WBEM Based Management in Linux

A Dell Technical White Paper

February 2011

Praveen Kumar Paladugu (praveen_paladugu@dell.com)
Dell Linux Engineering Team

March 2010

Contents

Figures

Introduction

The more datacenters grow in size and heterogeneity, the harder it gets for the administrators to manage all the hardware. As most of the enterprise class datacenter is heterogeneous in nature, the number of tools required to manage the servers swell beyond control, unless some open standards are adopted by the manufacturers/vendors.

The Total Cost of Ownership (TCO) in an IT environment depends highly on the effectiveness of the management tools deployed. The more diverse tools required for managing systems from different vendors/manufacturers, the higher TCO will be. DMTF (Distributed Management Task Force) targets to reduce the TCO by defining open standards which can be adopted by all the vendors and there by allow a common set of tools be used for management of all the systems/devices.

In this paper, we will primarily look at the IBM initiated open-source implementation of the WBEM, CIM standards and how to use them. This whitepaper is written to serve as a primer for users configuring/working with CIM based management tools on Linux systems. We will be concentrating mostly on "SBLIM sfcb" and "openwsman" implementations as the CIMOM (CIM Object Manager) and WS-Management (Web Services Management Specification) protocol. These implementations are adopted by many Linux distributions like RHEL, SLES, and Ubuntu etc. Also VMware ESX uses SFCB as the core CIMOM to communicate the management information of the ESX Host. All the details (or commands) mentioned below should work as such, on SLES and RHEL distributions.

Common Information Model (CIM)

CIM is an open standard that defines how the managed elements in an IT environment are represented and the relation between these elements. CIM is an extensible data model for logically organizing management data in a consistent, unified manner in a managed environment. CIM attempts to unify and extend the current instrumentation standards like IPMI, SNMP and CMPI etc. CIM primarily allows the information from multiple sources to be unified into a single object-oriented model. The guidelines for mapping data from other instrumentation standards (like SNMP, IPMI) to CIM objects have already been defined. CIM being an open standard obviates the need for the admins to learn any vendor/manufacturer specific data models. Once the instrumentation providers for the hardware/software components are installed, all the related objects can be handled similarly. CIM model has primarily two parts: a specification and a schema.

CIM specification:

The specification defines the syntax, naming convention and the properties of devices. The specification primarily defines the syntax language called MOF (Managed Object Format) which is based on IDL (Interface Definition Language). The specification also defines the mapping to other management standards like SNMP and IPMI.

CIM Schema:

The schema primarily captures notions that are common to a particular management area, but are independent of the particular technology and implementation. Schema, primarily defines a specific set of base classes and associations between these classes. This set serves as the base for the managed elements in the IT environments. CIM schema defines most of the common elements in an IT

environment like servers, Operating systems, network, services, and storage etc. Vendors can extend these common elements and add their customizations or extensions.

The management data as well as the class definitions are expressed in ASCII based MOF (Managed Object Format) file. The CIM clients talk to the remote CIMOM using http or https protocol. The CIM operations over HTTP/S use xmlCIM bindings to transport CIM information across the network.

To deploy WBEM based system management tools five components are to be considered:

1) The managed device (In our case a Linux Host)

2) A CIM Object Manager and information repository for data (SFCB CIMOM)

3) Providers (instrumentation agents that provides data about the managed device/component)

4) A management client (an application using that data either for reporting or taking a scripted action)

5) Finally the communication between these elements (WBEM).

