WBEM Scripting tools for ESXi
PyWBEM How-To
Version 0.2

Document Revision History

Date	Author	Version	Comments
06-12-2008	Shashi Dande	0.1	Document Started
06-14-2008	Shashi Dande	0.2	Added Installer screenshots

Table of Contents

1. Glossary

Terms	Definition
CIM	Common Information Model
CIMOM	CIM Object Manager
DMTF	Distributed Management Task Force
IPMI	Intelligent Platform Management Interface
Python	Interpreted object-oriented programming language. Python runs on Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, and Nokia mobile phones.
PyWBEM	Python WBEM Client
OS	Operating System
SFCB	Small footprint connection broker. The CIMOM uesd on VMware ESXi
WBEM	Web-Based Enterprise Management

2. HP Provider Architecture on ESXi

2.1 Overview

The WBEM provider initiative on VMware ESXi aims to improve the customer experience for HP systems management by providing more secure and standards based management model which is simple and consistent in data presentation. This will be alternative to SNMP via the implementation of WBEM providers that will provide event notifications, system status and inventory data for HP hardware.

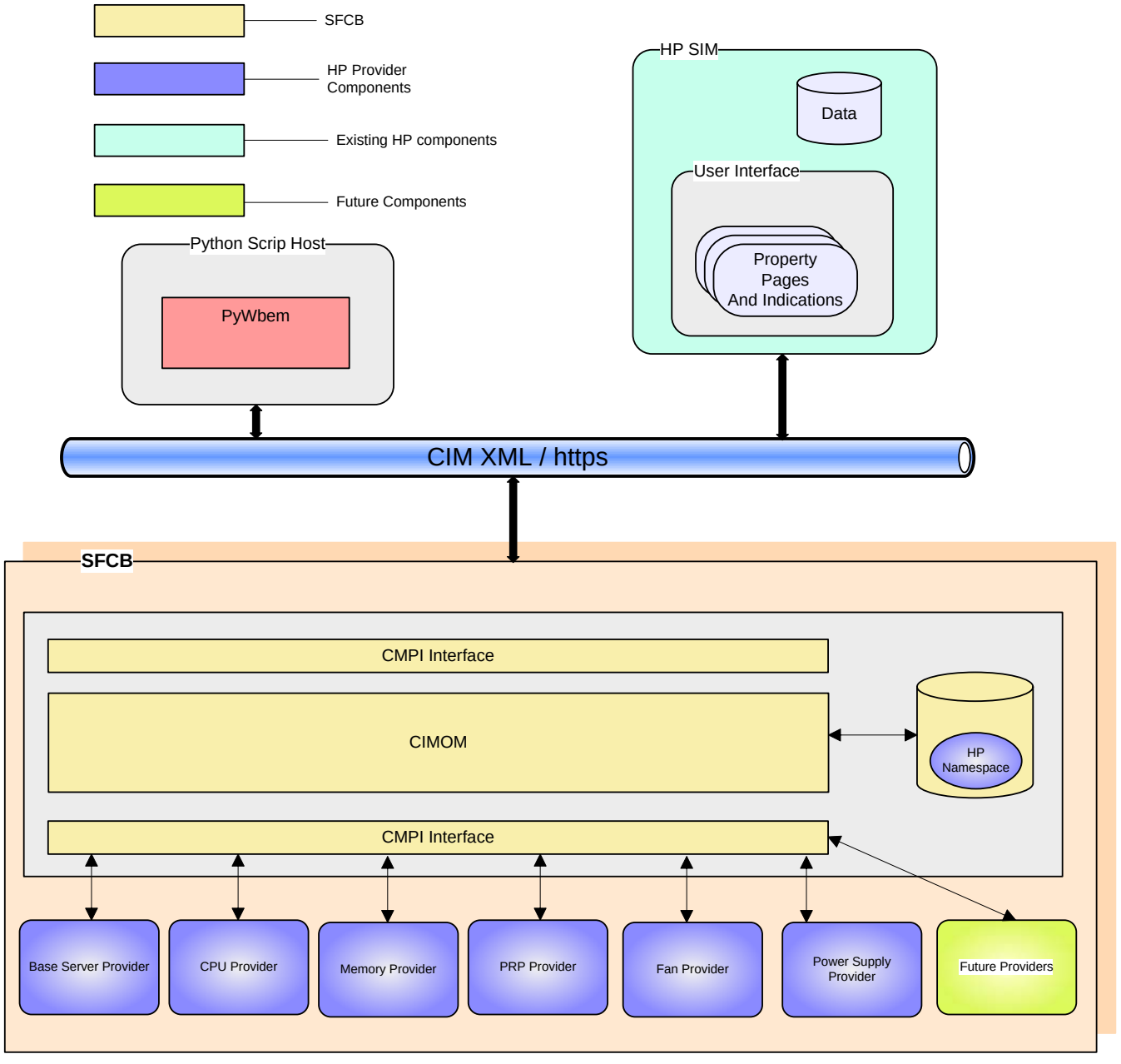


Figure 1 – Overall Architecture

3. PyWBEM

3.1 Python Overview

