OCPI

WBEM Server Benchmarks

Nortel Networks

Issue Date: December 3, 2003
Revision: 0.5
Document Status: Work in Progress
Document Number: ATI-2003.723
Issue A
Security Status: Open
Authors: Ying Zeng, Chris Hobbs, John Bell, Brian Quirt

Copyright Nortel Networks 2003
This document is the property of Nortel Networks who own the copyright therein.
Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>August 26, 2003</td>
<td>First draft</td>
</tr>
<tr>
<td>0.2</td>
<td>September 09, 2003</td>
<td>Introduction addition, changes from the first review.</td>
</tr>
<tr>
<td>0.3</td>
<td>October 1, 2003</td>
<td>First set of result on class tests in section 6.3.1, and test script re-organization in section 5.2.2, and re-addition of property operations in section 4.2.2.</td>
</tr>
<tr>
<td>0.4</td>
<td>October 23, 2003</td>
<td>Addition of section 5.2.3, and Appendix B - D. Update of results for instance operations in section 6.3.2 and 6.4.1.</td>
</tr>
<tr>
<td>0.5</td>
<td>October 24, 2003</td>
<td>General test condition update</td>
</tr>
</tbody>
</table>

Contributors

ATI, BCM OAM team
# Table of Contents

1 Introduction
2 Choices of WBEM Implementations
3 General Benchmark Conditions
4 Benchmark Feature Specification
   4.1 Repository Operations
   4.1.1 CIM Class Operations
   4.1.2 CIM Instance Operations
   4.1.3 CIM Property Operations
   4.1.4 CIM Association Operations
   4.1.5 CIM Qualifier Operations
   4.1.6 ExecQuery Operation
2.1.2 Provider Operations
   4.2.1 CIM Instance Operations
   4.2.2 CIM Property Operations
   4.2.3 CIM Association Operations
   4.2.4 CIM Extrinsic Method Operations
3 Indications
   4.3.1 Indication Test Setup in openPegasus
   4.4 Client Library XML Encoding/Decoding
4 Benchmark Test Instrumentation and Implementation
   5.1 Instrumentation
   5.1.1 Option 1: Instrumentation at Client Session End
   5.1.2 Option 2: Instrumentation at Client Protocol End
   5.1.3 Option 3: Instrumentation at CIMOM Server Protocol End
   5.1.4 Option 4: Instrumentation using Binary Interface
   5.2 Implementation
   5.2.1 wbemexec Utility
   5.2.2 Test Suite
   5.2.3 Test Providers
5 Benchmark Results
   5.1 HTTP Session Setup Time
   5.2 HTTPS Session Setup Time
   6 Repository Operations
   6.1 Class Operations
   6.2 Instance Operations
   6.3 Provider Operations
   6.4.1 Instance Operations
   6.4.2 Property Operations
   6.4.3 Association Operations
   6.4.4 Indication Operations
6 Future Work
7 References
8 Appendix A - Sample XML Messages
9 Appendix B - Instrumentation in the wbemexec Utility
10 Appendix C - Bugs/Problems in WBEM Implementations
11 Appendix D - Sample Test Providers and MOFs
   12.1 Sample MOF for Instance Provider Tests
   12.2 OpenWBEM Sample Instance Provider for Condition A
   12.3 OpenWBEM Sample Instance Provider for Condition B

Nortel Networks
1 Introduction

This document describes a methodology and a set of tests that have been constructed to quantify the performance characteristics of systems that implement the CIM management model. The purpose of these benchmarks is to provide the results necessary to compare different implementations of a CIM system, as well as providing the data necessary to determine if the resource budgets for target managed elements can be met.

The major system components under consideration are the client/server protocol, the CIM Server (or CIMOM), the repository and the CIM providers. The instrumentation interface will reside at the CIM client.

The benchmarks will perform measurements along a number of different dimensions: the latency for a set of CIM operations (where an operation describes a management - i.e., monitor or control - action on a CIM modeled resource); the in-memory requirements for the CIM server; and the persistent-storage requirements for the repository. The performance and storage requirements will be gauged by discretely varying the size of the models and the number of instances in the system. The dimensions to be measured are seen as external attributes of the system. Treating the system as a black box has the advantage of being able to infer the properties of the system using external instrumentation interfaces. This in turn means that the observations will have negligible impact on the overall system performance while also providing the means to perform an apples-to-apples comparison of different CIM implementations. This methodology allows the various elements of a CIM system to be characterized and compared. For example, changing the size of the model and the number of instances under consideration will provide a clear indication of how efficient the object encoding is (the in-memory
requirements), how efficient the object serialization is (the persistent-memory requirements) and what effect varying these dimensions has on the observed performance of CIM operations (the latency).

<table>
<thead>
<tr>
<th>Functional Group</th>
<th>CIM Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Read</td>
<td>GetClass, EnumerateClasses,</td>
</tr>
<tr>
<td></td>
<td>EnumerateClassNames, GetInstance,</td>
</tr>
<tr>
<td></td>
<td>EnumerateInstances, EnumerateInstanceNames,</td>
</tr>
<tr>
<td></td>
<td>GetProperty</td>
</tr>
<tr>
<td>Basic Write</td>
<td>SetProperty</td>
</tr>
<tr>
<td>Schema Manipulation</td>
<td>CreateClass, ModifyClass, DeleteClass</td>
</tr>
<tr>
<td>Instance Manipulation</td>
<td>CreateInstance, ModifyInstance, DeleteInstance</td>
</tr>
<tr>
<td>Association Traversal</td>
<td>Associators, AssociatorNames, References, ReferenceNames</td>
</tr>
<tr>
<td>Query</td>
<td>ExecQuery</td>
</tr>
<tr>
<td>Qualifier Declaration</td>
<td>GetQualifier, SetQualifier, DeleteQualifier,</td>
</tr>
<tr>
<td></td>
<td>EnumerateQualifier</td>
</tr>
</tbody>
</table>

Table 1: Benchmark test sets for extrinsic method operations

CIM operations can be broken down into functional groups, each supporting a number of set of method calls. The following table lists the functional groups a the operations associated with each group. The benchmarks are designed to measure the performance characteristics for all of the operations within each group. A further discrimination is made between on those operations that are performed by both intrinsic methods and extrinsic methods. (Intrinsic methods operate on the CIM model and are internal to the CIM Server while extrinsic methods are executed by a provider.) Distinguishing between intrinsic and extrinsic operations characterizes the performance overhead of invoking a provider. A final set of tests characterize the performance of indication operations – of particular interest here is what effect indication filters have on event delivery.
2 Choices of WBEM Implementations

There are several choices of WBEM server implementations for benchmarks:

- OpenPegasus

This is an open source implementation driven by IBM, HP, EMC, etc. Currently this implementation has a lot of activities and development work being done, and it attracts much industrial attention. The advantages of this implementation are that the repository is implemented in XML which is easy to debug, and that it also comes with a debug client. However, the disadvantages are that the repository is way too large, and a few CIM operations are NOT yet supported. The version used in the benchmark tests is release 2.2.

- OpenWBEM

This is an open source implementation driven by Center7. The openWBEM implementation has a better architecture than openPegasus, and most of the CIM operations are supported. It is also smaller in repository size and has a slightly better memory footprint. However, the openWBEM implementation appears to have less industrial interests at this point. The version used in the benchmark tests is release 2.0.6.
3 General Benchmark Conditions

Unless otherwise stated, all benchmark results described in this document were obtained under the following conditions:

- Platform - Debian Linux running kernel 2.4.18; if available, all benchmark tests will be performed on a PPC G4 machine.
- 100Mb/s LAN connection
- CPU speed - Pentium III 550MHz, 512KB CPU Cache, RAM size - 128MB
- WBEM client and WBEM server are running on the same computer
- WBEM client and WBEM server are running on different computers given that the computers are connected with a 100bps Ethernet connection. The Ethernet connection is shared with other traffic streams in order to simulate the internet cloud.
- OpenPegasus WBEM server version - release 2.2
- OpenWBEM WBEM server version - release 2.0.6

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor_id</td>
<td>GenuineIntel</td>
</tr>
<tr>
<td>CPU family</td>
<td>6</td>
</tr>
<tr>
<td>Model</td>
<td>7</td>
</tr>
<tr>
<td>Model name</td>
<td>Pentium III (Katmai)</td>
</tr>
<tr>
<td>Stepping</td>
<td>3</td>
</tr>
<tr>
<td>CPU MHz</td>
<td>550MHz (548.630MHz)</td>
</tr>
<tr>
<td>Cache size</td>
<td>512KB</td>
</tr>
<tr>
<td>Fdiv_bug</td>
<td>No</td>
</tr>
<tr>
<td>Hlt_bug</td>
<td>No</td>
</tr>
<tr>
<td>F00f_bug</td>
<td>No</td>
</tr>
<tr>
<td>Coma_bug</td>
<td>No</td>
</tr>
<tr>
<td>Fpu</td>
<td>yes</td>
</tr>
<tr>
<td>Fpu_exception</td>
<td>yes</td>
</tr>
<tr>
<td>Cpuid level</td>
<td>2</td>
</tr>
<tr>
<td>Wp</td>
<td>yes</td>
</tr>
<tr>
<td>Flags</td>
<td>Fpu vme do pse tsc msr pae mce cx8 sep mtrr pge mca cmov pat pse36 mmx fxsr sse</td>
</tr>
<tr>
<td>Bogomips</td>
<td>1094.45</td>
</tr>
</tbody>
</table>

Table 2: Processor Characteristics
4 Benchmark Feature Specification

The main goal of the benchmark tests is to evaluate the WBEM server performance with respect to the normal CIM operations, including latency, size of executables and shared libraries, size of the repository, CPU utilization and memory footprint.

The benchmark tests can be categorized into three groups:

1. CIM class, instance, property, association and qualifier operations on the repository,
2. CIM instance, association, property and method invocation operations on a provider,
3. process indication operations (the class indication and instance indication are not yet supported in most of the WBEM/CIM implementations), and
4. ExecQuery

Note that for each type of CIM operation, the benchmark tests are divided by groups depending on the varying factors of the tests. For example, the repository class operations can be benchmarked against three factors: number of classes in the repository, class inheritance and HTTP/HTTPS protocols. The benchmark tests are grouped according to the conditions so that only one factor is varying at one time while all others are fixed.

4.1 Repository Operations

The operations on the repository include class operations, instance operations, association operations and qualifier operations.

The repository implementations differ from each WBEM server implementations; therefore, the benchmark tests consider the overall performance of the server and the repository operations.

The following performance aspects are measured: latency of a request, size of the repository, CPU utilization, and memory footprint.
4.1.1 CIM Class Operations

The class operations include five types of operations:

- **GetClass**

  The GetClass operation retrieves the inheritance, properties, qualifiers and methods of the specified class. An exception is thrown if the class does not exist. This is possibly the only useful class operation to BCM.

- **EnumerateClassNames**

  The EnumerateClassNames operation returns the class names excluding any qualifiers or properties within the namespace specified in the request. This operation maybe used in some cases in BCM.

- **CreateClass**

  The CreateClass operation creates a new class in the namespace in the repository. If a class with the same class name and the version number already exists, then a CIM exception is thrown and the operation fails.

- **DeleteClass**

  The DeleteClass operation removes a class from the given namespace in the repository. If the class does not exist in the namespace, a CIM exception is thrown.

- **ModifyClass**

  The ModifyClass operation modifies the properties and/or qualifiers of a class. Note that some CIM implementations do NOT support ModifyClass feature yet.
Since the BCM box uses mostly GetClass and occasionally EnumerateClassNames, we will focus the benchmarks on these two operations.

The performance of the above operations is assumed to depend mainly on the number of classes within the specified namespace. Therefore, the benchmarks are tested against the number of classes in the namespace. The reasonable lower boundary for the benchmarks would be around 120 classes. This is because the new class is extended either from the CIM core models or from other standard CIM models which are extended from the core models, and thus the classes in the core models are always presented in a namespace. A reasonable upper boundary for number of classes in the repository is around 5000 classes. This number may vary depending on the complexity of the models.

The class operations are also tested using CIMXML over HTTP and CIMXML over HTTPS protocols, and the level of inheritance of a class.

The following tables capture the benchmark test groups for a class operation. The same sets of the tests are repeated for all the CIM class operations.

<table>
<thead>
<tr>
<th>Benchmark tests against scales</th>
<th>Lower boundary</th>
<th>Upper boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Classes in the repository</td>
<td>120 500 1000 2000 3000 5000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark tests against qualifiers</th>
<th>Lower boundary</th>
<th>Upper boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of qualifiers of a class</td>
<td>10 100 150 200 250 300</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark tests against protocols</th>
<th>CIMXML over HTTP/HTTPS given a fixed number of classes</th>
<th>HTTP</th>
<th>HTTPS</th>
</tr>
</thead>
</table>

Table 3: Benchmark test sets for CIM class operations

### 4.1.2 CIM Instance Operations

There are six types of instance operations. The instance operation performance would likely be affected by the number of classes in the repository, number of instances of a class, number of keys of an instance and key types.