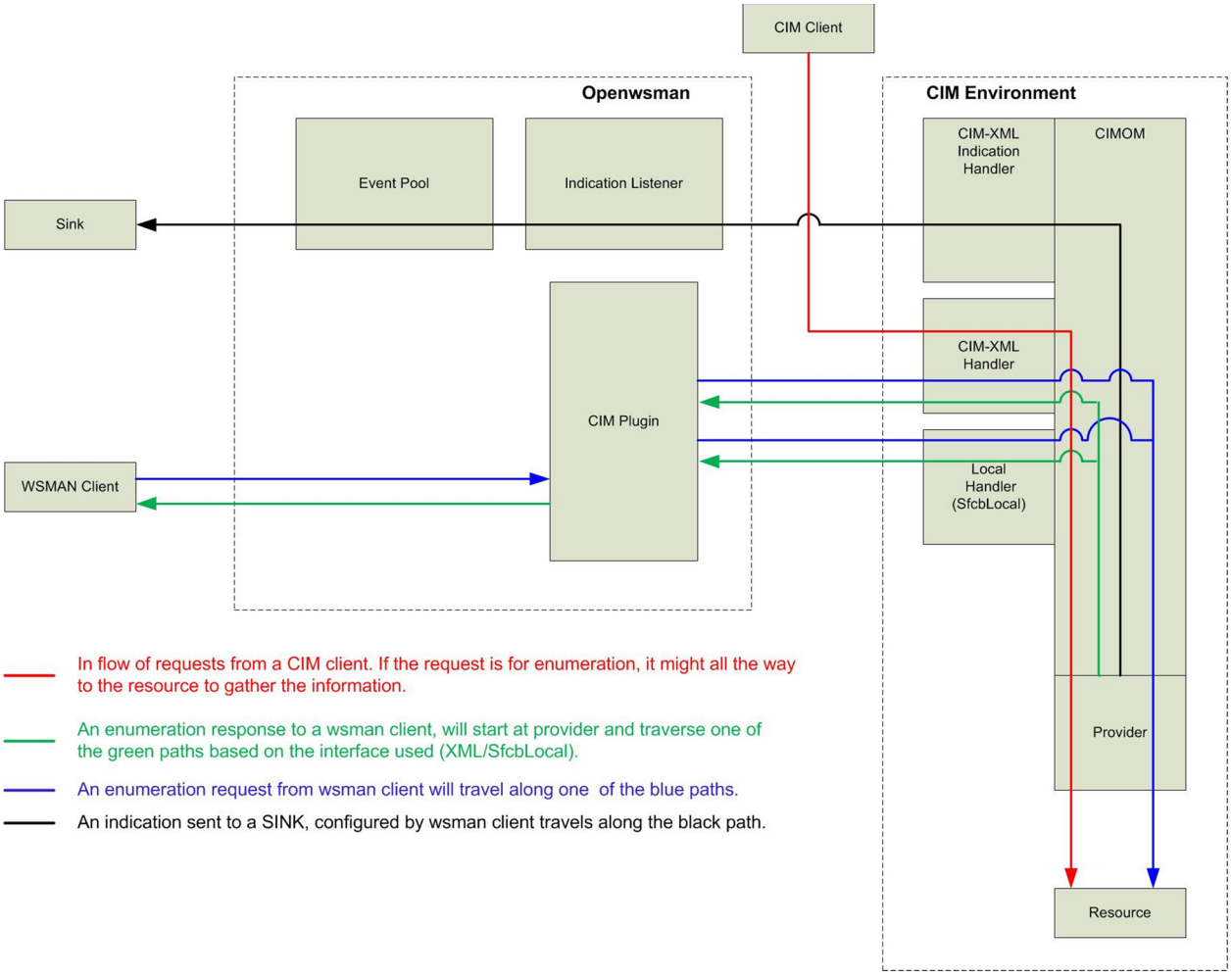

Figure 1: Components of WBEM Management.

SFCB CIMOM

SFCB is a lightweight CIMOM (CIM Object Manager) that responds to client requests and/or performs system management tasks as required. SFCB is at the core of the CIMOM which is extended by installing required providers and adapters. Along with the support for basic CIM operations like GetClass, GetInstance, CreateInstance, DeleteInstance, and EnumerateInstance etc, SFCB also supports CIM process and lifecycle indications.

SFCB is part of the SBLIM project hosted at http://sblim.sourceforge.net/. SFCB CIMOM only supports the Open Group's CMPI provider interface. CMPI defines a common standard to interface the providers with the management brokers (SFCB CIMOM in this context). The CMPI interface is designed to relieve provider writers of all the CIMOMs' implementation details and a CMPI provider is expected to work seamlessly with various CIMOMs. Other implementations of CIMOMs include SNIA, tog-pegasus, openwbem etc.

The CIM schema (http://dmtf.org/standards/cim) is used as the base for the class hierarchy used in SFCB. The CIM schema is available as cim-schema package in most of the standard Linux distributions and is installed along with the SFCB CIMOM.

SFCB stores all the CIM definition information from all the installed providers in a repository. The CIM repository is primarily a collection of Class definitions and any dynamic data required to handle the incoming client requests. There is one repository defined for each namespace and the commonly used namespaces are root/cimv2 and root/interop. Typically, the "root/cimv2" is used for all the instrumentation information and the "root/interop" namespace is used to handle indications. The default directories used by the repositories of these two namespaces are:/var/lib/sfcb/registration/repository/root/cimv2 and /var/lib/sfcb/registration/repository/root/interop respectively.

NOTE: The defaults mentioned here are from the rpms/packages in SLES and RHEL distributions. These may not apply when directly compiled from sources.

Make sure sblim-sfcb, cim-schema and a provider package (sblim-cmpi-*) is installed on the system. The first provider installed will register some classes with SFCB and the CIM repository will be populated.

For the providers to work properly, the related libraries have to be placed in specific directories. Also the providers have to register themselves with the CIMOM. All these details are handled by the rpm/deb packaging in various distributions. For example, the sblim-cmpi-base package has the following command, in the post install steps, which registers the provider's Classes to the CIMOM (sfcb) after installation:

```
/usr/share/sblim-cmpi-base/provider-register.sh \
      -r /usr/share/sblim-cmpi-base/Linux_BaseIndication.registration -m
/usr/share/sblim-cmpi-base/Linux_Base.mof /usr/share/sblim-cmpi-
base/Linux_BaseIndication.mof
```

Staging of the MOFs is part of the registration process. If the registration of the provider completes without any issues, the provider's mof files will be staged (or copied) at /var/lib/sfcb/stage/mofs/root/cimv2 (root/cimv2 is the default namespace) and the related registration files will be created at /var/lib/sfcb/stage/regs. After the staging, the sfcbrepos command will be ran by the registration script to compile the classes into the SFCB's repository.