Python is an interpreted object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools, comes with extensive standard libraries.

Python runs on Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, and Nokia mobile phones. Python has also been ported to the Java and .NET virtual machines.
Python is distributed under an open source license that makes it free to use, even for commercial products.

Python can be downloaded from the following site.
http://www.python.org/download/

3.2 PyWBEM Overview

PyWBEM is a Python library for making CIM operations over HTTP/HTTPS using the WBEM CIM-XML protocol. It is based on the idea that a good WBEM client should be easy to use and not necessarily require a large amount of programming knowledge. PyWBEM is suitable for a large range of tasks from writing web and GUI applications to simple command line scripts.

Since PyWBEM is a pure Python module, it should work on any operating system that has a Python interpreter installed although it has only been tested on Linux and Windows. Python 2.3 or higher is required.

PyWBEM can be downloaded from the following page.

http://sourceforge.net/project/showfiles.php?group_id=133883

3.3 PyWBEM Installation

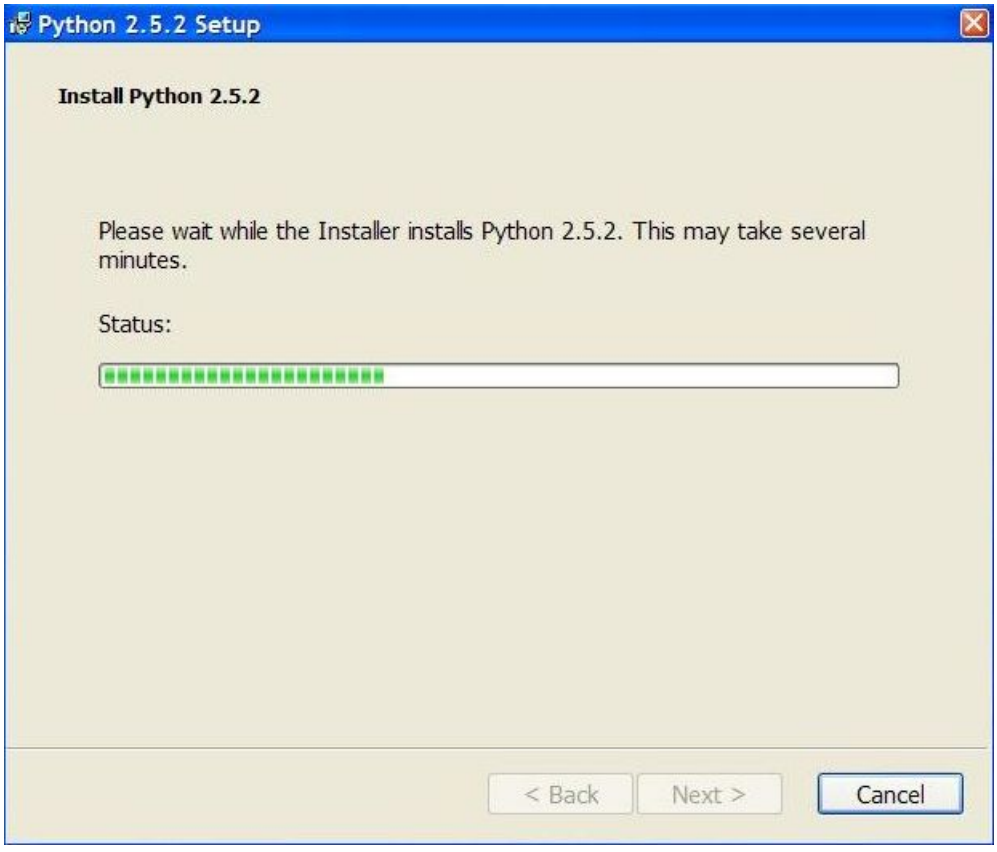
PyWBEM can be installed on Windows client using the following steps.