- EnumerateInstanceNames
The EnumerateInstanceNames operation returns the CIM object path in a string format including the namespace, class name and key property values of all instances of a requested class.

- EnumerateInstances

The EnumerateInstances operation returns all the instances of a specified class. If there is a filter condition specified in the request, only the specified properties of the instances will be returned. The combination usage of LocalOnly and DeepInheritance allows instances of the superclass(es) and/or the subclass(es) to be returned. But the benchmark tests described in this document returns only the instances of the requested class.

- CreateInstance

The CreateInstance operation creates an instance of a specified class in the repository. A CIM exception is thrown if there is already an instance with the same key property value or values.

- GetInstance

The GetInstance operation returns an instance with the specified key property values. All the properties of the instance are returned. This operation is tested with IncludeQualifiers field is set to TRUE.

- DeleteInstance

The DeleteInstance operation deletes an instance from the repository. A CIM exception is thrown if the instance does not exist.

- ModifyInstance

The ModifyInstance operation changes the property or properties of an instance. Note that the key properties of an instance cannot be changed and a CIM exception is thrown if one attempts to change a key property.

The benchmark tests include only GetInstance, EnumerateInstanceNames and EnumerateInstances operation as they are the only ones being used in BCM.

The instance operations are tested using the following benchmark tests:

<table>
<thead>
<tr>
<th>Benchmark tests against scales</th>
<th>Lower boundary</th>
<th>Upper boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of instances of the requested class given a fixed number of classes in the</td>
<td>0</td>
<td>50</td>
</tr>
</tbody>
</table>

Nortel Networks
4.1.3 CIM Property Operations

The property operations will not be used in BCM, and the related benchmark tests are therefore omitted.

The property operations include two types of operations:

- **GetProperty**

  The GetProperty operation returns the requested property value of a particular instance.

- **SetProperty**

  The SetProperty operation modifies the requested property value of a particular instance.
4.1.4 CIM Association Operations

The association operations will not be used in BCM, and the related benchmark tests are therefore omitted.

The CIM association operations include four types of operations.

- **Associators**

The Associators operation returns a list of instances associated with a particular instance or classes associated with a particular class.

- **AssociatorNames**

The AssociatorNames operation returns a list of names of instances associated with a particular instance or class names associated with a particular class.

- **References**

The References operation returns a list of associations for a particular class or an instance.

- **ReferenceNames**

The ReferenceNames operation returns a list of names of associations for a particular class or an instance.

<table>
<thead>
<tr>
<th>Benchmark tests against scales</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Lower boundary</strong></td>
</tr>
<tr>
<td>Number of association instances with fixed number of classes in the repository</td>
</tr>
<tr>
<td>***</td>
</tr>
</tbody>
</table>

**Table 6: Benchmark test sets for CIM association operations**

4.1.5 CIM Qualifier Operations

The qualifier operations will not be used in BCM, and the related benchmark tests are therefore omitted.
The qualifier operations retrieve and set the qualifier of a class in the repository. This set of operations only applies to the repository.

- **EnumerateQualifiers**

The EnumerateQualifiers operation returns a list of qualifiers in the namespace.

- **GetQualifier**

The GetQualifier operation returns a qualifier from a given namespace.

- **SetQualifier**

The SetQualifier operation creates or modifies a particular qualifier in a given namespace.

- **DeleteQualifier**

The DeleteQualifier operation removes a qualifier from a namespace.

<table>
<thead>
<tr>
<th>Benchmark tests against scales</th>
<th>Lower boundary</th>
<th>Upper boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of Classes in the repository</strong></td>
<td>120</td>
<td>500</td>
</tr>
<tr>
<td><strong>Number of qualifiers of the requested class with fixed number of classes in the repository</strong></td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark tests against protocols</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMXML over HTTP/HTTPS given a fixed number of qualifiers on a class and a fixed number of classes</td>
<td>HTTP</td>
<td>HTTPS</td>
</tr>
</tbody>
</table>

Table 7: Benchmark test sets for CIM qualifier operations

4.1.6 **ExecQuery Operation**

The ExecQuery operation will not be used in BCM, and the related benchmark tests are therefore omitted.

The ExecQuery operation executes a database-style query on the classes and instances.
<table>
<thead>
<tr>
<th>Number of Classes in the repository</th>
<th>120</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of instances of the requested class given a fixed number of classes in the repository</td>
<td>0</td>
<td>10</td>
<td>50</td>
<td>200</td>
<td>500</td>
<td>1000</td>
</tr>
</tbody>
</table>

**Benchmark tests against protocols**

<table>
<thead>
<tr>
<th>CIMXML over HTTP/HTTPS given a fixed number of qualifiers on a class and a fixed number of classes</th>
<th>HTTP</th>
<th>HTTPS</th>
</tr>
</thead>
</table>

Table 8: Benchmark test sets for ExecQuery operation

### 4.2 Provider Operations

The operations on the provider are similar to those on the repository except the exclusion of class operations and qualifier operations. The normal provider operations include instance operations, association operations, method invocations and property operations. Note that the indication operation is a special type of provider operation, and it is covered in the indication benchmark test section.

The following performance aspects are measured: latency of a request, size of the repository, size of the shared libraries, CPU utilization, and memory footprint.

The provider benchmarks do not consider the specialty of the provider implementations, and thus the providers used in the tests are designed to perform NULL operation and immediately return with OK status.

Unless otherwise stated, all provider benchmarks are measured under the condition that the provider shared libraries are loaded prior to the tests and thus the library initial loading time is not considered.

![Figure 3: CIM Provider Operation Flow](image-url)
4.2.1 CIM Instance Operations

There are six types of instance operations. The instance operation performance may be affected by the number of classes in the repository, number of instances of a class, number of keys of an instance and key types.

- EnumerateInstanceNames
- EnumerateInstances - least used
- CreateInstance
- GetInstance
- DeleteInstance
- ModifyInstance

The instance operations on a provider are similar to those of the instance operations on the repository. The instance operations are tested using the following benchmark tests:

<table>
<thead>
<tr>
<th>Benchmark tests against scales</th>
<th>Lower boundary</th>
<th>Upper boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of instances of the requested class given a fixed number of classes in the repository</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>Number of keys on an instance given a fixed number of instances of a class</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark tests against key types</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Types</td>
<td>String</td>
</tr>
</tbody>
</table>

| Benchmark tests against protocols                      |
|---------------------------------------------------------|----------------|
| CIMXML over HTTP/HTTPS given a fixed number of classes and a fixed number of instances of a class | HTTP | HTTPS |

Table 9: Benchmark test sets for CIM instance operations

4.2.2 CIM Property Operations

The property operations on a provider are similar to those on the repository and include two types of operations:

- GetProperty
- SetProperty
| Number of instances of the requested class given a fixed number of classes | 0 | 10 | 50 | 200 | 500 | 1000 |
| Number of properties on an instance given a fixed number of instances of a class | 1 | 5 | 10 | 20 | 30 | 50 |

**Benchmark tests against property types**

| Property Types | String | Integer |

**Benchmark tests against protocols**

| CIMXML over HTTP/HTTPS given a fixed number of classes and a fixed number of instances of a class | HTTP | HTTPS |

Table 10: Benchmark test sets for CIM property operations

4.2.3 **CIM Association Operations**

The association operations on a provider are similar to those on the repository and include four types of operations:

- Associators
- AssociatorNames
- References
- ReferenceNames

| Number of association instances with fixed number of classes in the repository | Lower boundary | Upper boundary |
| Number of references defined in the association class | 0 | 500 | 1000 | 3000 | 4000 | 5000 |

| Numbe of references defined in the association class | 2 | 3 | 4 |

**Benchmark tests against scales**

| CIMXML over HTTP/HTTPS with fixed number of association instances and fixed number of classes | HTTP | HTTPS |

Table 11: Benchmark test sets for CIM association operations

According to the spec., an association class can have more than two references defined. However, the spec. does not provide descriptions on how enumerating association and reference works. In the case of a four-way association, it is unclear how the references are returned from a request of enumerating references. TODO
4.2.4 CIM Extrinsic Method Operations

- InvokeMethod

The extrinsic method invocation operations are tested using the following benchmark tests:

<table>
<thead>
<tr>
<th>Benchmark tests against scales</th>
<th>Lower boundary</th>
<th>Upper boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number input parameters in the method invocation</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Number of output parameters in the method invocation</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark tests against parameter types</th>
<th>Types of the input/output parameters</th>
<th>String</th>
<th>Integer</th>
</tr>
</thead>
</table>

| Benchmark tests against protocols | CIMXML over HTTP/HTTPS | HTTP | HTTPS |

Table 12: Benchmark test sets for extrinsic method operations

4.3 Indications

TODO

The indication tests measures the latency and throughput of indications. The latency does NOT consider the delay of the handler to the client because the handlers are highly customized by programmers.

4.3.1 Indication Test Setup in openPegasus

The indication tests are performed using the RT_Indication example provided by openPegasus. The setup is shown as in the figure below:

FIGURE: TODO

4.4 Client Library XML Encoding/Decoding

This benchmark test is to evaluate the WBEM client XML encoding and decoding performance using the client library provided by each WBEM implementation.
<table>
<thead>
<tr>
<th>Request Type</th>
<th>Simple operation</th>
<th>Multiple operations</th>
</tr>
</thead>
</table>

*Table 13: Benchmark test sets for extrinsic method operations*
5 Benchmark Test Instrumentation and Implementation

5.1 Instrumentation

There are two options of the instrumentation locations. Please see the diagram below

![Diagram showing WBEM Client and WBEM Server with Instrumentation Points]

**Figure 4: Instrumentation Points**

5.1.1 **Option 1: Instrumentation at Client Session End**

The instrumentation at the client session end implies that the benchmark is the duration of the period between the HTTP or HTTPS session is setup and the same session finishes. This instrumentation provides benchmarks of overall performance of the CIM operations over HTTP/HTTPS.

5.1.2 **Option 2: Instrumentation at Client Protocol End**

The instrumentation at the client protocol end implies that the benchmark is the duration of the period between the CIMXML request is put onto a HTTP or HTTPS protocol stack and the CIMXML response is received. This instrumentation provides benchmarks of CIMOM operation performance without consideration of the HTTP/HTTPS session setup procedures.

5.1.3 **Option 3: Instrumentation at CIMOM Server Protocol End**

The instrumentation at the CIMOM server protocol end implies that the benchmark is the duration of the period between a CIMXML request is received by WBEM server and a
CIMXML response is sent by the server. This operation considers only the server operation benchmarks without any consideration of the CIMXML messaging and HTTP/HTTPS session setup procedures. This instrumentation concerns mainly the performance of WBEM server repository interface and provider interface. To work with this option, one must insert the timestamps in the corresponding CIMOM server process.

### 5.1.4 Option 4: Instrumentation using Binary Interface

The instrumentation can also be done using the binary interface provided by some CIMOM implementations. The binary instrumentation point is not shown in the figure, but the measurement point would be on the backend of the client side.

### 5.2 Implementation

#### 5.2.1 wbemexec Utility

OpenPegasus provides a command line utility called "wbemexec". This tool reads the formatted CIMXML request from standard input, generates and sends a CIMXML request to the WBEM server, receives the CIMXML response and print the response to standard output. The wbemexec is a CIM client. Once a XML file containing a CIMXML request is created, the wbemexec utility performs the following steps. The steps in red refer to the timestamp insertion points for the benchmarks.

1. open the XML file
2. read the CIMXML request into a memory buffer
3. check for the XML syntax error
4. make a copy of CIMXML request
5. encapsulate XML into HTTP message
6. debug print the request message
   • start timer at instrumentation point 1
7. connect to the WBEM server
   • start timer at instrumentation point 2
8. send the CIMXML request and receive a CIMXML response
   • stop timer for both instrumentation points
9. put the response into a buffer
10. format the response and write to standard output

A Perl script is written to repeatedly invoke the wbemexec utility for TODO times and the benchmark result is the average time in a single invocation.

#### 5.2.2 Test Suite

- Introduction
The test suite is designed to allow the performance of an assortment of CIM servers to be benchmarked. Any number of tests and/or test scripts may be written to examine whatever performance aspects are desired. The scripts expect to have access to the wbemexec command-line program (a part of OpenPegasus), in particular, a version which has been modified to display how long the current request took to complete in the form

Operation took X.x/Y.y seconds

where X.x is the total number of seconds required (not including the time required to connect to the server) and Y.y is the total number of seconds required including the time required to connect to the server.