The SFCB CIMOM by default is configured to listen on https (5989) port. These settings can be changed in the /etc/sfcb/sfcb.cfg file. A configuration file for SFCB which listens on http, https ports and doesn't require any authentication is provided in the appendix. Following is a simple command to list all class names compiled into the root/cimv2 namespace's repository.

```
linux## wbemcli ecn http://username:password@hostname:root/cimv2
```

* wbemcli is a simple client that can talk to sfcb server.
* ecn – function to list all the classnames complied into repository
* http –what protocol to use to talk to the SFCB CIMOM.
* root/cimv2 -- The default namespace used it root/cimv2. If vendor-specific providers are installed they can be staged in a different namespace.

Among the basic operations supported by CIMOM is enumeration. For example:

```
linux## wbemcli ei http://hostname:5988/root/cimv2:Linux_Processor
```

The above command will enumerate all the processors installed in the system. This enumeration instance (ei) operation can be run on all the classes that are registered with type "instance".

When the classes are defined, some of the properties are identified as "Key" properties. The properties with Key as qualifier are able to uniquely identify a particular instance of a class. For example, the class *Linux_Processor* has the properties *SystemCreationClassName*, *SystemName*, *CreationClassName*, and *DeviceID* as Key properties. This implies a single processor can be identified using these key properties. Following is an example of such a command:

```
linux## wbemcli gi
'http://10.208.46.61:5988/root/cimv2:Linux_Processor.SystemCreationClassName=
"Linux_ComputerSystem",SystemName="vmware-
share.us.dell.com",CreationClassName="Linux_Processor",DeviceID="0"'
```

The above command is requesting information of a processor, which has the following name-value pairs:

SystemCreationClassName="Linux_ComputerSystem"
SystemName="vmware-share.us.dell.com"
CreationClassName="Linux_Processor"
DeviceID="0"

Please note some of the classes have multiple properties defined as Key properties. So, unless all the Key properties values are given to the command, you cannot identify a specific instance of a class. As mentioned above, the *Linux_Processor* class has four Key properties, so a request of the following kind will not work.

```
linux## wbemcli gi
'http://10.208.46.61:5988/root/cimv2:Linux_Processor.SystemCreationClassName=
"Linux_ComputerSystem",SystemName="vmware-
share.us.dell.com",CreationClassName="Linux_Processor"'
```

To identify the Key properties of a class, enumerate instance names (ein) command can be used. For example the following command:

```
linux## wbemcli ein http://192.168.46.61:5988/root/cimv2:Linux_Processor

192.168.46.61:5988/root/cimv2:Linux_Processor.SystemCreationClassName="Linux_
ComputerSystem",SystemName="vmware-
share.us.dell.com",CreationClassName="Linux_Processor",DeviceID="0"
192.168.46.61:5988/root/cimv2:Linux_Processor.SystemCreationClassName="Linux_
ComputerSystem",SystemName="vmware-
share.us.dell.com",CreationClassName="Linux_Processor",DeviceID="1"
```

returns output as shown above, indicating that SystemCreationClassName, SystemName, CreationClassName and DeviceID properties are key properties as mentioned above.

CIMOM's other interfaces

The classes registered to the CIMOM's repository can be of different types, primarily differentiated by the type of the operations allowed by the class. The type of a class can be verified in the SomeName.reg file in /var/lib/sfcb/stage/regs/ directory in which the class name is listed. For example the Linux_Base.reg (registered by sblim-cmpi-base provider) file has the following lines:

```
[Linux_Porocessor]
        provider:       OSBase_ProcessorProvider
        location:       compiOSBase_ProcessorProvider
        type:           instance
        namespace:      root/cimv2
```

The above lines indicate that the name of the class is Linux_Processor; the name of the "provider" is OSBase_ProcessorProvider. The value of the "location" indicates the name of the provider library (/usr/lib/cmpi/libOSBase_ProcessorProvider.so in this case). The "type" indicates the operations that can be performed on an instance of this class. The possible values are:

instance: objects can be created and enumerated

association: Objects of this class will define relationship between two or more other objects.

method: methods of this class can be called extrinsically

indication: This class can generate indications.

The type defined for a class can be any combination of the above mentioned types. The operations GetInstance, CreateInstance, EnumerateInstance etc can be run only on classes of type "instance".

Method type

The classes registered as type "method" allow the methods defined in the class to be called remotely. For example the sblim-cmpi-syslog provider exposes some functions of the Syslog_Service class. The registration file for the sblim-cmpi-syslog provider has the following lines along with other lines:

```
[Syslog_Service]
  provider: Syslog_Service
  location: Syslog_Service
  type: instance method
  namespace: root/cimv2
```