1. Install the Python MSI module downloaded from the above download site. For example if you have downloaded Python 2.5.2 MSI (python-2.5.2.msi). Install it by double clicking it from Windows explorer window.

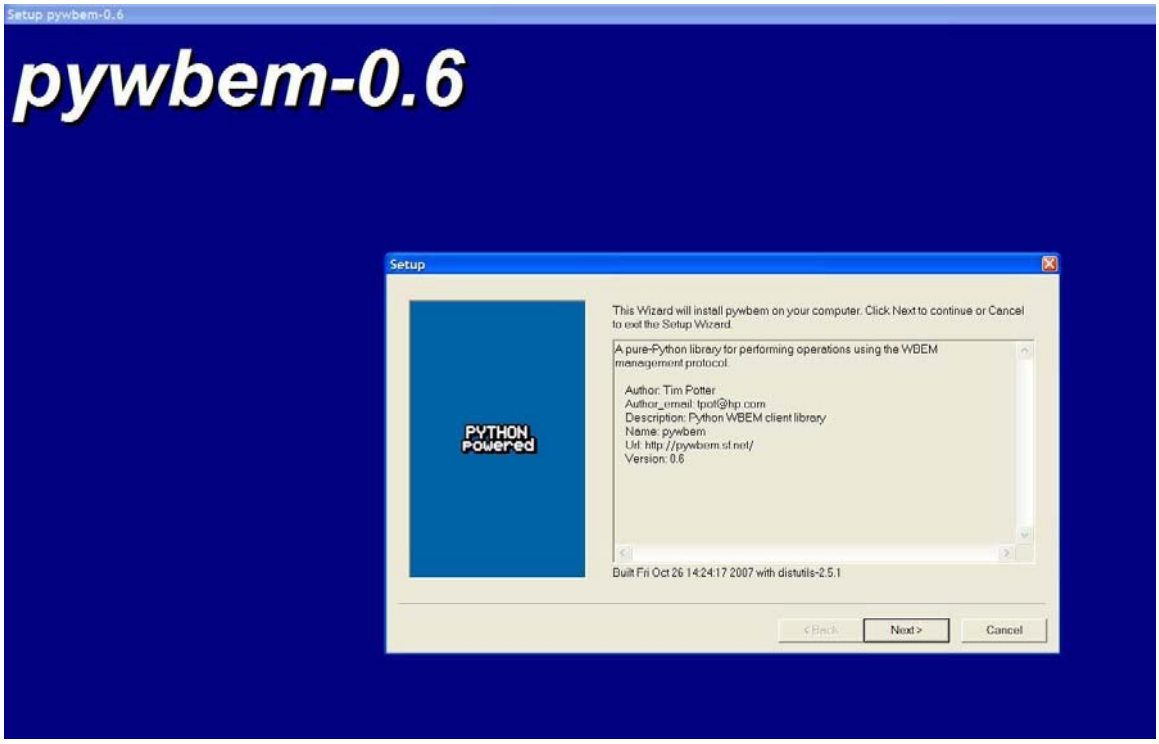

Python 2.5.2 Setup
Select whether to install Python 2.5.2 for all users of this computer.
Install for all users
Install just for me
python
for
windows
Back
Next >
Cancel