The test suite is started by typing

./TestSuite.pl <Testfile>

at the command line, where <Testfile> is a file containing the tests to be run (its format will be discussed later). The file "Script.txt" is included as a default file for the test suite.

- Directory Structure in CVS

```
bcm/
  cimBenchmarks/
   TestSuite.pl
   <TypeOfTest>Tests.txt
   openwbem/
    <TypeOfTest>/
     <TypeOfTest>Provider.cpp
     <TypeOfTest>Provider.h
     Makefile
    ...
   Pegasus/
    <TypeOfTest>/
     <TypeOfTest>Provider.cpp
     <TypeOfTest>Provider.h
     <TypeOfTest>ProviderMain.cpp
     Makefile
    ...
   repository/
    <TypeOfTest><Action>.pl
   ...
   provider/
    <TypeOfTest><Action>.pl
   ...
   xmlRequests/
   cimResponses/
   utils/
    XmlGen.pm
```

<TypeOfTest> can be "class", "instance", "property", "association" and "method" depending on what type of CIM operation is tested. <Action> refers to "setup", "tests" or "cleanup" of the specified type of CIM operation. For instance, the test scripts for the instance operation with provider would be in directory bcm/cimBenchmark/provider,
named as ClassSetup.pl, ClassTests.pl, and ClassCleanup.pl, and the provider code would be in directory bcm/cimBenchmark/openwbem-instance. The sequence of setup, tests and cleanup would allow one to produce a number of dummy classes, instances, or qualifiers in order to achieve a certain set of discrete test points, to perform the corresponding tests and finally to do the proper cleanup.

- Series format

The TestSuite.pl file invokes a series of tests by reading a script file. The script file contains information about which tests are to be performed and is formatted as below:

```
# lines starting with a '#' are comments
#
# Test Comment: <script> <command line options>
#
# Class setup options: <namespace> <dummyClassName>
# <total#OfClassesInRep> <serverHostname>
Example 1: ./repository/ClassSetup.pl root/widget DummyClass 5000 pkany216

# Class test options: <repetitions> <namespace> <serverHostname>
Example 2: ./repository/ClassTests.pl 100 root/widget pkany216

# Class cleanup options: <namespace> <dummyClassName>
# <total#OfClassesToClean> <serverHostname>
Example 3: ./repository/ClassCleanup.pl root/widget DummyClass 4020 pkany216
```

Note that the test scripts are invoked just as if they were being invoked from a shell. Any arguments after the test name are passed to the appropriate test script as command line options.

- Test Script Format

Each individual test script may be in any format, as long as it adheres to the following rules:

- It must be possible to invoke the script from the command line with any needed options
- The test script must return an integer value which, when shifted 8 bits to the left, indicates the number of errors which occurred when running the scripts (EG if a script was run for 1000 repetitions and 4 of those repetitions resulted in errors, the script would return (4 << 8). In the case of a perl script, it would simply return 4).

The current test scripts are written in perl, and may serve as a template for any future test scripts. They make use of the XmgGen perl module (which functions as an object) to create an xml file which will be passed to wbemexec, and they then receive and interpret the results.

- Error Checking
The TestSuite.pl will, before anything is run, go through the test scripts and verify that each file exists. Note that, as a result, a full path to any script file must be provided, since the test script will not use $PATH when checking for existence. During running, TestSuite.pl will report on how many errors were experienced by each test.

- Some Sample Scripts
These scripts are not currently used in the benchmark tests. They are template scripts for the new test scripts and located in the directory bcm/cimBenchmark/sampleScripts.

**EnumerateClassNames.pl**
*Usage:* EnumerateClassNames.pl <Repetitions> <Namespace>
*Purpose:* This script tests how long it takes to return a list of all classes in the current namespace

**CreateClass.pl**
*Usage:* CreateClass.pl <Repetitions> <Namespace> <ClassName>
*Purpose:* This script creates classes ClassName0 to ClassNameN in the supplied namespace where (N+1) is the number of repetitions.

**CreateSubClass.pl**
*Usage:* CreateSubClass.pl <Repetitions> <Namespace> <ClassName> <SuperClass>
*Purpose:* This script creates classes ClassName0 to ClassNameN (N+1 is the number of repetitions) in the supplied namespace, where all of the classes inherit from the supplied SuperClass. The SuperClass must, of course, exist.

**DeleteClass.pl**
*Usage:* DeleteClass.pl <Repetitions> <Namespace> <ClassName> <Numeric>
*Purpose:* This script deletes the specified class in the specified namespace. If Numeric is 1, it will delete the classes ClassName0 to ClassNameN (where N+1 is the number of repetitions). If Numeric is 0, it will delete ClassName.

**GetClass.pl**
*Usage:* GetClass.pl <Repetitions> <Namespace> <ClassName> <LocalOnly> <IncludeOrigin> <IncludeQualifiers> <Numeric>
*Purpose:* This script retrieves information about the specified class. If Numeric is 1, it will retrieve ClassName0 to ClassNameN (as per the other numeric scripts). If Numeric is 0, it will retrieve ClassName. LocalOnly, IncludeOrigin, and IncludeQualifiers are all IPARAM’s to be passed to the cimserver. They take either TRUE or FALSE, and their effect is as the effect of the equivalent IPARAM.

### 5.2.3 Test Providers

The tested providers include the instance provider, property provider, association provider, method provider and indication provider.
Usually, the actions inside the provider are highly customized depending on the features of the managed element. Thus, the provider processing time varies greatly from application to application. In order to obtain fair measurements of the provider interface of different WBEM implementations, we designed the providers under the following conditions.

**Test condition A:** The tests are to measure the CIMOM-provider interface latency, without considering the provider internal processing time. Therefore, the provided providers contain the minimum actions that are required to return a successful response or to throw an exception immediately upon an incoming request. The providers do NOT retain any instance/property/association data corresponding to the incoming request. The test results listed in this section are obtained under this test condition. The sample provider code can be found in the appendix D.

**Test condition B:** The providers are designed to perform minimum set of real actions. For example, the instance provider retains a local array of instances and return a valid instance upon a GetInstance request. The provider processing time is recorded and an average of the provider processing time for each method is calculated at the end of the test. The client-server roundtrip latency is measured using the total roundtrip latency minus the provider processing time. The provider may also take measurement of the memory usage and CPU usage. A BmDataClass is used to store the provider processing time as its properties. Note that the tests under this condition are listed as future work if effort allows.