The above lines indicate that an instance of Syslog_Service class also exposes a method interface. This implies the methods defined in the Syslog_Service class can be called. The definition of the Syslog_Service class can be checked with the following command:

```
linux## wbemcli gcd http://10.208.46.61:5988/root/cimv2:Syslog_Service

<CLASS NAME="Syslog_Service" SUPERCLASS="CIM_Service">
<PROPERTY NAME="Started" TYPE="boolean">
</PROPERTY>
<METHOD NAME="RequestStateChange" TYPE="uint32">
<PARAMETER NAME="RequestedState" TYPE="uint16">
</PARAMETER>
<PARAMETER.REFERENCE NAME="Job" REFERENCECLASS="CIM_ConcreteJob">
</PARAMETER.REFERENCE>
<PARAMETER NAME="TimeoutPeriod" TYPE="datetime">
</PARAMETER>
</METHOD>
<METHOD NAME="StartService" TYPE="uint32">
</METHOD>
<METHOD NAME="StopService" TYPE="uint32">
</METHOD>
<METHOD NAME="RestartService" TYPE="uint32">
</METHOD>
<METHOD NAME="ReloadService" TYPE="uint32">
</METHOD>
<METHOD NAME="CondRestartService" TYPE="uint32">
</METHOD>
<METHOD NAME="TryRestartService" TYPE="uint32">
</METHOD>
<METHOD NAME="ForceReloadService" TYPE="uint32">
</METHOD>
<METHOD NAME="ProbeService" TYPE="uint32">
</METHOD>
</CLASS>
```

According to the above class definition, "StartService" and "StopService" are among the methods that can be called on the Linux server via CIMOM. In order to call the "StartService" function, run the following command:

```
linux## wbemcm
'http://10.208.46.61:5988/root/cimv2:Syslog_Service.CreationClassName="Syslog
_Service",SystemCreationClassName="CIM_UnitaryComputerSystem",SystemName="vmw
are-share.us.dell.com",Name="syslog"'       StartService
```

TIP: Please don't forget the single quotations (')

The first argument here would be the object (in the right namespace) along with its key values. The second part of the "wbemcm" command is the name of the method to call. Since the "StartService" method (according to the class definition) doesn't need any parameters, no parameters. If a method like "RequestStateChange" has to be called, the values of the parameters "RequestedState" and "TimeoutPeriod" have to be provided.

Association

An association defines the relationship between two or more classes or objects. **Component associations** establish “part of” relationship between managed elements and the **Dependency associations** describe functional dependencies (one object cannot function with the other) or existence dependencies (one object cannot exist without the other) between managed elements.

For example the *CIM_USBPortOnHub* association describes an existence dependency between *CIM_USBPort* and *CIM_USBHub* classes. This class has two references defined. The reference *CIM_USBHub* is defined as an antecedent and *CIM_USBPort* is defined as a dependent. This association implies an usb port doesn’t exist without its hub.

Another example would be *CIM_ChassisInRack* association which makes a containing relationship between a *CIM_Rack* and *CIM_Chassis*. The Groupcomponent is CIM_Rack and the PartComponent is CIM_Chassis, indicating that a chassis should be part of a Rack.

Indication

An indication represents the occurrence of an event of interest. An event could be, a new device attached to the server or a new service was started in the system or the CPU utilization has reached a beyond a defined threshold etc.

Instances of indication classes are transient and cannot be obtained using the standard CIM Operations like enumeration, get instance etc. They can only be received by subscribing to them. In order to receive indications of interest, a Filter, a Handler and a Subscription instance have to be defined.

Filter (CIM_IndicationFilter): As the name suggests, a filter is used to filter the indications of interest from all the indications received by the CIMOM. For example, a filter could request any indications of CPU and memory usage going beyond a defined threshold. This filter will not report anything about any h/w errors or any changes in the services of the system because it doesn't meet the criterion defined in the filter.

Handler (CIM_IndicationHandlerCIMXML): This is the component which will receive/handle the incoming indications. The handler can either just log all the incoming indications or take a scripted action based on the indications received.

Subscription (CIM_IndicationSubscription): The subscription object primarily defines an association between a Filter and a Handler instances. A Filter instance and a Handler are the references defined for a subscription. Additionally, the Subscription instance also defines other properties of a subscription's life like:

a) How many times should an indication instance be sent to the Handler
b) The Start time for the subscription
c) Duration of the subscription etc

Indications are broadly categorized into two types:

a) Life Cycle Indications
b) Process Indications

The *CIM_InstIndcation* and *CIM_ClassIndication* are the base classes for Life Cycle Indications and the *CIM_PorcessIndication* class is the base class for Process Indications.

i) *CIM_InstIndications*: Tracks changes in objects. This implies any creation, deletion or modification of objects in the CIM namespace along with any explicit calls to any extrinsic functions of the objects.

ii) *CIM_ClassIndications*: Tracks Class life cycle events. Creation, deletion or modification of any classes.

iii) *CIM_ProcessIndication*: Tracks any alert notifications from individual objects. Along with handling low-level instrumentation alerts, this class also provides SNMP Traps, DMI alerts and TMN events.

Let us walk through an example here. The sblim-cmpi-base package has an indication provider *Linux_OperatingSystemIndication*. This can be verifiedverfied by the *.reg files staged at /var/lib/sfcb/stage/regs/Linux_BaseIndication.reg file. The class *Linux_OperatingSystemIndication* is defined of type “indication” in the registration file. An instance of this indication is created every time when the value of *OperationalStatus* changes in the instance of *Linux_OperatingSystem* class. The value of *OperationalStatus* variable is set based on the state of the system's CPU. If the CPU usage goes above 90%, this value is set to 4 and if it below 90%, it is set to 2.