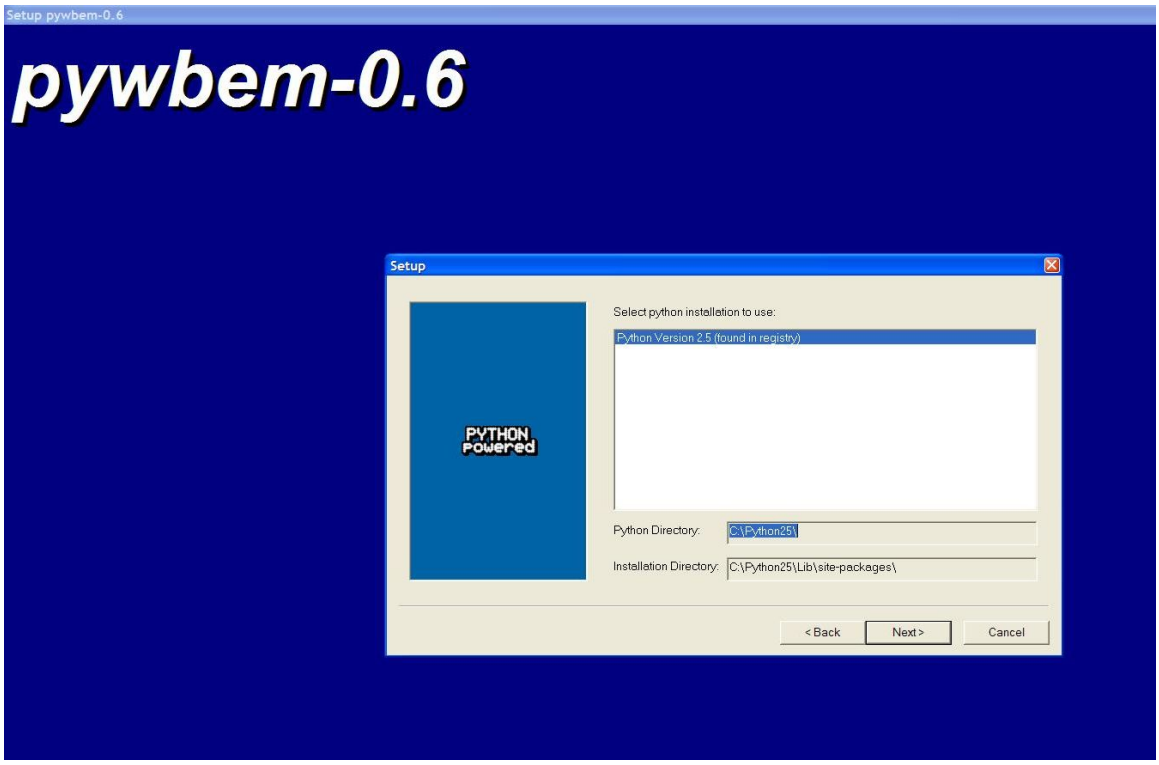

Python 2.5.2 Setup
Select Destination Directory
Please select a directory for the Python 2.5.2 files.
Python25
Up
New
Lib
C:\Python25\
python
for
windows
< Back
Next >
Cancel