For tests under condition B, a benchmark test class is defined to hold the latency data for the provider processing. For example, the mof file below defines such a class:

```mof
[Description ("Benchmark Data Class")]
class BmDataClass : CIM_System
{
    [Description ("For benchmark test purpose. If its value is set to 1, "
    "it marks the end of the benchmark tests, and the provider will print "
    "the elapsed time for internal provider data processing") ]
    uint32 TestProperty;

    [Description ("Elapsed time in usec for successful CreateInstance
    operations")]
    uint64 CiElapsedTime;

    [Description ("Elapsed time in usec for exceptions in CreateInstance
    operation")]
    uint64 ECiElapsedTime;

    [Description ("Elapsed time in usec for successful GetInstance
    operations")]
    uint64 GiElapsedTime;

    [Description ("Elapsed time in usec for exceptions in GetInstance
    operation")]
    uint64 EGiElapsedTime;

    [Description ("Elapsed time in usec for successful EnumerateInstanceNames
    operations")]
    uint64 EnElapsedTime;

    ...

}```
instance of BmDataClass
{
    CreationClassName="BmDataClass";
    Name="Benchmark";
    TestProperty="0";
    CiElapsedTime=0;
    ECiElapsedTime=0;
}
GiElapsedTime=0;
EGiElapsedTime=0;
EnElapsedTime=0;
ENiElapsedTime=0;
EiElapsedTime=0;
EEiElapsedTime=0;
MiElapsedTime=0;
EMiElapsedTime=0;
DiElapsedTime=0;
EDiElapsedTime=0;
CiCount=0;
ECiCount=0;
GliCount=0;
EGiCount=0;
EnCount=0;
EECount=0;
iCount=0;
EEiCount=0;
MiCount=0;
EMiCount=0;
DiCount=0;
EDiCount=0;
}

A sample provider method looks like the following:

CIMInstance GetInstance (...) {
    mark startTime;
    inst = getInstance(BmDataClass.
    CreateClassName="BmDataClass",Name="Benchmark");
    // do real work
    mark endTime;
    inst.setProperty("Property", "value");
    modifyInstance(inst);
    return;
}
6 Benchmark Results

6.1 HTTP Session Setup Time

From the results of all tests, the HTTP session setup time for openPegasus server with the server and client on the same machine is $2.35 \text{ msec} \pm 0.02\text{msec}$. The HTTP session setup time on different machine is $5.5\text{msec} \pm 2.0\text{msec}$. This value is not accurate because the two test machines are connected to a LAN which is not dedicated to the benchmark tests.

The openWBEM server has HTTP session setup time around $3.0 \text{ msec} \pm 0.1\text{msec}$ with the server and client on the same machine, and $4.0 \text{ msec} \pm 0.1\text{msec}$ with the server and client on different machines.

The values listed in the following sections are the roundtrip time including the session setup time.

6.2 HTTPS Session Setup Time

From the results of all tests, the HTTPS session setup time is TODO. The values listed in the following sections are the roundtrip time including the session setup time.

6.3 Repository Operations

6.3.1 Class Operations

The class operations are tested with the following settings:

IncludQualifiers = TRUE
IncludClassOrigins = TRUE
LocalOnly = FALSE
DeepInheritance = FALSE

All roundtrip latency values use msec as unit and they are measured including the HTTP/HTTPS session setup time.

Test 1: The test class is created with only the "Description" qualifier and with no super class. The protocol is HTTP. The WBEM client and server are on the same machine.

<table>
<thead>
<tr>
<th>Number of classes in the repository</th>
<th>123</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>4000</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetClass</td>
<td>12.61</td>
<td>13.16</td>
<td>14.09</td>
<td>15.57</td>
<td>16.99</td>
<td>18.46</td>
<td>19.87</td>
</tr>
<tr>
<td>EnumerateClassNames</td>
<td>19.05</td>
<td>37.71</td>
<td>73.57</td>
<td>124.84</td>
<td>179.94</td>
<td>230.96</td>
<td>296.48</td>
</tr>
<tr>
<td>CreateClass</td>
<td>13.82</td>
<td>13.87</td>
<td>14.08</td>
<td>14.31</td>
<td>14.47</td>
<td>14.77</td>
<td>15.03</td>
</tr>
<tr>
<td>DeleteClass</td>
<td>13.23</td>
<td>14.46</td>
<td>16.37</td>
<td>19.73</td>
<td>22.98</td>
<td>26.30</td>
<td>29.65</td>
</tr>
</tbody>
</table>
Test 2: The test class is created with CIM_System as super class and the qualifiers of the super class are included in the get operation. Including the qualifiers inherited from CIM_System, the test class has total 55 qualifiers. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openPegasus WBEM server.

<table>
<thead>
<tr>
<th>Number of classes in the repository</th>
<th>123</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>4000</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumerateClassNames</td>
<td>19.41</td>
<td>37.64</td>
<td>72.97</td>
<td>124.06</td>
<td>179.74</td>
<td>230.43</td>
<td>294.32</td>
</tr>
<tr>
<td>CreateClass</td>
<td>87.33</td>
<td>87.49</td>
<td>87.51</td>
<td>87.95</td>
<td>88.34</td>
<td>88.72</td>
<td>88.91</td>
</tr>
<tr>
<td>GetClass</td>
<td>49.51</td>
<td>49.33</td>
<td>50.30</td>
<td>51.81</td>
<td>53.17</td>
<td>54.62</td>
<td>57.63</td>
</tr>
<tr>
<td>DeleteClass</td>
<td>31.12</td>
<td>32.46</td>
<td>34.37</td>
<td>37.68</td>
<td>41.39</td>
<td>44.82</td>
<td>47.73</td>
</tr>
</tbody>
</table>

![Class Operations - 1](image)

Figure 5: openPegasus WBEM Server Class Operations against Number of Classes with Server/Client on the Same Machine

Test 3: There are 1000 classes in the repository, 980 of which are from standard CIM model and 20 of which are dummy ones. The test class is created as a subclass of a certain superclass. The roundtrip latency time is measured against the total number of properties and qualifiers of the tested class. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openPegasus WBEM server.
<table>
<thead>
<tr>
<th>Number of qualifiers in the class</th>
<th>24</th>
<th>54</th>
<th>99</th>
<th>158</th>
<th>198</th>
<th>267</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumerateClassNames</td>
<td>72.84</td>
<td>73.82</td>
<td>73.05</td>
<td>73.43</td>
<td>73.07</td>
<td>73.08</td>
</tr>
<tr>
<td>CreateClass</td>
<td>43.84</td>
<td>76.79</td>
<td>130.87</td>
<td>202.19</td>
<td>238.83</td>
<td>319.60</td>
</tr>
<tr>
<td>GetClass</td>
<td>30.01</td>
<td>48.64</td>
<td>76.81</td>
<td>113.51</td>
<td>132.12</td>
<td>178.34</td>
</tr>
<tr>
<td>DeleteClass</td>
<td>24.02</td>
<td>32.69</td>
<td>47.64</td>
<td>69.77</td>
<td>81.74</td>
<td>106.61</td>
</tr>
</tbody>
</table>

**Figure 6:** openPegasus WBEM Server Class Operations against Qualifiers with Server/Client on Separate Machines

**Test 4:** The test class is created with CIM_System as super class and the qualifiers of the super class are included in the get operation. Including the qualifiers inherited from CIM_System, the test class has total 55 qualifiers. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openPegasus WBEM server.

<table>
<thead>
<tr>
<th>Number of classes in the repository</th>
<th>123</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>4000</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumerateClassNames</td>
<td>17.38</td>
<td>36.70</td>
<td>82.93</td>
<td>129.92</td>
<td>186.32</td>
<td>237.98</td>
<td>305.34</td>
</tr>
<tr>
<td>CreateClass</td>
<td>77.61</td>
<td>75.48</td>
<td>76.76</td>
<td>76.14</td>
<td>78.06</td>
<td>76.89</td>
<td>76.56</td>
</tr>
<tr>
<td>GetClass</td>
<td>48.56</td>
<td>48.35</td>
<td>50.00</td>
<td>54.16</td>
<td>51.63</td>
<td>53.12</td>
<td>54.79</td>
</tr>
<tr>
<td>DeleteClass</td>
<td>29.15</td>
<td>29.08</td>
<td>35.21</td>
<td>35.49</td>
<td>39.51</td>
<td>43.82</td>
<td>45.58</td>
</tr>
</tbody>
</table>
Test 5: There are 1000 classes in the repository, 980 of which are from standard CIM model and 20 of which are dummy ones. The test class is created as a subclass of a certain superclass. The roundtrip latency time is measured against the total number of properties and qualifiers of the tested class. The protocol is HTTP. The WBEM client and server are on different machine. The server is openPegasus WBEM server.

<table>
<thead>
<tr>
<th>Number of qualifiers in the class</th>
<th>24</th>
<th>54</th>
<th>99</th>
<th>158</th>
<th>198</th>
<th>267</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumerateClassName</td>
<td>74.37</td>
<td>74.13</td>
<td>75.14</td>
<td>73.79</td>
<td>73.62</td>
<td>72.84</td>
</tr>
<tr>
<td>CreateClass</td>
<td>43.81</td>
<td>75.74</td>
<td>129.51</td>
<td>200.84</td>
<td>239.12</td>
<td>332.00</td>
</tr>
<tr>
<td>GetClass</td>
<td>33.08</td>
<td>47.54</td>
<td>78.19</td>
<td>117.06</td>
<td>137.28</td>
<td>181.12</td>
</tr>
<tr>
<td>DeleteClass</td>
<td>26.31</td>
<td>32.00</td>
<td>49.92</td>
<td>71.44</td>
<td>85.11</td>
<td>110.05</td>
</tr>
</tbody>
</table>
**Figure 8: openPegasus WBEM Server Class Operations against Qualifiers with the Client/Server on Separate Machines**

**Test 6:** The test class is created with CIM\_System as super class and the qualifiers of the super class are included in the get operation. Including the qualifiers inherited from CIM\_System, the test class has total 55 qualifiers. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openWBEM server.

<table>
<thead>
<tr>
<th>Number of classes in the repository</th>
<th>123</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>4000</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumerateClassNames</td>
<td>436.77</td>
<td>751.50</td>
<td>8751.26</td>
<td>9478.80</td>
<td>10194.40</td>
<td>10979.30</td>
<td>11715.07</td>
</tr>
<tr>
<td>CreateClass</td>
<td>34.64</td>
<td>35.05</td>
<td>44.30</td>
<td>44.33</td>
<td>44.19</td>
<td>44.30</td>
<td>44.40</td>
</tr>
<tr>
<td>GetClass</td>
<td>64.05</td>
<td>64.05</td>
<td>66.03</td>
<td>65.41</td>
<td>65.69</td>
<td>65.20</td>
<td>65.08</td>
</tr>
<tr>
<td>DeleteClass</td>
<td>21.33</td>
<td>21.35</td>
<td>30.09</td>
<td>30.29</td>
<td>30.27</td>
<td>30.25</td>
<td>30.51</td>
</tr>
</tbody>
</table>
Figure 9: openWBEM Server Class Operations against Number of Classes with Client/Server on the Same Machine
**Test 7:** There are 1000 classes in the repository, 980 of which are from standard CIM model and 20 of which are dummy ones. The test class is created as a subclass of a certain superclass. The roundtrip latency time is measured against the total number of properties and qualifiers of the tested class. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openWBEM server.

<table>
<thead>
<tr>
<th>Number of qualifiers in the class</th>
<th>24</th>
<th>37</th>
<th>55</th>
<th>105</th>
<th>163</th>
<th>203</th>
<th>272</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumerateClassNames</td>
<td>8731.16</td>
<td>8732.16</td>
<td>8748.58</td>
<td>8730.75</td>
<td>8733.46</td>
<td>8726.25</td>
<td>8719.29</td>
</tr>
<tr>
<td>CreateClass</td>
<td>35.16</td>
<td>40.60</td>
<td>44.30</td>
<td>56.88</td>
<td>111.59</td>
<td>165.61</td>
<td>303.32</td>
</tr>
<tr>
<td>GetClass</td>
<td>22.90</td>
<td>30.11</td>
<td>64.80</td>
<td>51.33</td>
<td>81.22</td>
<td>102.85</td>
<td>127.75</td>
</tr>
<tr>
<td>DeleteClass</td>
<td>29.78</td>
<td>29.77</td>
<td>30.09</td>
<td>29.85</td>
<td>30.13</td>
<td>30.32</td>
<td>30.35</td>
</tr>
</tbody>
</table>

**Figure 10:** openWBEM Server Class Operations against Number of Qualifiers with Client/Server on the Same Machine

**Test 8:** The test class is created with CIM_System as super class and the qualifiers of the super class are included in the get operation. Including the qualifiers inherited from CIM_System, the test class has total 55 qualifiers. The protocol is HTTP. The WBEM client and server are on different machine. The server is openWBEM server.

<table>
<thead>
<tr>
<th>Number of classes in the repository</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>4000</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumerateClassNames</td>
<td>8642.33</td>
<td>9372.08</td>
<td>10113.39</td>
<td>10852.44</td>
<td>11600.98</td>
</tr>
<tr>
<td>CreateClass</td>
<td>40.02</td>
<td>40.42</td>
<td>39.96</td>
<td>40.20</td>
<td>40.35</td>
</tr>
<tr>
<td>GetClass</td>
<td>30.65</td>
<td>30.97</td>
<td>30.82</td>
<td>30.79</td>
<td>30.76</td>
</tr>
<tr>
<td>DeleteClass</td>
<td>27.71</td>
<td>27.75</td>
<td>26.34</td>
<td>27.30</td>
<td>27.53</td>
</tr>
</tbody>
</table>
Figure 11: openWBEM Server Class Operations against Number of Classes with Client/Server on the Separate Machines
**Test 9:** There are 1000 classes in the repository, 980 of which are from standard CIM model and 20 of which are dummy ones. The test class is created as a subclass of a certain superclass. The roundtrip latency time is measured against the total number of properties and qualifiers of the tested class. The protocol is HTTP. The WBEM client and server are on different machines. The server is openWBEM server.

<table>
<thead>
<tr>
<th>Number of qualifiers in the class</th>
<th>24</th>
<th>55</th>
<th>104</th>
<th>163</th>
<th>203</th>
<th>272</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumeratingClassNames</td>
<td>8644.62</td>
<td>8641.06</td>
<td>8640.81</td>
<td>8638.56</td>
<td>8634.61</td>
<td>8628.82</td>
</tr>
<tr>
<td>CreateClass</td>
<td>31.87</td>
<td>40.11</td>
<td>53.62</td>
<td>109.84</td>
<td>163.37</td>
<td>300.52</td>
</tr>
<tr>
<td>GetClass</td>
<td>21.32</td>
<td>31.34</td>
<td>48.87</td>
<td>65.41</td>
<td>75.80</td>
<td>97.77</td>
</tr>
<tr>
<td>DeleteClass</td>
<td>26.32</td>
<td>27.49</td>
<td>28.35</td>
<td>28.40</td>
<td>27.76</td>
<td>28.80</td>
</tr>
</tbody>
</table>

**Figure 12:** openWBEM Server Class Operations against Number of Qualifiers with Client/Server on the Separate Machines

**Observations:**

For the openPegasus WBEM server, all class operations behave linearly against some factors.

EnumeratingClassNames: This operation depends solely on the number of classes in the repository. It does NOT depend on class inheritance or the number of qualifiers.
GetClass: Based on the test environment settings, the GetClass operation depends greatly on the number of qualifiers, because the GetClass operation returns the qualifiers of the class itself and its superclass. For example, to get a class which is a subclass of CIM_System takes nearly 37msec longer than that to get a class without any superclass or any qualifier. The number of classes in the repository has very little effect on the latency.

CreateClass: This operation depends greatly on the number of qualifiers. For example, to create a class which is a subclass of CIM_System takes nearly 73msec longer than that to create a class without any superclass or any qualifier. The number of classes in the repository has very little effect on the latency.

DeleteClass: This operation depends greatly on the number of qualifiers and class inheritance. For example, to delete a class which is a subclass of CIM_System takes nearly 28msec longer than that to delete a class without any superclass or any qualifier. The number of classes in the repository has very little effect on the latency.

For the openWBEM server, the behaviours of the class operations are similar to those of the openPegasus, but there are a few noticeable differences.

The EnumerateClassNames latency increases linearly against number of classes in the repository and does not depend on any other factors. The EnumerateClassNames operation has significant longer latency for openWBEM server. This maybe related to the repository implementation. Unlike the OpenPegasus who uses the plain XML files, the openWBEM uses the binary database and the search process in all index (*.ndx) and data (*.dat) files takes a longer period.

The number of classes in the repository does not affect the latency of CreateClass and GetClass operations. Both latencies increase against the number of qualifiers in the specified class. While the GetClass latency increases linearly, the CreateClass latency seems to increase more significantly when the number of qualifiers in the class increases.
Comparisons:

Figure 13: Class Operation Comparison 1 - Client/Server on the Same Machine
Figure 14: Class Operation Comparison 2 - Client/Server on the Separate Machines
Class Operations Comparison 3 - Client/Server on Same Machine

Figure 15: Class Operation Comparison 1 - Client/Server on the Same Machine
Figure 16: Class Operation Comparison 2 - Client/Server on the Separate Machines
6.3.2 Instance Operations

The instance operations are tested with the following settings:

IncludeQualifiers = FALSE  
IncludeClassOrigins = FALSE  
LocalOnly = FALSE  
DeepInheritance = FALSE

All roundtrip latency values use msec as unit and they are measured including the HTTP/HTTPS session setup time.

Test 1: Instance operations against number of keys of the instance, and the key type is string. The test class is a subclass of CIM_EnabledLogicalElement. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openWBEM server.

<table>
<thead>
<tr>
<th>Number of keys in the instance</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateInstance</td>
<td>38.30</td>
<td>40.68</td>
<td>44.76</td>
<td>48.28</td>
<td>53.01</td>
<td>58.36</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>170.13</td>
<td>247.56</td>
<td>346.66</td>
<td>397.36</td>
<td>523.16</td>
<td>617.81</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>815.79</td>
<td>942.16</td>
<td>1205.31</td>
<td>1457.01</td>
<td>1786.34</td>
<td>2081.13</td>
</tr>
<tr>
<td>GetInstance</td>
<td>36.66</td>
<td>36.61</td>
<td>38.95</td>
<td>41.62</td>
<td>44.97</td>
<td>47.63</td>
</tr>
<tr>
<td>ModifyInstance</td>
<td>48.75</td>
<td>47.74</td>
<td>53.93</td>
<td>59.02</td>
<td>65.39</td>
<td>72.10</td>
</tr>
<tr>
<td>DeleteInstance</td>
<td>42.43</td>
<td>42.56</td>
<td>46.58</td>
<td>50.88</td>
<td>55.22</td>
<td>60.40</td>
</tr>
</tbody>
</table>

Test 2: Instance operations against number of keys of the instance, and the key type is uint32. The test class is a subclass of CIM_EnabledLogicalElement. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openWBEM server.

<table>
<thead>
<tr>
<th>Number of keys in the instance</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateInstance</td>
<td>38.45</td>
<td>39.85</td>
<td>42.99</td>
<td>45.86</td>
<td>49.78</td>
<td>54.09</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>169.52</td>
<td>250.88</td>
<td>306.81</td>
<td>404.19</td>
<td>523.91</td>
<td>620.24</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>809.04</td>
<td>935.14</td>
<td>1199.18</td>
<td>1476.47</td>
<td>1749.59</td>
<td>2093.73</td>
</tr>
<tr>
<td>GetInstance</td>
<td>35.52</td>
<td>37.61</td>
<td>41.83</td>
<td>45.14</td>
<td>49.79</td>
<td>54.40</td>
</tr>
<tr>
<td>ModifyInstance</td>
<td>46.34</td>
<td>50.74</td>
<td>53.63</td>
<td>64.61</td>
<td>73.52</td>
<td>81.57</td>
</tr>
<tr>
<td>DeleteInstance</td>
<td>31.89</td>
<td>44.64</td>
<td>50.63</td>
<td>56.35</td>
<td>63.60</td>
<td>69.91</td>
</tr>
</tbody>
</table>

Test 3: Instance operations against number of keys of the instance, and the key type is string. The test class is a subclass of CIM_EnabledLogicalElement. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openPegasus server.
### Test 4: Instance operations against number of keys of the instance, and the key type is `uint32`. The test class is a subclass of `CIM_EnabledLogicalElement`. The protocol is `HTTP`. The WBEM client and server are on the same machine. The server is openPegasus server.

<table>
<thead>
<tr>
<th>Number of keys in the instance</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateInstance</td>
<td>97.36</td>
<td>104.54</td>
<td>135.83</td>
<td>151.29</td>
<td>183.37</td>
<td>197.73</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>76.05</td>
<td>90.07</td>
<td>147.78</td>
<td>176.76</td>
<td>237.86</td>
<td>270.46</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>3929.09</td>
<td>3987.60</td>
<td>4148.43</td>
<td>4287.81</td>
<td>4464.81</td>
<td>4598.71</td>
</tr>
<tr>
<td>GetInstance</td>
<td>64.20</td>
<td>61.37</td>
<td>72.34</td>
<td>67.71</td>
<td>78.68</td>
<td>71.44</td>
</tr>
<tr>
<td>ModifyInstance</td>
<td>100.23</td>
<td>101.79</td>
<td>123.05</td>
<td>128.71</td>
<td>149.99</td>
<td>154.41</td>
</tr>
<tr>
<td>DeleteInstance</td>
<td>50.55</td>
<td>61.08</td>
<td>82.19</td>
<td>103.09</td>
<td>124.99</td>
<td>147.79</td>
</tr>
</tbody>
</table>

![Instance Operations Comparison - Client/Server on Same Machine](image_url)

**Figure 17: Instance Operation Comparison 1 - Client/Server on the Same Machines**
Figure 18: Instance Operation Comparison 2 - Client/Server on the Same Machines

Figure 19: Instance Operation Comparison 3 - Client/Server on the Same Machine
Test 5: Instance operations against number of instances of the requested class, and the key type is string. The test class is a subclass of CIM_EnabledLogicalElement and the instance has four keys. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openWBEM server.

<table>
<thead>
<tr>
<th>Number of keys in the instance</th>
<th>0</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateInstance</td>
<td>35.21</td>
<td>35.16</td>
<td>35.62</td>
<td>35.54</td>
<td>36.25</td>
<td>37.14</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>13.37</td>
<td>189.07</td>
<td>336.84</td>
<td>653.33</td>
<td>1460.91</td>
<td>2935.62</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>23.56</td>
<td>628.07</td>
<td>1191.72</td>
<td>2315.10</td>
<td>5790.18</td>
<td>11537.80</td>
</tr>
<tr>
<td>GetInstance</td>
<td>30.78</td>
<td>31.47</td>
<td>31.45</td>
<td>31.56</td>
<td>31.50</td>
<td>31.55</td>
</tr>
<tr>
<td>ModifyInstance</td>
<td>45.00</td>
<td>45.14</td>
<td>45.04</td>
<td>45.52</td>
<td>46.09</td>
<td>47.02</td>
</tr>
<tr>
<td>DeleteInstance</td>
<td>38.34</td>
<td>38.74</td>
<td>38.18</td>
<td>38.22</td>
<td>38.69</td>
<td>39.57</td>
</tr>
</tbody>
</table>

Test 6: Instance operations against number of instances of the requested class, and the key type is uint32. The test class is a subclass of CIM_EnabledLogicalElement and the instance has four keys. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openWBEM server.

<table>
<thead>
<tr>
<th>Number of keys in the instance</th>
<th>0</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateInstance</td>
<td>33.07</td>
<td>33.15</td>
<td>33.10</td>
<td>33.15</td>
<td>33.30</td>
<td>33.18</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>13.41</td>
<td>194.91</td>
<td>302.20</td>
<td>614.66</td>
<td>1485.66</td>
<td>1931.26</td>
</tr>
</tbody>
</table>
Test 7: Instance operations against number of instances of the requested class, and the key type is string. The test class is a subclass of CIM EnabledLogicalElement and the instance has four keys. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openPegasus server.

<table>
<thead>
<tr>
<th>Number of keys in the instance</th>
<th>0</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateInstance</td>
<td>81.28</td>
<td>107.56</td>
<td>135.76</td>
<td>197.70</td>
<td>380.05</td>
<td>678.05</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>46.71</td>
<td>96.27</td>
<td>151.94</td>
<td>255.84</td>
<td>584.72</td>
<td>1137.98</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>83.75</td>
<td>2119.74</td>
<td>4161.97</td>
<td>8296.74</td>
<td>20672.30</td>
<td>41263.00</td>
</tr>
<tr>
<td>GetInstance</td>
<td>72.60</td>
<td>72.97</td>
<td>72.30</td>
<td>72.43</td>
<td>72.60</td>
<td>116.22</td>
</tr>
<tr>
<td>ModifyInstance</td>
<td>99.19</td>
<td>111.03</td>
<td>123.16</td>
<td>148.83</td>
<td>226.07</td>
<td>354.00</td>
</tr>
<tr>
<td>DeleteInstance</td>
<td>24.71</td>
<td>53.86</td>
<td>82.02</td>
<td>143.99</td>
<td>325.58</td>
<td>625.30</td>
</tr>
</tbody>
</table>

Test 8: Instance operations against number of instances of the requested class, and the key type is uint32. The test class is a subclass of CIM EnabledLogicalElement and the instance has four keys. The protocol is HTTP. The WBEM client and server are on the same machine. The server is openPegasus server.

<table>
<thead>
<tr>
<th>Number of keys in the instance</th>
<th>0</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateInstance</td>
<td>73.64</td>
<td>96.05</td>
<td>116.48</td>
<td>162.25</td>
<td>295.74</td>
<td>515.34</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>37.59</td>
<td>85.75</td>
<td>133.21</td>
<td>228.94</td>
<td>526.02</td>
<td>1024.74</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>77.18</td>
<td>2098.55</td>
<td>4125.32</td>
<td>8172.65</td>
<td>20510.60</td>
<td>40889.00</td>
</tr>
<tr>
<td>GetInstance</td>
<td>69.15</td>
<td>70.04</td>
<td>69.25</td>
<td>69.28</td>
<td>69.36</td>
<td>69.35</td>
</tr>
<tr>
<td>ModifyInstance</td>
<td>94.65</td>
<td>104.54</td>
<td>112.69</td>
<td>131.26</td>
<td>186.75</td>
<td>279.53</td>
</tr>
<tr>
<td>DeleteInstance</td>
<td>24.04</td>
<td>45.39</td>
<td>65.76</td>
<td>111.21</td>
<td>244.79</td>
<td>464.47</td>
</tr>
</tbody>
</table>
Figure 21: Instance Operation Comparison 5 - Client/Server on the Same Machine

Figure 22: Instance Operation Comparison 6 - Client/Server on the Same Machine
Figure 23: Instance Operation Comparison 7 - Client/Server on the Same Machine

Figure 24: Instance Operation Comparison 8 - Client/Server on the Same Machine
6.4 Provider Operations

The test results listed in this section are obtained under test condition A described in section 5.2.3. All latency units are in msec.

6.4.1 Instance Operations

6.4.1.1 Tests against States in openWBEM

**Test 1:** The provider returns a success response for all operations. The instance class is a subclass of CIM EnabledManagedElement and it has four keys whose type is string. The WBEM server is openWBEM server.

The minimum actions required in the methods: EnumerateInstanceNames and EnumerateInstances return empty list; the CreateInstance returns an CIMObjectPath from the incoming instance; the GetInstance returns an instance constructed by the given className; ModifyInstance and DeleteInstance immediately return with no action taken.

<table>
<thead>
<tr>
<th>Test state</th>
<th>The initial state of the WBEM server</th>
<th>The state of the first-time provider invocation after the WBEM server has been already started for a while</th>
<th>The state after a long provider idle period if a provider timeout is defined</th>
<th>The normal operation state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library conditions</td>
<td></td>
<td>The provider library is NOT yet loaded, but the request handler library is already loaded</td>
<td>The provider library is previously loaded and unloaded. It is to be RE-loaded, while the request handler library is already loaded</td>
<td>Both provider library and the request library are already loaded</td>
</tr>
<tr>
<td>CreateInstance</td>
<td>103.85</td>
<td>43.55</td>
<td>35.23</td>
<td>29.34</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>53.06</td>
<td>24.86</td>
<td>18.50</td>
<td>12.64</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>54.74</td>
<td>26.53</td>
<td>19.97</td>
<td>13.85</td>
</tr>
<tr>
<td>GetInstance</td>
<td>102.22</td>
<td>40.90</td>
<td>34.53</td>
<td>28.04</td>
</tr>
<tr>
<td>ModifyInstance</td>
<td>105.89</td>
<td>45.05</td>
<td>36.56</td>
<td>30.80</td>
</tr>
<tr>
<td>DeleteInstance</td>
<td>101.01</td>
<td>40.89</td>
<td>32.50</td>
<td>26.73</td>
</tr>
</tbody>
</table>

**Test 2:** The provider returns a success response for all operations. The instance class is a subclass of CIM EnabledManagedElement and it has four keys whose type is uint32. The WBEM server is openWBEM server.

The minimum actions required in the methods: EnumerateInstanceNames and EnumerateInstances return empty list; the CreateInstance returns an CIMObjectPath from
the incoming instance; the GetInstance returns an instance constructed by the given
className; ModifyInstance and DeleteInstance immediately return with no action taken.

<table>
<thead>
<tr>
<th>Test state</th>
<th>The initial state of the WBEM server</th>
<th>The state of the first-time provider invocation after the WBEM server has been already started for a while</th>
<th>The state after a long provider idle period if a provider timeout is defined</th>
<th>The normal operation state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library conditions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreateInstance</td>
<td>101.75</td>
<td>41.21</td>
<td>33.51</td>
<td>28.12</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>52.73</td>
<td>24.82</td>
<td>18.48</td>
<td>12.64</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>54.46</td>
<td>26.63</td>
<td>19.43</td>
<td>13.82</td>
</tr>
<tr>
<td>GetInstance</td>
<td>103.81</td>
<td>42.53</td>
<td>35.72</td>
<td>29.63</td>
</tr>
<tr>
<td>ModifyInstance</td>
<td>107.67</td>
<td>47.21</td>
<td>37.13</td>
<td>32.04</td>
</tr>
<tr>
<td>DeleteInstance</td>
<td>103.40</td>
<td>42.30</td>
<td>34.39</td>
<td>28.39</td>
</tr>
</tbody>
</table>

**Test 3:** The provider returns an exception for all operations. The instance class is a
subclass of **CIM_ENABLEDELEMENT** and it has four keys whose type is
string. The WBEM server is openWBEM server.

The minimum actions required in the methods: all operations performs no action except
to throw an FAILED exception with a message.

<table>
<thead>
<tr>
<th>Test state</th>
<th>The initial state of the WBEM server</th>
<th>The state of the first-time provider invocation after the WBEM server has been already started for a while</th>
<th>The state after a long provider idle period if a provider timeout is defined</th>
<th>The normal operation state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library conditions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreateInstance</td>
<td>61.34</td>
<td>33.39</td>
<td>26.91</td>
<td>20.42</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>51.69</td>
<td>23.77</td>
<td>17.33</td>
<td>11.20</td>
</tr>
</tbody>
</table>
Test 4: The provider returns an exception for all operations. The instance class is a subclass of CIM_EnabledManagedElement and it has four keys whose type is uint32. The WBEM server is openWBEM server.

The minimum actions required in the methods: all operations performs no action except to throw an FAILED exception with a message.

<table>
<thead>
<tr>
<th>Test state</th>
<th>The initial state of the WBEM server</th>
<th>The state of the first-time provider invocation after the WBEM server has been already started for a while</th>
<th>The state after a long provider idle period if a provider timeout is defined</th>
<th>The normal operation state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library conditions</td>
<td>The provider library and the request handler library are NOT loaded yet</td>
<td>The provider library is NOT yet loaded, but the request handler library is already loaded</td>
<td>The provider library is previously loaded and unloaded. It is to be RE-loaded, while the request handler library is already loaded</td>
<td>Both provider library and the request library are already loaded</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>CreateInstance</th>
<th>EnumerateInstanceNames</th>
<th>EnumerateInstances</th>
<th>GetInstance</th>
<th>ModifyInstance</th>
<th>DeleteInstance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (in seconds)</td>
<td>59.97</td>
<td>31.59</td>
<td>25.10</td>
<td>18.62</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>51.72</td>
<td>23.75</td>
<td>18.34</td>
<td>11.45</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>53.83</td>
<td>25.74</td>
<td>19.18</td>
<td>13.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>57.75</td>
<td>28.96</td>
<td>22.45</td>
<td>16.19</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>58.01</td>
<td>29.59</td>
<td>23.02</td>
<td>16.43</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>56.23</td>
<td>27.91</td>
<td>21.28</td>
<td>14.76</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.4.1.2 Tests against Number of Keys in openWBEM

Test 1: The provider returns a success response for all operations. The instance class is a subclass of CIM_EnabledManagedElement and it has four keys whose type is string. The WBEM server is openWBEM server.

The minimum actions required in the methods: EnumerateInstanceNames and EnumerateInstances return empty list; the CreateInstance returns an CIMObjectPath from the incoming instance; the GetInstance returns an instance constructed by the given className; ModifyInstance and DeleteInstance immediately return with no action taken.
Test 2: The provider returns an exception for all operations. The instance class is a subclass of CIM_EnabledManagedElement and it has four keys whose type is string. The WBEM server is openWBEM server.

The minimum actions required in the methods: all operations performs no action except to throw an FAILED exception with a message.
Test 3: The provider returns a success response for all operations. The instance class is a subclass of CIM_EnabledManagedElement and it has four keys whose type is uint32. The WBEM server is openWBEM server.

The minimum actions required in the methods: EnumerateInstanceNames and EnumerateInstances return empty list; the CreateInstance returns an CIMObjectPath from the incoming instance; the GetInstance returns an instance constructed by the given className; ModifyInstance and DeleteInstance immediately return with no action taken.

<table>
<thead>
<tr>
<th>Number of keys in the instance</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateInstance</td>
<td>24.33</td>
<td>25.71</td>
<td>28.12</td>
<td>30.72</td>
<td>33.46</td>
<td>37.79</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>11.26</td>
<td>11.65</td>
<td>12.64</td>
<td>13.18</td>
<td>13.73</td>
<td>14.47</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>13.05</td>
<td>13.27</td>
<td>13.82</td>
<td>14.69</td>
<td>15.53</td>
<td>16.35</td>
</tr>
<tr>
<td>GetInstance</td>
<td>26.47</td>
<td>27.25</td>
<td>29.63</td>
<td>31.77</td>
<td>33.83</td>
<td>36.05</td>
</tr>
<tr>
<td>ModifyInstance</td>
<td>28.60</td>
<td>29.80</td>
<td>32.04</td>
<td>34.88</td>
<td>36.85</td>
<td>39.19</td>
</tr>
<tr>
<td>DeleteInstance</td>
<td>24.92</td>
<td>26.04</td>
<td>28.39</td>
<td>31.01</td>
<td>33.33</td>
<td>35.91</td>
</tr>
</tbody>
</table>
**Test 4:** The provider returns an exception for all operations. The instance class is a subclass of `CIM_EnabledManagedElement` and it has four keys whose type is `uint32`. The WBEM server is `openWBEM` server.

The minimum actions required in the methods: all operations performs no action except to throw an FAILED exception with a message.

<table>
<thead>
<tr>
<th>Number of keys in the instance</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateInstance</td>
<td>16.05</td>
<td>17.02</td>
<td>18.62</td>
<td>21.39</td>
<td>23.68</td>
<td>26.48</td>
</tr>
<tr>
<td>EnumerateInstanceNames</td>
<td>11.46</td>
<td>11.46</td>
<td>11.45</td>
<td>11.28</td>
<td>11.08</td>
<td>11.40</td>
</tr>
<tr>
<td>EnumerateInstances</td>
<td>13.10</td>
<td>12.95</td>
<td>13.09</td>
<td>13.03</td>
<td>13.16</td>
<td>13.16</td>
</tr>
<tr>
<td>GetInstance</td>
<td>14.37</td>
<td>14.66</td>
<td>16.19</td>
<td>17.71</td>
<td>18.54</td>
<td>20.00</td>
</tr>
<tr>
<td>ModifyInstance</td>
<td>14.42</td>
<td>15.84</td>
<td>16.43</td>
<td>18.10</td>
<td>18.86</td>
<td>20.14</td>
</tr>
<tr>
<td>DeleteInstance</td>
<td>13.02</td>
<td>13.50</td>
<td>14.76</td>
<td>16.54</td>
<td>17.61</td>
<td>19.20</td>
</tr>
</tbody>
</table>

**Figure 26:** `openWBEM` Provider Instance Operation - Key Type is `Uint32`

Under the test condition 1, the openPegasus server has the following results: TODO
6.4.1.3 Tests in openPegasus

TODO: In Pegasus release 2.2, the providers are automatically unloaded after a very short random timeout. This is a known bug and is expected to be fixed in release 2.3. Thus, we are pending the bug fix and the provider tests for openPegasus are NOT performed.

Notes (Nov. 27, 2003): Release 2.3 has been downloaded and installed. The initial tests proved that the providers are NOT unloaded after a short random period. However, extensive tests are required to verify the bug fix.

6.4.2 Property Operations

TODO

6.4.3 Association Operations

TODO

6.4.4 Indication Operations

TODO
7 Future Work

The benchmark tests described in previous sections are performed under the condition of a single client connected to the WBEM server. The future work would include the benchmark tests under the condition of multiple clients connected to a WBEM server simultaneously. In this case, the throughput (how many messages per second) is measured and the result is compared with that of a single client.
8 References

9 Appendix A - Sample XML Messages

Some sample XML requests for operations:

- **EnumerateClassNames**

The EnumerateClassNames operation returns the class names within the namespace specified in the request. A sample request in CIMXML would be as follows:

```xml
<?xml version="1.0" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="7873" PROTOCOLVERSION="1.0">
    <SIMPLEREQ>
      <IMETHODCALL NAME="EnumerateClassNames">
        <LOCALNAMESPACEPATH>
          <NAMESPACE NAME="root"/>
          <NAMESPACE NAME="widget"/>
        </LOCALNAMESPACEPATH>
        <IPARAMVALUE NAME="DeepInheritance">
          <VALUE>TRUE</VALUE>
        </IPARAMVALUE>
      </IMETHODCALL>
    </SIMPLEREQ>
  </MESSAGE>
</CIM>
```

- **CreateClass**

The CreateClass operation creates a new class in the namespace in the repository. A sample request in CIMXML would be as follows:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="7" PROTOCOLVERSION="1.0">
    <SIMPLEREQ>
      <IMETHODCALL NAME="CreateClass">
        <LOCALNAMESPACEPATH>
          <NAMESPACE NAME="root"/>
          <NAMESPACE NAME="widget"/>
        </LOCALNAMESPACEPATH>
        <IPARAMVALUE NAME="NewClass">
          <CLASS NAME="BmTestClass" SUPERCLASS="CIM_System">
            <QUALIFIER NAME="Version" TYPE="string" >
              <VALUE>4.5.7</VALUE>
            </QUALIFIER>
            <QUALIFIER NAME="Description" TYPE="string" >
              <VALUE>Widget is a class which defines the Widget developed for use within all left-handed thronge units. Note that it should not be used as a superclass of any device which has been developed to support brillig dews.</VALUE>
            </QUALIFIER>
            <PROPERTY NAME="colour" TYPE="uint8" >
              <QUALIFIER NAME="Write" TYPE="boolean" >
                <VALUE>true</VALUE>
              </QUALIFIER>
              <QUALIFIER NAME="Description" TYPE="string" >
                <VALUE>Colour of the external box</VALUE>
              </QUALIFIER>
            </PROPERTY>
            <QUALIFIER NAME="ValueMap" TYPE="string" >
              <VALUE.ARRAY>
```

