and the development of specifications and standards.

  - **The OpenPegasus community defined processes** which insure that there is an orderly progression of activities from conception to release that is understood and approved by the OpenPegasus community (i.e. the OpenPegasus users).
  - **Cooperation with DMTF** – There is a close cooperation between members of the OpenPegasus community and the DMTF. For example, OpenPegasus community members are key members of the DMTF work groups and directly affect many of the DMTF specifications. This allows OpenPegasus implementation to directly affect the directions of the standards to help insure interoperability and completeness and for OpenPegasus to correctly implement the specifications as they become available.
  - **Portable** –OpenPegasus has proven portable to a number of different types of environments, not simply to different flavors of Unix. The fact that it has been ported to such widely different environments as VMS, Windows, OS400 indicates that the base portability assumptions were correct and that the simple build mechanisms are also portable.
  - **Complete CIM/WBEM environment** –OpenPegasus is a complete environment with all of the components required to implement CIM management solutions including compilers, the CIMOM, the CIM Server, defined and public APIs for client and provider, sample clients and providers, a growing set of real providers and an integrated test environment.
  - **Message based architecture** --This architecture provides for effective server design using threads to enhance parallel processing within the environment.
  - **Commitment to RAS (reliability, availability, serviceability)** – The OpenPegasus community is committed to producing a quality product as open source that is not simply a code base but a working product with through testing meeting defined criteria for functionality and quality.

- Describe the limitations of your WBEM product/platform. In general there are two types of limitations, those that are inherent in the design/implementation and those that are transient, and may change with a particular release version.
  Inherent Limitations
  - Footprint –There are continuing efforts to reduce footprint and to provide options so that users can limit functionality and reduce footprint. Obviously the combination of functionality implemented and programming language chosen set limits on the footprint.
  - Porting required to new platforms – Because C++ was chosen as the base implementation language, porting is required for new environments. There is a level of expertise required to accomplish this porting. While the OpenPegasus environment is structured to enhance porting, it is remains an effort to map the OpenPegasus system functions to any given operating system, hardware, and compiler environment.
  - Portability limits of the C++ language, particularly binary compatibility between compilers. This means that C++ providers have definite compiler limitations. The solutions for OpenPegasus is to offer a number of provider interfaces (C, Java, C++) with interfaces that are standardized so that providers from other environments can be used.

  Transient Limitations
  - There are a number of limitations left over from the original code that have not yet been corrected including: 1)the repository implementation is today simplistic and verbose, 2)there are some compiler limitations that deviate from the CIM specifications (.e.g. no alias functions today), 3) digest authentication is not implemented. These examples represent areas where the original code had limitations and open areas that must be refactored and extended as the platform moves forward. Many of this type of transient limitation are on the work list for the next version.
  - Functionality limitations such as the execQuery and lifecycle indications which are not implemented today. These limitations are being corrected as the specifications stabilize and

    the OpenPegasus users demand the functions. There will always be these differences as the specifications from DMTF grow and change. Also, the speed at which these functions are incorporated into OpenPegasus dependents on the commitment and priorities of the OpenPegasus developer and user community, not on the requests of outside users who make no contributions back to the community. OpenPegasus is a community project which requires community commitment to produce the code, not simply commitment to use it.
  - Dynamic Memory Utilization – Today OpenPegasus builds complete responses and the XML messages for these responses before sending. This can result in high temporary memory usage for large responses. The work to modify this is on the work list for version 2.4. Note that the providers today provide incremental responses back to the CIMOM so that the extensions required are in the CIMOM and HTTP protocol, not the providers.
  - Providers in the CIMOM process – Today the OpenPegasus providers operate in the CIMOM process so that any provider failure is also a CIMOM failure. The extension to include out-of-process provider support is on the work list for version 2.4.

- Can your WBEM product/platform interoperate in a heterogeneous, distributed environment? Aside from the CIM schema, what standards are employed by your WBEM product/platform to facilitate interoperability in a heterogeneous, distributed environment. Include in your answer a description of tests and benchmarks used to assess interoperability. OpenPegasus is designed to operate in heterogeneous environments:
  - it is portable, it adheres to the DMTF standards, and will adhere to Open Group API standards (ex the CMPI standard that is in process today) as they are released,
  - it is regularly tested on multiple platforms as part of the development and release process,
  - it is regularly tested in interoperability tests such as the SNIA interoperability tests.
  - It utilizes the distributed management features of CIM/WBEM including the concept of cross-environment associations.
  - It completely separates the client from the server with the CIM-XML protocols.
  - It is planned that the OpenPegasus will support distributed providers in the future.

  In addition to the CIM Schema, OpenPegasus also adheres to the DMTF specifications for CIM Operations over HTTP and the Specification for the Representation of CIM CIM in XML for communications.

- Describe major enhancements, bug-fixes, *etc.* that will be made within the next 12 months. OpenPegasus normally releases about 2 revisions per year. In 2003 OpenPegasus version 2.2 and version 2.3 were released. The current work is on version 2.4. The list of functionality for each release is in the OpenPegasus PEPs which act as the community working document to control the definition, development, and testing for each release. PEP 97 is the controlling document for release 2.4; the list of functionality, responsibilities, status, etc. for this release can be found in that document. Some of the major enhancements planned for version 2.4 (to be released about August 2004) include the following:

  - Pluggable Provider manager and CMPI C Language provider interface. This code is in the process of being tested now and will be part of the first evolution of this release
  - execQuery support and support for the DMTF CQL query language.
  - Out-of-process provider support.
  - Remote provider support.
  - Update of the MOF compiler to include alias functionality.
  - Dynamic memory usage management through use of HTTP chunking and detailed error reporting to eliminate the memory spikes that occur with large responses.
  - Refactoring of a number of components including the repository and messaging components.
  - Lifecycle indication support including embedded objects.
  - Java provider support through a Java Provider Manager.

  In addition, OpenPegasus maintains a bug list on the web site (http://cvs.rdg.opengroup.org/bugzilla) that is available for review and contribution of new bugs.

Within the community process there is a regular bug reporting, review, and correction process and bug fixing is a normal part of the development and maintenance of OpenPegasus releases.

Generally the OpenPegasus project looks about one version ahead with firm plans but maintains a general roadmap of work that is requested. This roadmap is maintained as a review document on the OpenPegasus web site, PEP 131, .

Attached Documentation

1. PEP 131, Pegasus Roadmap.
2. OpenPegasus Technical Workshop Presentations (July 2004)
3. PEP 97, OpenPegasus Version 2.4 Planning