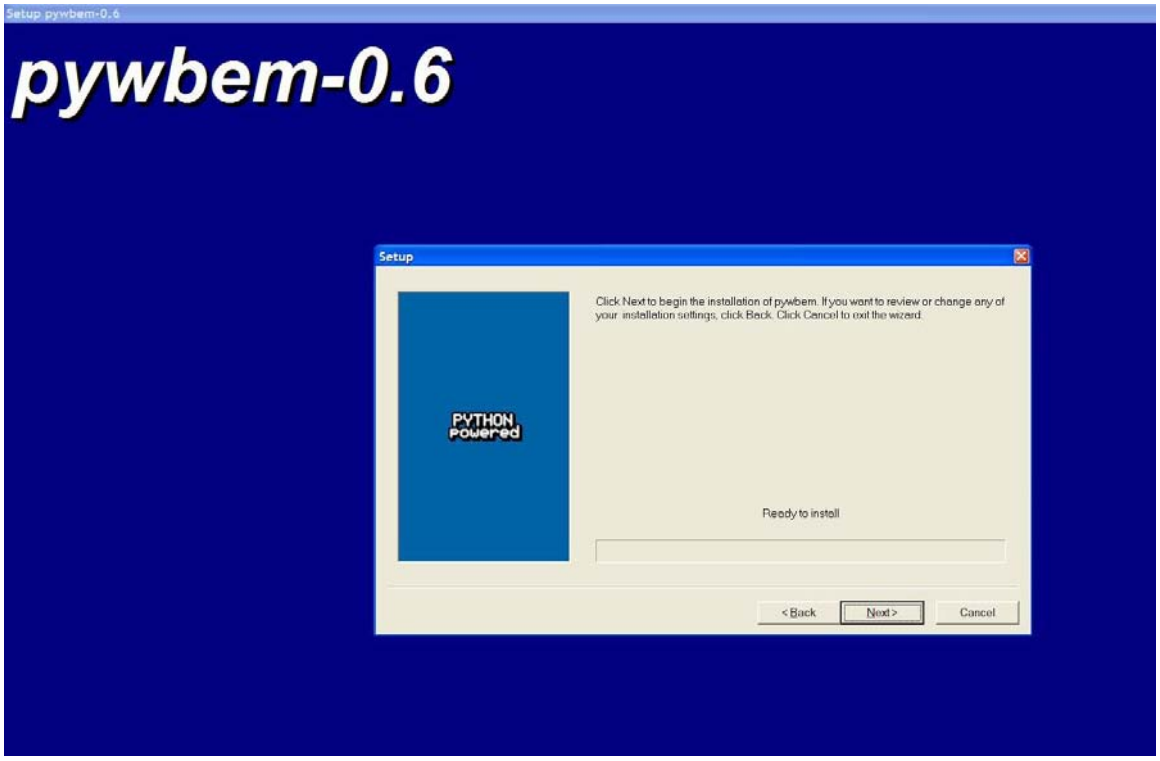

Python 2.5.2 Setup
Customize Python 2.5.2
Select the way you want features to be installed.
Click on the icons in the tree below to change the way features will be installed.
Python
Register Extensions
Tcl/Tk
Documentation
Utility Scripts
Test suite
Python Interpreter and Libraries
This feature requires 18MB on your hard drive. It has 5 of 5 subfeatures selected. The subfeatures require 18MB on your hard drive.
python
for
windows
Disk Usage
Advanced
< Back
Next >
Cancel