Nortel Networks
GetClass

The GetClass operation retrieves the inheritance, properties, qualifiers and methods of the specified class. A sample CIMXML request is as follows:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
<MESSAGE ID="1001" PROTOCOLVERSION="1.0">
<SIMPLEREQ>
<IMETHODCALL NAME="GetClass">
  <LOCALNAMESPACEPATH>
    <NAMESPACE NAME="root"/>
    <NAMESPACE NAME="widget"/>
  </LOCALNAMESPACEPATH>
  <IPARAMVALUE NAME="ClassName"/>
  <CLASSNAME NAME="BmTestClass"/>
</IMETHODCALL>
<IPARAMVALUE NAME="LocalOnly"/>
<VALUE>FALSE</VALUE>
</SIMPLEREQ>
</MESSAGE>
</CIM>
```
The DeleteClass operation removes a class from the given namespace in the repository. A sample CIMXML request is as follows:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">  
  <MESSAGE ID="1001" PROTOCOLVERSION="1.0">  
    <SIMPLEREQ>  
      <METHODCALL NAME="DeleteClass">  
        <LOCALNAMESPACEPATH>  
          <NAMESPACE NAME="root"/>  
          <NAMESPACE NAME="widget"/>  
        </LOCALNAMESPACEPATH>  
        <IPARAMVALUE NAME="ClassName">  
          <CLASSNAME NAME="BmTestClass"/>  
        </IPARAMVALUE>  
      </METHODCALL>  
    </SIMPLEREQ>  
  </MESSAGE>  
</CIM>
```
10 Appendix B - Instrumentation in the wbemexec Utility

The wbemexec is a simple command line utility to read the CIMXML request from standard input, send CIMXML message, receive the CIMXML response and output the response to standard output. The following is the function (method) in which wbemexec command sends CIMXML request and receives the CIMXML response.

/**
   * Executes the command using HTTP. A CIM request encoded in XML is read from the input, and encapsulated in an HTTP request message. A channel is obtained for an HTTP connection, and the message is written to the channel. The response is written to the specified outPrintWriter, and consists of the CIM response encoded in XML.
   *
   @param outPrintWriter the ostream to which output should be written
   @param errPrintWriter the ostream to which error output should be written
   @exception WbemExecException if an error is encountered in executing the command
   */
void WbemExecCommand::__executeHttp (ostream& outPrintWriter,
                                      ostream& errPrintWriter)
throw (WbemExecException)
{
    Uint32 size;
    Array <Sint8> content;
    Array <Sint8> contentCopy;
    Array <Sint8> message;
    Array <Sint8> httpHeaders;
    Array <Sint8> httpResponse;
    WbemExecClient client;

    client setTimeout( _timeout );
    // Read in the CIMXML request from a file and perform syntax checks, etc.
    // .......
    // Make a copy of the content because the XmlParser constructor
    // modifies the text
    // contentCopy << content;
    XmlParser parser ((char*) contentCopy.getData ());
    // Here is the place to insert the timestamp as the beginning of the operation
    try
    {
        _connectToServer( client, outPrintWriter );
        //
        // Encapsulate XML request in an HTTP request
        //
        message = XMLProcess::encapsulate( parser, _hostName,
                                            _useMPost, _useHTTP11,
                                            content, httpHeaders );
        if (_debugOutput1)
        {
            outPrintWriter << message.getData () << endl;
        }
    }
try
{
    httpResponse = client.issueRequest( message );
}
catch (ConnectionTimeoutException & ex)
{
    WbemExecException e(WbemExecException::TIMED_OUT);
    throw e;
}
catch (UnauthorizedAccess & ex)
{
    WbemExecException e(WbemExecException::CONNECT_FAIL, ex.getMessage ());
    throw e;
}
catch (Exception & ex)
{
    WbemExecException e(WbemExecException::CONNECT_FAIL, ex.getMessage ());
    throw e;
}

