Abstract: This PEP proposes adding a simple benchmark Test utility to OpenPegasus.

Definition of the Problem

By supporting a common, industry standard interface for monitoring and accessing management data, WBEM offers a significant number of benefits to instrumentation developers, management application developers and system administrators.

However, with the introduction of any interface layer comes an additional cost. This cost needs to be considered as part of a decision to use the WBEM interface to expose and access management data. The benchmarkTest utility has been developed as a tool to help developers characterize the performance impact of moving to WBEM.

In addition, this tool could help the OpenPegasus Community monitor performance changes during the development lifecycle.

Proposed Solution

Note 1: The code segments and output shown in this PEP, although representative of the final submission, are still subject to change.

Note 2: A base set of tests are proposed in this PEP. As part of this PEP, we are proposing that the addition of new test cases and configurations to this utility be handled using the Bugzilla process.

Solution Description
The WBEM benchmarkTest utility consists of five components:

- **benchmarkTest**: the benchmarkTest CIM Client application
- **libbenchmarkProviderModule**: the benchmarkTest CIM Provider
- **libbenchmarkDefinition**: a shared library containing common definitions and functions
- **benchmarkProvider.mof**: the benchmarkTest class definitions
- **benchmarkProviderR.mof**: the benchmarkTest provider registration MOF

**Benchmark Utility Components**

The **benchmarkTest Client** is a CIM Client that (1) issues a canned set of requests against the benchmarkTest Provider, (2) monitors the Elapse Time between requests and responses, and (3) generates a simple report describing the results.

The **benchmarkTest Provider** is a CIM Instance Provider that generates a response of a requested size.

**Class Definitions**
The benchmarkTest Provider uses the class name to determine the amount of data to return in the response. The benchmark Provider will "support" classes that use the following format for class name:

BenchmarkClassPxxxxSyyyyIzzzz

The benchmarkTest Provider uses the values of xxxx, yyyy, and zzzz in the class name string (i.e., BenchmarkClassPxxxxSyyyyIzzzz) to determine the number of non-key Properties (xxxx), the Size (yyyy) of each non-key Property, and the number of Instances (zzzz) to return in the response. E.g., calling enumerateInstances with the class name benchmarkClassP0001S1000I0010 will return a response with 10 instances; each instance will have 50 properties and each property will return a string with 100 characters.

The following figure shows class definitions for the benchmarkClassP0001S1000I0010, benchmarkClassP0010S0010I0010 and benchmarkClassP0010S0100I0010. The class definitions for the benchmark classes are defined in benchmarkProvider.mof.

```plaintext
class benchmarkClassP0001S1000I0010 
{
    [key] Uint32 Identifier;
    string Property0001;
};

class benchmarkClassP0010S0010I0010 
{
    [key] Uint32 Identifier;
    string Property0001;
    string Property0002;
    string Property0003;
    string Property0004;
    string Property0005;
    string Property0006;
    string Property0007;
    string Property0008;
    string Property0009;
    string Property0010;
};

class benchmarkClassP0010S0100I0010 
{
    [key] Uint32 Identifier;
    string Property0001;
    string Property0002;
    string Property0003;
    string Property0004;
    string Property0005;
    string Property0006;
    string Property0007;
    string Property0008;
    string Property0009;
    string Property0010;
};
```
**Note 1:** The benchmarkTest Provider will never return more instances or properties than allowed by an operation. E.g., calling getInstance with the class name benchmarkClassP0050S0100I0010 will return a response with a single instance; this instance will have 50 non-key properties of size 100.

**Note 2:** All benchmarkTest classes MUST contain a single key property named Identifier.

```
[key] Uint32 Identifier;
```

**Provider**

The benchmarkTest Provider is a CIM Instance Provider that generates a response of a requested size. As an example, the following figure contains the implementation of enumerateInstances.

```cpp
void benchmarkProvider::enumerateInstances(
    const OperationContext & context,
    const CIMObjectPath & classReference,
    const Boolean includeQualifiers,
    const Boolean includeClassOrigin,
    const CIMPropertyList & propertyList,
    InstanceResponseHandler & handler)
{
    CIMInstance _instance;
    Uint32 numberOfProperties;
    Uint32 sizeOfPropertyValue;
    Uint32 numberOfInstances;

    CIMName className = classReference.getClassName();
    test.getConfiguration(className, numberOfProperties,
                           sizeOfPropertyValue, numberOfInstances);

    // begin processing the request
    handler.processing();

    for (Uint32 i = 1; i <= numberOfInstances; i++)
    {
        _instance = _buildInstance(className,
                                    numberOfProperties,
                                    sizeOfPropertyValue , CIMValue(i));
        handler.deliver(_instance);
    }

    // complete processing the request
    handler.complete();
}
```