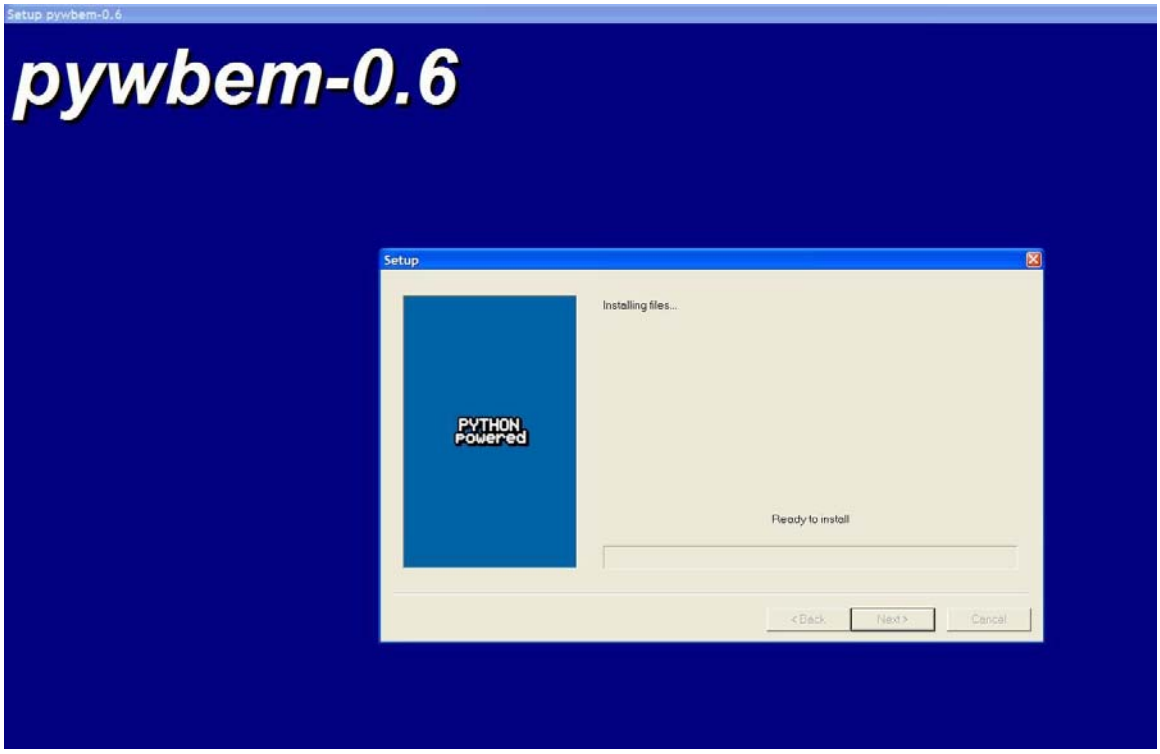

Python 2.5.2 Setup
Install Python 2.5.2
Please wait while the Installer installs Python 2.5.2. This may take several minutes.
Status:
< Back
Next >
Cancel

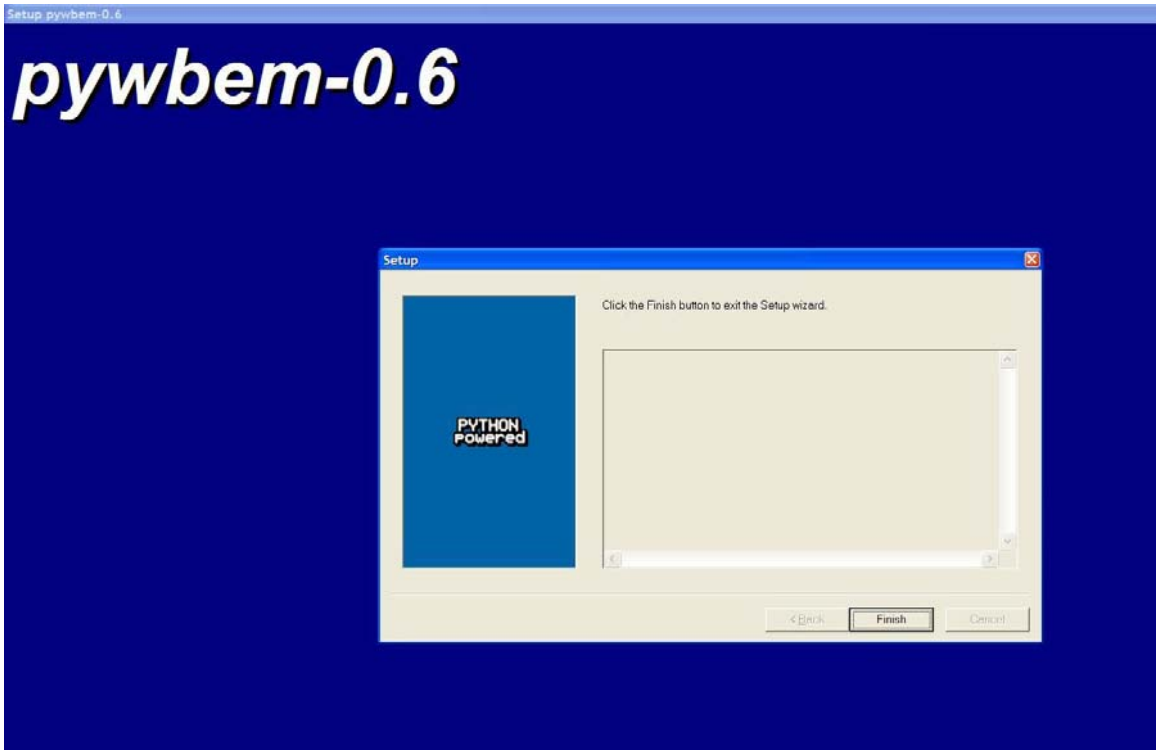
2. Install the PyWBEM module downloaded from the above site. For example if you have downloaded PyWBEM 0.6 install executable (PyWBEM-0.6.win32.exe). Install it by double clicking it from Windows explorer window.