The current state of the CPU can be verified by:

```
linux## wbemcli -nl ei http://localhost:5988/root/cimv2:Linux_OperatingSystem
| grep OperationalStatus
```

```
-OperationalStatus=2
```

The Filter, Handler and Subscriptions instances will be created using the SFCBCreateFilter.localhost.xml, SFCBCreateHandler.localhost.xml and SFCBCreateSubscription.localhost.xml files which are part of the sblim-cmpi-base package's sources. These XML files have the XML payload to be sent to the SFCB CIMOM to create a Filter, Handler and a Subscription respectively. wbemcat command can be used as follows to send the xml content to CIMOM.

```
linux## wbemcat -t http -h localhost -p 5988 -u root -pwd password
SFCBCreateFilter.localhost.xml
```

```
linux## wbemcat -t http -h localhost -p 5988 -u root -pwd password
SFCBCreateHandler.localhost.xml
```

```
linux## wbemcat -t http -h localhost -p 5988 -u root -pwd password
SFCBCreateSubscription.localhost.xml
```

The Filter, Handler and Subscription instances are going to be created in the root/interop namespace. The following command can be used to check if they are created properly.

For Filter:

```
linux## wbemcli  ei http://localhost:5988/root/interop:CIM_IndicationFilter

localhost:5988/root/interop:CIM_IndicationFilter.SystemCreationClassName="CIM
_ComputerSystem",SystemName="localhost.localdomain",CreationClassName="CIM_In
dicationFilter",Name="OperatingSystemFilter0"
TemplateVariable=,QueryLanguage="WQL",Query="SELECT * FROM
CIM_InstModification",IndividualSubscriptionSupported=TRUE,SourceNamespaces="
root/cimv2",SourceNamespace="root/cimv2",Name="OperatingSystemFilter0",Creati
onClassName="CIM_IndicationFilter",SystemName="localhost.localdomain",SystemC
reationClassName="CIM_ComputerSystem",InstanceID=,Caption=,Description=,Eleme
ntName=,Generation=
```

For handler:

```
linux## wbemcli  ei
http://localhost:5988/root/interop:CIM_IndicationHandlerCIMXML

localhost:5988/root/interop:CIM_IndicationHandlerCIMXML.SystemCreationClassNa
me="CIM_ComputerSystem",SystemName="localhost.localdomain",CreationClassName=
"CIM_IndicationHandlerCIMXML",Name="OperatingSystemHandler0"
Destination="file:///tmp/SFCB_OS_Listener.txt",OtherPersistenceType=,Persiste
nceType=2,OtherProtocol=,Protocol=2,Name="OperatingSystemHandler0",CreationCl
assName="CIM_IndicationHandlerCIMXML",SystemName="localhost.localdomain",Syst
emCreationClassName="CIM_ComputerSystem",InstanceID=,Caption=,Description=,El
ementName=,Generation=,Owner=
```

For subscription:

```
linux## wbemcli ei
http://localhost:5988/root/interop:CIM_IndicationSubscription

localhost:5988/root/interop:SFCB_IndicationSubscription.Handler=root/interop:
CIM_IndicationHandlerCIMXML.SystemCreationClassName="CIM_ComputerSystem",Syst
emName="localhost.localdomain",CreationClassName="CIM_IndicationHandlerCIMXML
",Name="OperatingSystemHandler0",Filter=root/interop:CIM_IndicationFilter.Sys
temCreationClassName="CIM_ComputerSystem",SystemName="localhost.localdomain",
CreationClassName="CIM_IndicationFilter",Name="OperatingSystemFilter0"
Handler=root/interop:CIM_IndicationHandlerCIMXML.SystemCreationClassName="CIM
_ComputerSystem",SystemName="localhost.localdomain",CreationClassName="CIM_In
dicationHandlerCIMXML",Name="OperatingSystemHandler0",Filter=root/interop:CIM
_IndicationFilter.SystemCreationClassName="CIM_ComputerSystem",SystemName="lo
calhost.localdomain",CreationClassName="CIM_IndicationFilter",Name="Operating
SystemFilter0",OnFatalErrorPolicy=,OtherOnFatalErrorPolicy=,FailureTriggerTim
eInterval=,SubscriptionState=2,OtherSubscriptionState=,TimeOfLastStateChange=
,SubscriptionDuration=,SubscriptionStartTime=20110101165017.656706-
360,SubscriptionTimeRemaining=,RepeatNotificationPolicy=,OtherRepeatNotificat
ionPolicy=,RepeatNotificationInterval=,RepeatNotificationGap=,RepeatNotificat
ionCount=,AlertOnStateChange=FALSE,LastIndicationIdentifier=,LastIndicationPr
oductionDateTime=,SubscriptionInfo=
```

The Handler and Filter references in the subscription instance are defined as highlighted in the above output. Please note, the Handler and Filter instances have to be created before a Subscription instance can be created.