**Client**
The benchmarkTest Client is a CIM Client that (1) issues a canned set of requests against the benchmarkTest Provider, (2) monitors the Elapse Time between requests and the responses, and (3) generates a simple report describing the results.

```
benchmarkTest [ -h hostname ] [ -p portnumber ] [ -t timeout ] [ -u username ] [ -w password ] [ -s ] [ -i iterations ]
```

**Test Definition**

The benchmarkTest Client implements 5 different tests. With the exception of test 2, each test is designed to run a specified number of iterations. The default number of iterations is 10. This value can be changed using the `[-i iterations]` parameter on the benchmarkTest command line.

- **Test 1 – Connect/Disconnect:**

  This test simply connects to the CIM Server and then immediately disconnects.

  The following figure shows the output generated for Test 1. This output shows that the test was run 10 times with a Total Elapse Time of 0.004 seconds. The Average Elapse Time is computed by dividing the Total Elapse Time by the number of iterations (e.g., 10).

  1: Benchmark Test #1: Connect/Disconnect Test
  10 requests processed in 0.004 Seconds (Average Elapse Time = 0.0004)

- **Test 2 – Provider Load:**

  This test times the "first invocation" of the provider. It includes the overhead associated with the load and initialization of the provider library.

  The benchmarkTest client uses a getInstance request on the class benchmarkClassP0001S0010I0001 for this operation. The following output shows the time required to perform this operation was 0.032 seconds.

  2: Benchmark Test #2: Load Provider Test on class benchmarkClassP0001S0010I0001
  Connect time = 0
  Unload Module time = 1.773
  First getInstance request processed in 0.032 Seconds

  **Note 1:** The output of this test also includes the time (i.e., Unload Module Time) required to "disable" and "enable" the Provider. The Unload Module Time will vary depending on whether or not the Provider was already loaded.
Note 2: If the Elapse Time between the request and the response is small (i.e., < 0.001 second), a "Connect time" of "0" will be displayed.

- **Test 3 – getInstance:**

  This test calls getInstance on a specified class.

  The following figure shows the Test 3 code segment used to invoke getInstance.

  ```java
  stopwatchTime.reset();
  
  CIMObjectPath reference =
      benchmarkTestCommand::_buildObjectPath(className,
      CIMValue(99));
  
  for (Uint32 i = 0; i < _iterations; i++)
  {
    CIMInstance cimInstance = client.getInstance(NAMESPACE,
    reference);
    CIMObjectPath instanceRef = cimInstance.getPath();
    if ( !(instanceRef.getClassName().equal(className)))
    {
      outPrintWriter << "Returned ClassName = "
          << instanceRef.getClassName().getString() << endl;
      outPrintWriter << "Expected ClassName = "
          << className.getString() << endl;
      errorExit(errPrintWriter, "getInstance failed");
    }
  }
  
  double elapsedTime = stopwatchTime.getElapsed();
  
  The following figure shows the output generated for Test 3. This output shows that the test was run 10 times on class benchmarkClassP0001S0010I0001 with a Total Elapse Time of 0.147 seconds.

  3: Benchmark Test #3: getInstance Test on benchmarkClassP0001S0010I0001
  Connect time = 0
  Number of Non-Key Properties Returned = 1
  Size of Each Non-Key Property Returned = 10
  Number of Instances Returned = 1
  10 requests processed in 0.147 Seconds (Average Elapse Time = 0.0147)
**Test 4 – enumerateInstanceNames:**

This test calls enumerateInstanceNames on a specified class.

The following figure shows the output generated for Test 4. This output shows that the test was run 10 times on class benchmarkClassP0001S0010I0001 with a Total Elapse Time of 0.153 seconds.

```
4: Benchmark Test #4: enumerateInstanceNames Test on class benchmarkClassP0001S0010I0001
Connect time = 0
Number of Non-Key Properties Returned = 0
Number of Instances Returned = 1
10 requests processed in 0.153 Seconds (Average Elapse Time = 0.0153)
```

**Note 1:** The enumerateInstanceNames operation returns ‘0’ non-key properties.

**Test 5 – enumerateInstances:**

This test calls enumerateInstances on a specified class.

The following figure shows the output generated for Test 5. This output shows that the test was run 10 times on class benchmarkClassP0001S0010I0001 with a Total Elapse Time of 0.153 seconds.