Setup pywbem-0.6
pywbem-0.6
Setup
PYTHON powered
This Wizard will install pywbem on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
A pure-Python library for performing operations using the WBEM management protocol.
Author: Tim Potter
Author_email: tpot@hp.com
Description: Python WBEM client library
Name: pywbem
Url: http://pywbem.sf.net/
Version: 0.6
Built Fri Oct 26 14:24:17 2007 with distutils-2.5.1
< Back
Next >
Cancel


Setup pywbem-0.6
pywbem-0.6
Setup
PYTHON powered
Select python installation to use:
Python Version 2.5 (found in registry)
Python Directory:
C:\Python25\
Installation Directory:
C:\Python25\Lib\site-packages\
< Back
Next >
Cancel


Setup pywbem-0.6
pywbem-0.6
Setup
Click Next to begin the installation of pywbem. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.
PYTHON powered
Ready to install
< Back
Next >
Cancel


Setup pywbem-0.6
pywbem-0.6
Setup
Installing files...
PYTHON powered
Ready to install
< Back
Next >
Cancel



Linux Installation:

PyWBEM can be installed quite easily using the standard Python distutils that is part of the Python distribution. PyWBEM is built using the following shell command. Since PyWBEM is a pure-Python module, there isn't much that is done during the build process.

```
$ python setup.py build
running build
running build_py
creating build
creating build/lib
creating build/lib/PyWBEM
copying cim_xml.py -> build/lib/PyWBEM
[...]
```

PyWBEM is installed, as root, with the following shell command. This copies the PyWBEM source to the Python site-packages directory where it can be loaded by the interpreter.

```
# python setup.py install
running install
running build
running build_py
running install_lib
copying build/lib/PyWBEM/cim_xml.py -> /usr/lib/python [...]
[...]
```

If you do not have root access, or would like to install PyWBEM in a different directory, use the --install-lib option when installing.

```
$ python setup.py install --install-lib $HOME/python/lib
running install
running build
running build_py
```

```
running install_lib
creating /home/tpot/python/lib
creating /home/tpot/python/lib/PyWBEM
copying build/lib/PyWBEM/cim_xml.py -> /home/tpot/python/lib/PyWBEM
[...]
```

To test that PyWBEM is successfully installed, start up a Python interpreter and try to import the PyWBEM module.

```
$ python
Python 2.3.5 (#2, Mar 26 2005, 17:32:32)
[GCC 3.3.5 (Debian 1:3.3.5-12)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import PyWBEM
>>>
```

If you do not see any text after the import command, PyWBEM has been successfully installed.

3.4 Running the test script (test.py)

The following is the sample test program (test.py) to Enumerate WBEM provider class instances in any given namespace. For example this program can be used to enumerate any class listed in the HP WBEM Provider data sheets for VMware ESXi. The following web site provides PyWBEM API documentation, Tutorial and PyWBEM programming examples.
http://PyWBEM.sourceforge.net/documentation.shtml

```
import sys,pywbem

einCount=0
giCount=0

try:
    ip = sys.argv[1]
except IndexError, e:
    ip = "localhost"

try:
    unpw = sys.argv[2]
except IndexError, e:
    unpw = "root%wbem1"

try:
    nameSpace = sys.argv[3]
except IndexError, e:
    nameSpace = "root/hpq"

try:
    className = sys.argv[4]
except IndexError, e:
    className = "SMX_ComputerSystem"

classNames = [className,]
verbose=1


username, password = unpw.split('%')
conn = pywbem.WBEMConnection('https://' + ip,
                             (username, password),
                             default_namespace = nameSpace)

#instance tests
for className in classNames:
        instanceNames = conn.EnumerateInstanceNames( className)
        einCount=einCount + 1
        for instanceName in instanceNames:
                if verbose:
                        print
                        print 'Instance of: %s' % className
                        print 'InstanceName = %s' % instanceName
                        for key, value in instanceName.items():
                                print '   K[%s = %s]' % (key,value)
                instance = conn.GetInstance( instanceName)
                if verbose:
                        for name,value in instance.items():
                                print '   %s = %s' % (name,value)
                giCount = giCount + 1
```

```
C:\tests>test.py 10.1.1.15 root%mypass root/hpq CIM_processor
```

The arguments for the above sample script are the following.
Arg 1.) IP address
Arg 2.) Username and password separated be % delimiter. (for example root%mypassword)
Arg 3.) WBEM Namespace (for example root/hpq or root/cimv2)
Arg 4.) WBEM class. In the above example we used CIM_Processor class.