// Here is the place to insert the timestamp as the end of the operation

// Process the response message
// _handleResponse( httpResponse, outPrintWriter, errPrintWriter );

A Perl script invokes the wbemexec command for \( n \) (\( n = 10000 \)?) times. The roundtrip time is accumulated each time when the operation is successfully executed. When the script finishes invoking wbemexec, the accumulated time is divided by the total number of successful operations. The final result for a single roundtrip time would be the average of the time from all the successful operations.
## 11 Appendix C - Bugs/Problems in WBEM Implementations

In this appendix, we created a list of bugs and potential problems with the WBEM servers. The listed bugs and problems may not critical and some of them have workarounds, but they can sometimes cause strange CIMOM behaviours or affect the CIMOM performance significantly. Thus, we generated such a list to let developers be aware of these problems.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Affected Features</th>
<th>WBEM Server</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The incoming instance key array and property array were NOT synchronized properly in CIMOM</td>
<td>CreateInstance in repository</td>
<td>openWBEM 2.0.6</td>
<td>Bug/Fixed in 2.0.12</td>
</tr>
<tr>
<td>2 The incoming instance key array and property array were NOT synchronized properly in CIMOM</td>
<td>CreateInstance in provider</td>
<td>openWBEM 2.0.6</td>
<td>Bug/Fixed in 2.0.12</td>
</tr>
<tr>
<td>3 The EnumerateClassNames operation performs the unnecessary de-serialization of classes</td>
<td>EnumerateClassNames in repository</td>
<td>openWBEM 2.0.6</td>
<td>Bug/Fixed 2.0.12</td>
</tr>
<tr>
<td>4 The repository is implemented using XML files, which cause the repository operations become very inefficient when there are a large number of class or instances</td>
<td>All repository operations except EnumerateClassNames, EnumerateInstanceNames</td>
<td>openPegasus 2.2</td>
<td>Can be a big potential problem in future design; good for debug but not practical. No current proposal on fix.</td>
</tr>
<tr>
<td>5 The providers are unloaded in a very short period of time automatically in Pegasus.</td>
<td>All types of provider operations including indications</td>
<td>openPegasus 2.2</td>
<td>Bug/Fixed in release 2.3; not yet tested locally.</td>
</tr>
<tr>
<td>6 OpenPegasus allows a provider for a standard CIM class, the repository and the provider can both have instance with the same keys. The get Instance operation returns two instances with the same keys: one from repository and one from the provider.</td>
<td>Instance operations in repository and instance provider</td>
<td>openPegasus 2.2</td>
<td>Not investigated yet.</td>
</tr>
<tr>
<td>7 OpenPegasus repository do NOT handle association functions of an instance correctly</td>
<td>Association operations on an instance in the repository</td>
<td>openPegasus 2.2</td>
<td>Bug with Nortel’s own fixes.</td>
</tr>
</tbody>
</table>
12 Appendix D - Sample Test Providers and MOFs

12.1 Sample MOF for Instance Provider Tests

```c++
[Description (  
"Benchmark Instance Provider Test"),  
provider("c++:InstanceProvider") ]  
class BM_4_str_Class : CIM_EnabledLogicalElement  
{  
[Key, Description("Key 1 for benchmark test") ]  
string Key_1;  

[Key, Description("Key 2 for benchmark test") ]  
string Key_2;  

[Key, Description("Key 3 for benchmark test") ]  
string Key_3;  

[Key, Description("Key 4 for benchmark test") ]  
string Key_4;  

[Min(0), Max(2),  
Description ("For benchmark test purpose. If its value is set to 0, "  
"then the provider processing elapsed time is accumulated. "  
"If its value is set to 1, "  
"it marks the end of the benchmark tests, and the provider will print "  
"the elapsed time for internal provider data processing. "  
"If the value is set to 2, then the elapsed time inside the provider is  
reset to 0.") ]  
uint32 testProperty;  
};
```

```c++
[Description (  
"Benchmark Instance Provider Test"),  
provider("c++:InstanceProvider") ]  
class BM_4_int_Class : CIM_EnabledLogicalElement  
{  
[Key, Description("Key 1 for benchmark test") ]  
uint32 Key_1;  

[Key, Description("Key 2 for benchmark test") ]  
uint32 Key_2;  

[Key, Description("Key 3 for benchmark test") ]  
uint32 Key_3;  

[Key, Description("Key 4 for benchmark test") ]  
uint32 Key_4;  

[Min(0), Max(2),  
Description ("For benchmark test purpose. If its value is set to 0, "  
"then the provider processing elapsed time is accumulated. "  
"If its value is set to 1, "  
"it marks the end of the benchmark tests, and the provider will print "  
"the elapsed time for internal provider data processing. "  
"If the value is set to 2, then the elapsed time inside the provider is  
reset to 0.") ]  
uint32 testProperty;  
};
```
12.2 OpenWBEM Sample Instance Provider for Condition A

The following is an instance provider sample used in the instance provider tests under condition A:

```cpp
#include "instanceProvider.h"
namespace
{
    class InstanceProvider
    {
        public:
            InstanceProvider(): InstanceProvider()
            {
                return;
            }

            ~InstanceProvider()
            {
                return;
            }

            void initialize(const OW_ProviderEnvironmentIFCRef& env)
            {
                (void)env;
                return;
            }

            void cleanup()
            {
                return;
            }
    }
}
```
void InstanceProvider::cleanup()
{
    return;
}

void InstanceProvider::enumInstanceNames(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
    const OW_String& className,
    OW_CIMObjectPathResultHandlerIFC& result,
    const OW_Bool& deep,
    const OW_CIMClass& cimClass)
{
    // silly statements to get rid of compiler
    // warnings about unused variables

    (void)env;
    (void)deep;
    (void)cimClass;
    (void)className;
    (void)result;
    (void)ns;

    return;
}

void InstanceProvider::enumInstances(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
    const OW_String& className,
    OW_CIMInstanceResultHandlerIFC& result,
    const OW_Bool& deep,
    const OW_CIMClass& cimClass,
    const OW_Bool& localOnly)
{
    // silly statements to get rid of compiler
    // warnings about unused variables

    (void)env;
    (void)ns;
    (void)className;
    (void)result;
    (void)deep;
    (void)cimClass;
    (void)localOnly;

    return;
}
// method getInstance
// purpose standard openWBEM call for handling
// input as defined by openWBEM
// output as defined by openWBEM
// **************************************************************

OW_CIMInstance InstanceProvider::getInstance(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
    const OW_CIMObjectPath& instanceName,
    const OW_CIMClass& cimClass,
    const OW_Bool& localOnly)
{
    // silly statements to get rid of compiler
    // warnings about unused variables
    (void)env;
    (void)ns;
    (void)instanceName;
    (void)cimClass;
    (void)localOnly;

    OW_CIMInstance rval = OW_CIMInstance(instanceName.getObjectName());
    rval.setClassName(instanceName.getObjectName());
    return rval;
}

// **************************************************************
// class InstanceProvider
// method createInstance
// purpose standard openWBEM function for handling
// call to createInstance
// input as defined by openWBEM
// output as defined by openWBEM
// **************************************************************

OW_CIMObjectPath InstanceProvider::createInstance(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
    const OW_CIMInstance& cimInstance)
{
    // silly statements to get rid of compiler
    // warnings about unused variables
    (void)env;
    (void)ns;
    (void)cimInstance;
    OW_CIMObjectPath op(ns, cimInstance);
    op.setKeys(cimInstance.getKeyValuePairs());
    return op;
}

// **************************************************************
// class InstanceProvider
// method deleteInstance
// purpose standard openWBEM function for handling
// call to deleteInstance
// input as defined by openWBEM
// output as defined by openWBEM
// **************************************************************

void InstanceProvider::deleteInstance(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
    const OW_CIMObjectPath& cop)
{
    // silly statements to get rid of compiler
    // warnings about unused variables

    (void)env;
    (void)ns;
(void) cop;
return;
}

// *****************************************************************************
// class InstanceProvider
// method modifyInstance
// purpose standard openWBEM function for handling
//       call to modifyInstance
// input as defined by openWBEM
// output as defined by openWBEM
// *****************************************************************************

void InstanceProvider::modifyInstance(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
    const OW_CIMInstance& modifiedInstance)
{
    // silly statements to get rid of compiler
    // warnings about unused variables
    (void) env;
    (void) ns;
    (void) modifiedInstance;
    return;
}

} // specify the actual provider library for the CIMOM
OW_PROVIDERFACTORY(InstanceProvider, InstanceProvider)

12.3 OpenWBEM Sample Instance Provider for Condition B

The following is an instance provider sample used in the instance provider tests under
condition B:

//****************************************************************************
// file instanceProvider.cpp for class BmTestClass
// for the CIMOM benchmark tests
// written September 2003
// author Ying Zeng
// history v1.0 September 2003
//****************************************************************************
#include "instanceProvider.h"

namespace
{
    //*****************************************************************************
    // class InstanceProvider
    // method InstanceProvider
    // purpose constructor
    //*****************************************************************************

    InstanceProvider::InstanceProvider(void)
    {
        /*
            cout << "******** In constructor for InstanceProvider " <<
            " for the first time ******** " << endl;
        */
        bmDataPath=OW_CIMObjectPath(dataclassname);
bmDataPath.addKey("Name", OW_CIMValue("Benchmark"));
bmDataPath.addKey("CreationClassName", OW_CIMValue(dataclassname));
FILE *fp;
fp = fopen("/usr/local/lib/openwbem/c++providers/bmResult.txt","w");
fclose(fp);
}

// class InstanceProvider
// method ~InstanceProvider
// purpose destructor

_InstanceProvider::~InstanceProvider(void)
{
    cout<<"******** In InstanceProvider destructor ********"<<endl;
    return;
}

// class InstanceProvider
// method initialize
// purpose initialises the environment
// input reference to originator of
// this provider’s CIMOM handle
// note called by the CIM server after we are
// created

void _InstanceProvider::initialize(const OW_ProviderEnvironmentIFCRef& env)
{
    (void)env;
    env->getLogger()->logDebug("InstanceProvider initialize called");
    cout<<"InstanceProvider::initialize"<<endl;
    myEnv = env;
    OW_CIMOMHandleIFCRef hd1 = myEnv->getCIMOMHandle();
    OW_CIMInstance inst=hd1->getInstance(defaultns, bmDataPath);
    inst.setProperty("CIElapsedTime", OW_CIMValue(0));
    inst.setProperty("ECIElapsedTime", OW_CIMValue(0));
    inst.setProperty("GIElapsedTime", OW_CIMValue(0));
    inst.setProperty("EnElapsedTime", OW_CIMValue(0));
    inst.setProperty("EEEnElapsedTime", OW_CIMValue(0));
    inst.setProperty("EIElapsedTime", OW_CIMValue(0));
    inst.setProperty("EIIElapsedTime", OW_CIMValue(0));
    inst.setProperty("MIElapsedTime", OW_CIMValue(0));
    inst.setProperty("EIMiElapsedTime", OW_CIMValue(0));
    inst.setProperty("EDIElapsedTime", OW_CIMValue(0));
    inst.setProperty("ECiCount", OW_CIMValue(0));
    inst.setProperty("ECiCount", OW_CIMValue(0));
    inst.setProperty("GICount", OW_CIMValue(0));
    inst.setProperty("EnCount", OW_CIMValue(0));
    inst.setProperty("EEEnCount", OW_CIMValue(0));
    inst.setProperty("EICount", OW_CIMValue(0));
    inst.setProperty("EIECount", OW_CIMValue(0));
    inst.setProperty("MIECount", OW_CIMValue(0));
    inst.setProperty("EMiCount", OW_CIMValue(0));
    inst.setProperty("DiCount", OW_CIMValue(0));
    inst.setProperty("EDIcount", OW_CIMValue(0));
    hd1->modifyInstance(defaultns, inst);
    return;
}
class InstanceProvider
method cleanup
purpose frees up all the memory that this provider has used
note called by the CIM server before we are destroyed

void InstanceProvider::cleanup()
{
    myEnv->getLogger()->logDebug("InstanceProvider cleanup called");
    cout<<"InstanceProvider::cleanup"<<endl;
}

class InstanceProvider
method enumInstanceNames
purpose standard openWBEM function to handle the enumInstanceNames call
input as defined by openWBEM
output as defined by openWBEM

void InstanceProvider::enumInstanceNames(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
    const OW_String& className,
    OW_CIMObjectPathResultHandlerIFC& result,
    const OW_Bool& deep,
    const OW_CIMClass& cimClass)
{
    // silly statements to get rid of compiler warnings about unused variables
    (void)env;
    (void)deep;
    (void)cimClass;
    (void)className;
    (void)result;
    (void)ns;

timeval startT;
timeval endT;
OW(UInt64 elapsedT;
    // OW_CIMObjectPath op;
    unsigned int i;