After all the three objects are created, increase the CPU usage to more than 90%, and you will notice the following content appearing at the destination(**/tmp/SFCB_OS_Listener.txt**) of the Handler. Content similar to the one below will be seen in the destination whenever an indication is created.

```
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
<MESSAGE ID="1" PROTOCOLVERSION="1.0">
<SIMPLEEXPREQ>
<EXPMETHODCALL NAME="ExportIndication">
<EXPPARAMVALUE NAME="NewIndication">
<INSTANCE CLASSNAME="CIM_InstModification">
<PROPERTY NAME="PreviousInstance" TYPE="string">
</PROPERTY>
<PROPERTY NAME="IndicationFilterName" TYPE="string">
</PROPERTY>
<PROPERTY NAME="OtherSeverity" TYPE="string">
</PROPERTY>
<PROPERTY NAME="PerceivedSeverity" TYPE="uint16">
</PROPERTY>
<PROPERTY NAME="IndicationTime" TYPE="datetime">
<VALUE>20110101170830.361880-360</VALUE>
</PROPERTY>
<PROPERTY.ARRAY NAME="CorrelatedIndications" TYPE="string">
<VALUE.ARRAY>
</VALUE.ARRAY>
</PROPERTY.ARRAY>
<PROPERTY NAME="IndicationIdentifier" TYPE="string">
<VALUE>CIM_InstModification</VALUE>
</PROPERTY>
<PROPERTY NAME="SourceInstance" TYPE="string">
</PROPERTY>
<PROPERTY NAME="SourceInstanceModelPath" TYPE="string">
</PROPERTY>
<PROPERTY NAME="SourceInstanceHost" TYPE="string">
</PROPERTY>
</INSTANCE>
</EXPPARAMVALUE>
</EXPMETHODCALL>
</SIMPLEEXPREQ>
</MESSAGE>
</CIM>
```

In order to delete the Subscription, Handler and Filter instances, the contents of SFCBDeleteSubscription.localhost.xml, SFCBDeleteHandler.localhost.xml, SFCBDeleteFilter.localhost.xml files can be used.

Troubleshooting and Debugging Tips

i) The wbemcli command has an option (-dx) to print out the XML payload that is sent to the server and received from the server. An example of one such command is shown below:

```
linux## wbemcli -dx  ei http://10.208.46.61:5988/root/cimv2:Linux_Processor
```

ii) The sfcb service also has some tracing options that can be configured either in the configuration file (sfcb.cfg) or using the -t option while starting the service. The list of components that can be traced are returned for "sfcbd -t ?" command. If multiple components have to be traced, the final mask passed to CIMOM should be OR of the masks of the individual components.

iii) If you run into any errors with the following message, it is most likely because of the way shell handles the arguments to the commands. The best way to get around this problem would be enclose the arguments of wbemcli in single quotes (‘).

*

* wbemcli: Parse Exception: Invalid name-value pair

*

Openwsman

Web Services for Management (WS-Management/WSMAN) is a SOAP (Simple Object Access Protocol) based protocol defined by DTMF to seamlessly manage servers, services, devices, and applications across multiple hardware vendors and multiple operating systems. WS-Management provides a common way for systems to access and exchange information in an IT infrastructure from multiple and diverse sources. WS-Management protocol provides a standard for constructing XML messages using various Web Service standards like WS-Transfer, WS-Addressing, WS-Enumeration, WS-Eventing. The primary goal of this protocol is provide interoperability and consistency across different types of devices or components (operating system, firmware etc). openwsman (http://sourceforge.net/projects/openwsman/) project is an open source implementation of WS-Management specification enabling in-band management of linux/unix platforms. openwsman supports the CIM data model and the conversion of the CIM data into WS-Man resources happens on the fly, following the CIM bindings defined by DMTF.

WS-Management supports a common set of operations that are central to all systems. The supported operations include:

Get, put, create, delete: to handle individual managed resources.

Enumerate**:** To enumerate all the instances of a managed resource.

Subscribe, Unsubscribe: Capture the events sent by a managed resource.

Discover**:** For discovering the managed resources and navigating between them.

Invoke: To Invoke a method on a managed resource with defined parameters and gather the output.

openwsman-server is the "wsman server" package that has to be installed. Since openwsman in-turn talks to CIMOM for all the management requests, the configuration file (/etc/openwsman/openwsman.conf) lets the user to configure the details of the CIMOM. If an instance of sfcb (CIMOM) service is running on the server where "wsman server" is running, the "SfcbLocal" interface can be used to talk to the local sfcb server. This can be done by setting the value of cim_client_frontend to "SfcbLocal". If the wsman server has to talk to a remote sfcb(CIMOM), the value of cim_client_frontend has to be set to "XML" indicating that the communication with the sfcb (CIMOM) will be over http and the values of host and port in [cim] section have to be set.

After the wsman server is configured properly, the wsman (wsmancli package) command line client utility can be used to send requests to the server. A working copy of the configuration file for wsman is available in the appendix. This configuration will get the wsman server to listen on 443 by default and use the SfcbLocal interface to talk to the local SFCB CIMOM.

The following command can be used to get the version details of the wsman protocol supported by the remote server:

```
linux## wsman identify --port 5986  --cacert=servercert.pem    --username
root -p Password1 -h 10.208.46.61

<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsmid="http://schemas.dmtf.org/wbem/wsman/identity/1/wsmanidentity.xsd"
>
  <s:Header/>
  <s:Body>
    <wsmid:IdentifyResponse>

<wsmid:ProtocolVersion>http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd</wsmid:
ProtocolVersion>
      <wsmid:ProductVendor>Openwsman Project</wsmid:ProductVendor>
      <wsmid:ProductVersion>2.2.3.9</wsmid:ProductVersion>
    </wsmid:IdentifyResponse>
  </s:Body>
</s:Envelope>
```

This will return:

* The WS-Management protocol version. For example, "http://schemas.dmtf.org/wbem/wsman/1/wsman".