Here is the sample output of the above test program.

```
C:\tests>test.py 10.1.1.15 root%mypass root/hpq CIM_processor

Instance of: CIM_processor
InstanceName =
root/hpq:SMX_Processor.CreationClassName="SMX_Processor",SystemName="dl380root.
americas.hpqcorp.net",DeviceID="Proc
1",SystemCreationClassName="SMX_ComputerSystem"
   K[CreationClassName = SMX_Processor]
   K[SystemName = dl380root.americas.hpqcorp.net]
   K[DeviceID = Proc 1]
   K[SystemCreationClassName = SMX_ComputerSystem]
   RequestedState = 12
   HealthState = 5
   MaxQuiesceTime = None
   StatusDescriptions = [u'OK']
   Family = 179
   Characteristics = [3L, 2L]
   LoadPercentage = None
   OtherEnabledState = None
   SystemName = dl380root.americas.hpqcorp.net
   Stepping = 11
   CreationClassName = SMX_Processor
   ErrorDescription = None
   NumberOfEnabledCores = 2
   Availability = None
   OtherIdentifyingInfo = None
   PowerManagementSupported = None
   UpgradeMethod = None
   SystemCreationClassName = SMX_ComputerSystem
   ErrorCleared = None
   OperationalStatus = [2L]
   Role = Central Processor
   OtherFamilyDescription = None
   Status = None
   Description = Intel(R) Xeon(TM) 3GHz (x86 Family 179 Model 15 Stepping 11)
   InstallDate = None
   EnabledDefault = 2
   EnabledState = 2
   AdditionalAvailability = None
   TimeOfLastStateChange = None
   ExternalBusClockSpeed = 1333
   StatusInfo = None
   DataWidth = 64
   UniqueID = Proc 1
   CurrentClockSpeed = 3000
   PowerOnHours = None
   ElementName = Processor in socket 1
   Name = Processor in socket 1
   TotalPowerOnHours = None
   AddressWidth = 64
   LocationIndicator = None
   Caption = Processor in socket 1
```

```
   MaxClockSpeed = 4800
   DeviceID = Proc 1
   PowerManagementCapabilities = None
   LastErrorCode = None
   IdentifyingDescriptions = None
   CPUStatus = 1

Instance of: CIM_processor
InstanceName =
root/hpq:SMX_Processor.CreationClassName="SMX_Processor",SystemName="dl380root.
americas.hpqcorp.net",DeviceID="Proc
2",SystemCreationClassName="SMX_ComputerSystem"
   K[CreationClassName = SMX_Processor]
   K[SystemName = dl380root.americas.hpqcorp.net]
   K[DeviceID = Proc 2]
   K[SystemCreationClassName = SMX_ComputerSystem]
   RequestedState = 12
   HealthState = 5
   MaxQuiesceTime = None
   StatusDescriptions = [u'OK']
   Family = 179
   Characteristics = [3L, 2L]
   LoadPercentage = None
   OtherEnabledState = None
   SystemName = dl380root.americas.hpqcorp.net
   Stepping = 11
   CreationClassName = SMX_Processor
   ErrorDescription = None
   NumberOfEnabledCores = 2
   Availability = None
   OtherIdentifyingInfo = None
   PowerManagementSupported = None
   UpgradeMethod = None
   SystemCreationClassName = SMX_ComputerSystem
   ErrorCleared = None
   OperationalStatus = [2L]
   Role = Central Processor
   OtherFamilyDescription = None
   Status = None
   Description = Intel(R) Xeon(TM) 3GHz (x86 Family 179 Model 15 Stepping 11)
   InstallDate = None
   EnabledDefault = 2
   EnabledState = 2
   AdditionalAvailability = None
   TimeOfLastStateChange = None
   ExternalBusClockSpeed = 1333
   StatusInfo = None
   DataWidth = 64
   UniqueID = Proc 2
   CurrentClockSpeed = 3000
   PowerOnHours = None
   ElementName = Processor in socket 2
   Name = Processor in socket 2
   TotalPowerOnHours = None
   AddressWidth = 64
   LocationIndicator = None
   Caption = Processor in socket 2
   MaxClockSpeed = 4800
   DeviceID = Proc 2
   PowerManagementCapabilities = None
   LastErrorCode = None
   IdentifyingDescriptions = None
   CPUStatus = 4
```

Legal Notice