    //cout << "In InstanceProvider::enumInstanceNames " << endl;
    gettimeofday(&startT, NULL);
    OW_CIMOMHandleIFCRefhdl = myEnv->getCIMOMHandle();
    OW_CIMInstance inst = hdl->getInstance(defaultns, bmDataPath);

    for (i=0; i<myInstances.size(); i++)
    {
        OW_CIMObjectPath op(ns, myInstances[i]);
        result.handle(op);
    }

    OW(UInt32 n;
    inst.getProperty("EnCount").getValue().get(n);
    inst.setProperty("EnCount",OW_CIMValue(++n));
    OW(UInt64 t;
    inst.getProperty("EnElapsedTime").getValue().get(t);
    gettimeofday(&endT, NULL);
    elapsedT = (endT.tv_sec-startT.tv_sec)*10000000+endT.tv_usec-startT.tv_usec;
    inst.setProperty("EnElapsedTime",OW_CIMValue(t+elapsedT));
    hdl->modifyInstance(defaultns, inst);

    return;
}
void InstanceProvider::enumInstances(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
    const OW_String& className,
    OW_CIMInstanceResultHandlerIFC& result,
    const OW_Bool& deep,
    const OW_CIMClass& cimClass,
    const OW_Bool& localOnly)
{
    // silly statements to get rid of compiler
    // warnings about unused variables
    (void) env;
    (void) ns;
    (void) className;
    (void) result;
    (void) deep;
    (void) cimClass;
    (void) localOnly;

    timeval startT;
    timeval endT;
    OW(UInt64) elapsedT;
    unsigned int i;

    //cout << "In InstanceProvider::enumInstances " << endl;
    gettimeofday(&startT, NULL);
    OW_CIMOMHandleIFCRef hd1 = myEnv->getCIMOMHandle();
    OW_CIMInstance inst = hd1->getInstance(defaultNs, bmDataPath);
    for (i=0; i<myInstances.size(); i++)
    {
        result.handle(myInstances[i]);
    }

    OW(UInt32) n;
    inst.getProperty("EiCount").getValue().get(n);
    inst.setProperty("EiCount", OW_CIMValue(++n));
    OW(UInt64) t;
    inst.getProperty("EiElapsedTime").getValue().get(t);
    gettimeofday(&endT, NULL);
    elapsedT = (endT.tv_sec-startT.tv_sec)*10000000+endT.tv_usec-startT.tv_usec;
    inst.setProperty("EiElapsedTime", OW_CIMValue(t+elapsedT));
    hd1->modifyInstance(defaultNs, inst);

    return;
}

void InstanceProvider::getInstance(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
const OW_CIMObjectPath& instanceName,
const OW_CIMClass& cimClass,
const OW_Bool& localOnly)
{
    // silly statements to get rid of compiler
    // warnings about unused variables
    (void)env;
    (void)ns;
    (void)instanceName;
    (void)cimClass;
    (void)localOnly;

    OW_CIMInstance rval = OW_CIMInstance(instanceName.getObjectName());
    rval.setClassName(instanceName.getObjectName());

timeval startT;
timeval endT;
OW_UInt64 elapsedT;
OW_CIMObjectPath op;
unsigned int i;
unsigned int flag = 0;
OW_CIMObjectPath tp;

// cout << "In InstanceProvider::getInstance " << endl;
gettimeofday(&startT, NULL);
OW_CIMOMHandleIFCRef hd1 = myEnv->getCIMOMHandle();
OW_CIMInstance inst = hd1->getInstance(defaultns, bmDataPath);

for (i = 0; i < myInstances.size(); i++)
{
    op = OW_CIMObjectPath(ns, myInstances[i]);
    // This is to avoid the default root namespace problem in openWBEM
    tp = instanceName;
    tp.setNamespace(defaultns);
    if (op.equals(tp))
    {
        OW_UInt32 n;
        inst.getProperty("GICollection").getValue().get(n);
        inst.setProperty("GICollection", OW_CIMValue(+n));
        flag = 1; // found the instance
        rval = myInstances[i];
        break;
    }
}

if (flag == 0)
{
    OW_UInt32 n;
    inst.getProperty("EGICollection").getValue().get(n);
    inst.setProperty("EGICollection", OW_CIMValue(+n));
    OW_UInt64 t;
    inst.getProperty("EGElapsedCount").getValue().get(t);
    gettimeofday(&endT, NULL);
    elapsedT = (endT.tv_sec-startT.tv_sec)*10000000+endT.tv_usec-startT.tv_usec;
    inst.setProperty("EGElapsedCount", OW_CIMValue(t+elapsedT));
    hd1->modifyInstance(defaultns, inst);
    OW_THROWCIMMSG(OW_CIMException::NOT_FOUND,
                   "The requested test instance is not found");
}

OW_UInt64 t;
inst.getProperty("EGElapsedTime").getValue().get(t);
gettimeofday(&endT, NULL);
elapsedT = (endT.tv_sec-startT.tv_sec)*10000000+endT.tv_usec-startT.tv_usec;
inst.setProperty("EGElapsedTime", OW_CIMValue(t+elapsedT));
hd1->modifyInstance(defaultns, inst);

return rval;
}

// **********************************************************************

Nortel Networks
// class InstanceProvider
// method createInstance
// purpose standard openWBEM function for handling
// input as defined by openWBEM
// output as defined by openWBEM
// ***********************************************

OW_CIMObjectPath InstanceProvider::createInstance(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
    const OW_CIMInstance& cimInstance)
{
    // silly statements to get rid of compiler
    // warnings about unused variables
    (void)env;
    (void)ns;
    (void)cimInstance;
    OW_CIMObjectPath op(ns, cimInstance);
    op.setKeys(cimInstance.getKeyValuePairs());

    timeval startT;
    timeval endT;
    OW_UI64 elapsedT;
    OW_CIMObjectPath tmpop;
    unsigned int i;

    //cout << "In InstanceProvider::createInstance " << endl;
    gettimeofday(&startT, NULL);
    OW_CIMOMHandleIFCRef hd1 = myEnv->getCIMOMHandle();
    OW_CIMInstance inst = hd1->getInstance(defaultns, bmDataPath);
    OW_CIMObjectPath op(ns, cimInstance);
    for (i=0; i<myInstances.size(); i++)
    {
        tmpop=OW_CIMObjectPath(ns, myInstances[i]);
        if (tmpop.equals(op))
        {
            OW_UI32 n;
            inst.getProperty("ECiCount").getValue().get(n);
            inst.setProperty("ECiCount", OW_CIMValue(n+1));
            OW_UI64 t;
            inst.getProperty("ECiElapsedTime").getValue().get(t);
            gettimeofday(&endT, NULL);
            elapsedT = (endT.tv_sec-startT.tv_sec)*1000000+endT.tv_usec-
            startT.tv_usec;
            inst.setProperty("ECiElapsedTime", OW_CIMValue(t+elapsedT));
            hd1->modifyInstance(defaultns, inst);
            OW_THROWCIMMSG(OW_CIMException::ALREADY_EXISTS,
                "Unable to create instance: instance already exists");
        }
    }
    myInstances.append(cimInstance);

    OW_UI32 n;
    inst.getProperty("CiCount").getValue().get(n);
    inst.setProperty("CiCount", OW_CIMValue(n+1));
    OW_UI64 t;
    inst.getProperty("CiElapsedTime").getValue().get(t);
    gettimeofday(&endT, NULL);
    elapsedT = (endT.tv_sec-startT.tv_sec)*1000000+endT.tv_usec-startT.tv_usec;
    inst.setProperty("CiElapsedTime", OW_CIMValue(t+elapsedT));
    hd1->modifyInstance(defaultns, inst);
    return op;
}

// ***********************************************
// class InstanceProvider
// method deleteInstance
// purpose standard openWBEM function for handling
// input as defined by openWBEM
/**
 * This is an example of an instance provider that
 * implements the OAM NOTIFY function.
 */

void InstanceProvider::deleteInstance(
    const OW_ProviderEnvironmentIFCRef& env,
    const OW_String& ns,
    const OW_CIMObjectPath& cop)
{
    // silly statements to get rid of compiler
    // warnings about unused variables
    (void)env;
    (void)ns;
    (void)cop;

    timeval startT;
    timeval endT;
    OW(UInt64) elapsedT;
    unsigned int i;

    // cout << "In InstanceProvider::deleteInstance " << endl;
    gettimeofday(&startT, NULL);
    OW_CIMOMHandleIFCRef hd1 = myEnv->getCIMOMHandle();
    OW_CIMInstance inst = hd1->getInstance(defaultns, bmDataPath);
    OW_CIMObjectPath tmpop = cop;
    tmpop.setNamespace(ns);

    for (i=0; i<myInstances.size(); i++)
    {
        OW_CIMObjectPath op(ns, myInstances[i]);
        if (op.equals(tmpop))
        {
            myInstances.remove(i);
            OW(UInt32) n;
            inst.getProperty("DiCount").getValue().get(n);
            inst.setProperty("DiCount", OW_CIMValue(n++));
            OW(UInt64) t;
            inst.getProperty("DiElapsedTime").getValue().get(t);
            gettimeofday(&endT, NULL);
            elapsedT = (endT.tv_sec-startT.tv_sec)*1000000+endT.tv_usec-
            startT.tv_usec;
            inst.setProperty("DiElapsedTime", OW_CIMValue(t+elapsedT));
            hd1->modifyInstance(defaultns, inst);
            return;
        }
    }

    OW(UInt32) n;
    inst.getProperty("EdiCount").getValue().get(n);
    inst.setProperty("EdiCount", OW_CIMValue(n++));
    OW(UInt64) t;
    inst.getProperty("EdiElapsedTime").getValue().get(t);
    gettimeofday(&endT, NULL);
    elapsedT = (endT.tv_sec-startT.tv_sec)*1000000+endT.tv_usec-
    startT.tv_usec;
    inst.setProperty("EdiElapsedTime", OW_CIMValue(t+elapsedT));
    hd1->modifyInstance(defaultns, inst);
    OW_THROWCIMMSG(OW_CIMException::FAILED,
        "Unable to delete the requested instance: instance not found");

    return;
}

/**
 * This is an example of an instance provider that
 * implements the OAM NOTIFY function.
 */

void InstanceProvider::modifyInstance()
const OW_ProviderEnvironmentIFCRef& env,
const OW_String& ns,
const OW_CIMInstance& modifiedInstance)
{
    // silly statements to get rid of compiler
    // warnings about unused variables
    (void)env;
    (void)ns;
    (void)modifiedInstance;

timeval startT;
timeval endT;
OW_UInt64 elapsedT;
unsigned int i;
unsigned int flag=0;

    // cout << "In InstanceProvider::modifyInstance " << endl;
    gettimeofday(&startT, NULL);
    OW_CIMOMHandleIFCRefhdl=myEnv->getCIMOMHandle();
    OW_CIMInstance inst = hdl->getInstance(defaultns, bmDataPath);
    OW_CIMObjectPath op(defaultns, modifiedInstance);
    for (i=0; i<myInstances.size(); i++)
    {
        OW_CIMObjectPath tmp(defaultns, myInstances[i]);
        if (tmp.equals(op))
        {
            flag=1;
            myInstances[i].setProperties(modifiedInstance.getProperties());
            OW_UInt32 n;
            inst.getProperty("MiCount").getValue().get(n);
            inst.setProperty("MiCount", OW_CIMValue(++n));
            OW_UInt64 t;
            inst.getProperty("MiElapsedTime").getValue().get(t);
            gettimeofday(&endT, NULL);
            elapsedT = (endT.tv_sec-startT.tv_sec)*10000000+endT.tv_usec-startT.tv_usec;
            inst.setProperty("MiElapsedTime", OW_CIMValue(t+elapsedT));
            hdl->modifyInstance(defaultns, inst);
            return;
        }
    }
    if (flag == 0)
    {
        OW_UInt32 n;
        inst.getProperty("EMiCount").getValue().get(n);
        inst.setProperty("EMiCount", OW_CIMValue(++n));
        OW_UInt64 t;
        inst.getProperty("EMiElapsedTime").getValue().get(t);
        gettimeofday(&endT, NULL);
        elapsedT = (endT.tv_sec-startT.tv_sec)*10000000+endT.tv_usec-startT.tv_usec;
        inst.setProperty("EMiElapsedTime", OW_CIMValue(t+elapsedT));
        hdl->modifyInstance(defaultns, inst);
        OW_THROWCIMMSG(OW_CIMException::FAILED,
            "Unable to modify the requested instance: instance not found");
    }
    return;
}

// specify the actual provider library for the CIMOM
OW_PROVIDERFACTORY(InstanceProvider, InstanceProvider)