* The product vendor, (Openwsman Project)

* The product version (2.2.3.9)

In the openwsman.conf file, a couple of fake namespaces have been configured. The namespace to be used while talking to Linux systems is "http://sblim.sf.net/wbem/wscim/1/cim-schema/2". For example, the command to enumerate the processors on a linux host is:

```
linux## wsman enumerate http://sblim.sf.net/wbem/wscim/1/cim-
schema/2/Linux_CSProcessor  --port 443  --cacert=servercert.pem   --username
root -p Password1 -h 10.208.46.61  --namespace=root/cimv2
```

*wsman -- is the dcomman line utility which is part of the wsmancli package.
*enumerate -- calling the enumerate method on the wsman server to enumerate all the instance of Linux_CSProcessor.
*http://sblim.sf.net/wbem/wscim/1/cim-schema/2 -- Need to use this fake namespace on a Linux system.
*Linux_CSProcessor -- the class to enumerate.
*443 -- The port number on which the openwsman server is listening
*servercert.pem -- the CA certificate from the server
*root/cimv2 -- is the namespace to enumerate the class in, on the CIMOM(sfcb).

For the rest of the paper, we will go over some of the examples of the wsman commands. Similar to the get instance command to the CIMOM, wsman has the get operation which can be called as follows:

```
linux## wsman get 'http://sblim.sf.net/wbem/wscim/1/cim-
schema/2/Linux_Processor?SystemCreationClassName="Linux_ComputerSystem"&Syste
mName="linux-kjz7.site"&CreationClassName="Linux_Processor"&DeviceID="0"'  -u
user -p pass  --port=5986 --cacert=/etc/openwsman/servercert.pem   -h
localhost --namespace=root/cimv2
```

The get operation needs all the key properties listed just like for the get instance command to the CIMOM.

For creating a subscription to capture the indications of interest, the following command can be used:

```
linux## wsman subscribe 'http://schemas.dmtf.org/wbem/wscim/1/*' -x "SELECT *
FROM CIM_ProcessIndication" -D
'http://schemas.microsoft.com/wbem/wsman/1/WQL' -Z
'http://127.0.0.1:80/eventsink'  --namespace=root/interop  -H 10 --username
root  --password Password1  -h localhost -G push -R --port 443 --
cacert=/etc/openwsman/servercert.pem
```

The output will have the following lines, along with other lines:

```
<wse:SubscribeResponse>
      <wse:SubscriptionManager>
        <wsa:Address>https://localhost:443/wsman</wsa:Address>
        <wsa:ReferenceParameters>
          <wse:Identifier>uuid:c2466b2c-9ab4-1ab4-8042-
5577ac565000</wse:Identifier>
        </wsa:ReferenceParameters>
      </wse:SubscriptionManager>
    </wse:SubscribeResponse>
```

The highlighted part in the above output will be the id which will be used to identify this subscription. The Filter, Handler and the Subscription objects created by the above command can be verified with the following commands:

```
linux## wbemcli ei
http://root:pass@localhost:5988/root/interop:CIM_IndicationFilter| grep
c2466b2c-9ab4-1ab4-8042-5577ac565000

linux## wbemcli ei
http://root:pass@localhost:5988/root/interop:CIM_IndicationHandlerCIMXML|
grep c2466b2c-9ab4-1ab4-8042-5577ac565000

linux## wbemcli ei
http://root:pass@localhost:5988/root/interop:CIM_IndicationSubscription| grep
c2466b2c-9ab4-1ab4-8042-5577ac565000
```

The above subscribe command will send the indications to the target *http://127.0.0.1:80/eventsink*. The indications at this location can be captured by running wseventsink command (available at https://build.opensuse.org/project/show?project=systemsmanagement%3Awbem).

In the above subscription the delivery mode (-G) was chosen to be *push*. This implies, the indication will be pushed to the target as soon as it appears. Pull is another possible mode.

The above subscription can be cancelled by running:

```
linux## wsman unsubscribe -i uuid: c2466b2c-9ab4-1ab4-8042-5577ac565000 --
namespace=root/interop -u root -p pass -h localhost
```

WMI

WMI (Windows Management Instrumentation) technology is Microsoft implementation of DMTF's WBEM initiative that extends CIM (Common Information Model) to represent managed objects in windows based management environments. WMI is available in Win ME, Windows 2000 and newer Windows versions.

Winrm client which is part of the WMI in windows, can be used to talk to a remote openwsman server.

Follow the instructions at http://www.openwsman.org/openwsman-users-guide/vista-winrm-over-openwsman-setup to configure the winrm client.