```
5: Benchmark Test #5: enumerateInstances Test on class benchmarkClassP0001S0010I0001
Connect time = 0
Number of Non-Key Properties Returned = 1
Size of Each Non-Key Property Returned = 10
Number of Instances Returned = 1
10 requests processed in 0.153 Seconds (Average Elapse Time = 0.0153)
```

**Class Definitions**

In the benchmarkTest, tests 3, 4 and 5 are repeated for the following classes. A total of 83 tests are run during a single invocation of benchmarkTest.

**Note:** This PEP proposes that the addition of new test cases and configurations to this utility be handled using the Bugzilla process.
Connection Types

HP WBEM Services supports multiple CIM-XML connection types between a CIM Client and the CIM Server.

The following table describes the supported communication protocols between a remote CIM Client and CIM Server.

<table>
<thead>
<tr>
<th>From Component</th>
<th>To Component</th>
<th>Encoding</th>
<th>Communication Protocol</th>
<th>Port</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIM Client</td>
<td>CIM Server</td>
<td>CIM-XML</td>
<td>HTTP</td>
<td>5988</td>
<td></td>
</tr>
<tr>
<td>CIM Client</td>
<td>CIM Server</td>
<td>CIM-XML</td>
<td>HTTPS</td>
<td>5989</td>
<td></td>
</tr>
</tbody>
</table>

Local connection types are defined in the following table.
<table>
<thead>
<tr>
<th>From Component</th>
<th>To Component</th>
<th>Encoding</th>
<th>Communication Protocol</th>
<th>Port</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIM Client</td>
<td>CIM Server</td>
<td>CIM-XML</td>
<td>connectLocal</td>
<td></td>
<td>Protocol will vary depending on platform.</td>
</tr>
<tr>
<td>CIM Client</td>
<td>CIM Server</td>
<td>CIM-XML</td>
<td>HTTP</td>
<td>5988</td>
<td></td>
</tr>
<tr>
<td>CIM Client</td>
<td>CIM Server</td>
<td>CIM-XML</td>
<td>HTTPS</td>
<td>5989</td>
<td></td>
</tr>
</tbody>
</table>

The benchmarkTest utility supports command line parameters that allow the connection type to be specified.

**Synopsis**

```
benchmarkTest [ -h hostname ] [ -p portnumber ] [ -t timeout ] [ -u username ] [ -w password ] [ -s ] [ -i iterations ]
```

**Description**

By default, the benchmark tests are executed on the local host, using a connectLocal. By default, benchmarkTest waits 20000 milliseconds (20 seconds) on sending a request, then times out if a response hasn't been received. The -h option allows the user to specify a remote host. The -p option allows the user to specify a different port number. The -t option allows the user to specify, in milliseconds, a different timeout value for the request. The -u and -w options allow the user to specify a username and password to use for authentication of the user. The -s option enables the SSL protocol between benchmarkTest and the CIM Server.

**Options**

benchmarkTest recognizes the following options:

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

- **-p portnumber**
  Use the specified port number. The port number must be the port number on which the CIM Server is running on the specified host. If port 5989 is specified, the SSL protocol is used.

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

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

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

- **-s**
  Enable the use of the SSL protocol between benchmarkTest and the CIM Server.
server. If -s is specified and a port is not, port 5989 will be assumed.

-`i iterations`
Run each test the specified number of iterations. The default value for iterations is 10.

**Return Value**

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

The following return values are returned:

- 0 Success
- 1 Error

---

**Schedule**

<table>
<thead>
<tr>
<th>Action</th>
<th>Planned</th>
<th>Actual</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEP Submitted</td>
<td>15 May 2003</td>
<td>15 May 2003</td>
<td></td>
</tr>
<tr>
<td>PEP Reviewed</td>
<td>23 May 2003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEP Approved</td>
<td>31 May 2003</td>
<td>06 June 2003</td>
<td>Ballot 14</td>
</tr>
<tr>
<td>Code Committed</td>
<td>30 June 2003</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Copyright (c) 2003 BMC Software; Hewlett-Packard Development Company, L.P.; IBM Corp.; The Open Group**

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

THE ABOVE COPYRIGHT NOTICE AND THIS PERMISSION NOTICE SHALL BE INCLUDED IN ALL COPIES OR SUBSTANTIAL PORTIONS OF THE SOFTWARE. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.