Client configuration:

```
C:\Users\Administrator>winrm set winrm/config/client/auth @{Basic="true"}
C:\Users\Administrator>winrm set winrm/config/client
@{AllowUnencrypted="true"}
C:\Users\Administrator>winrm set winrm/config/client
@{TrustedHosts="192.168.1.100"}
```

Server Configuration:

```
C:\Users\Administrator>winrm quickconfig
C:\Users\Administrator>winrm set winrm/config/service/auth @{Basic="true"}
C:\Users\Administrator>winrm set winrm/config/service
@{AllowUnencrypted="true"}
```

To enumerate the processors on a remote openwsman server, the following command can be used:

```
C:\Users\Administrator>winrm enumerate http://sblim.sf.net/wbem/wscim/1/cim-
schema/2/Linux_CSProcessor -username:root -password:Password1 -
r:172.17.7.240:8889 -auth:basic -format:#text

Linux_CSProcessor
    GroupComponent
        Address =
http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
        ReferenceParameters
            ResourceURI = http://sblim.sf.net/wbem/wscim/1/cim-
schema/2/Linux_ComputerSystem
            SelectorSet
                Selector: CreationClassName = Linux_ComputerSystem, Name =
linux-cv22, __cimnamespace = root/cimv2
    PartComponent
        Address =
http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
        ReferenceParameters
            ResourceURI = http://sblim.sf.net/wbem/wscim/1/cim-
schema/2/Linux_Processor
            SelectorSet
                Selector: SystemCreationClassName = Linux_ComputerSystem,
SystemName = linux-cv22, CreationClassName = Linux_Processor, DeviceID = 0,
__cimnamespace = root/cimv2
```

The above winrm command will gather the same information as the following wsman client command (indicating there is only one processor in the system):

```
linux## wsman  enumerate http://sblim.sf.net/wbem/wscim/1/cim-
schema/2/Linux_CSProcessor  --port 8889   --username root -p Password1 -h
10.208.46.57   --namespace root/cimv2


<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration">
  <s:Header>

<wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:
To>

<wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/enumeration/EnumerateRespon
se</wsa:Action>
    <wsa:RelatesTo>uuid:72e4207f-964c-164c-8002-159967e82400</wsa:RelatesTo>
    <wsa:MessageID>uuid:9c56b0db-9647-1647-8006-0000b5565000</wsa:MessageID>
  </s:Header>
  <s:Body>
    <wsen:EnumerateResponse>
      <wsen:EnumerationContext>9c54b717-9647-1647-8005-
0000b5565000</wsen:EnumerationContext>
    </wsen:EnumerateResponse>
  </s:Body>
```

```
</s:Envelope>
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
xmlns:n1="http://sblim.sf.net/wbem/wscim/1/cim-schema/2/Linux_CSProcessor"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
  <s:Header>

<wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:
To>

<wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/enumeration/PullResponse</w
sa:Action>
    <wsa:RelatesTo>uuid:72ed5304-964c-164c-8003-159967e82400</wsa:RelatesTo>
    <wsa:MessageID>uuid:9c753d95-9647-1647-8007-0000b5565000</wsa:MessageID>
  </s:Header>
  <s:Body>
    <wsen:PullResponse>
      <wsen:Items>
        <n1:Linux_CSProcessor>
          <n1:GroupComponent>

<wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous<
/wsa:Address>
            <wsa:ReferenceParameters>
              <wsman:ResourceURI>http://sblim.sf.net/wbem/wscim/1/cim-
schema/2/Linux_ComputerSystem</wsman:ResourceURI>
              <wsman:SelectorSet>
                <wsman:Selector
Name="CreationClassName">Linux_ComputerSystem</wsman:Selector>
                <wsman:Selector Name="Name">linux-cv22</wsman:Selector>
                <wsman:Selector
Name="__cimnamespace">root/cimv2</wsman:Selector>
              </wsman:SelectorSet>
            </wsa:ReferenceParameters>
          </n1:GroupComponent>
          <n1:PartComponent>

<wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous<
/wsa:Address>
            <wsa:ReferenceParameters>
              <wsman:ResourceURI>http://sblim.sf.net/wbem/wscim/1/cim-
schema/2/Linux_Processor</wsman:ResourceURI>
              <wsman:SelectorSet>
                <wsman:Selector
Name="SystemCreationClassName">Linux_ComputerSystem</wsman:Selector>
                <wsman:Selector Name="SystemName">linux-cv22</wsman:Selector>
                <wsman:Selector
Name="CreationClassName">Linux_Processor</wsman:Selector>
                <wsman:Selector Name="DeviceID">0</wsman:Selector>
                <wsman:Selector
Name="__cimnamespace">root/cimv2</wsman:Selector>
              </wsman:SelectorSet>
            </wsa:ReferenceParameters>
          </n1:PartComponent>
        </n1:Linux_CSProcessor>
```

```
      </wsen:Items>
      <wsen:EndOfSequence/>
    </wsen:PullResponse>
  </s:Body>
</s:Envelope>
```

References

http://findarticles.com/p/articles/mi_m0BRZ/is_6_23/ai_105884207/

http://www.usenix.org/event/lisa09/tech/full_papers/routray.pdf

Sourceforge wiki page: http://sourceforge.net/apps/mediawiki/sblim/index.php?title=Sfcb

http://trac.openwsman.org/wiki/WsEventing

WS MAN Eventing: http://www.openwsman.org/book/export/html/50

http://www.openwsman.org/book/export/html/17

http://www.openwsman.org/openwsman-users-guide/vista-winrm-over-openwsman-setup

Life Cycle Controller and WSMAN:
http://dtcftp.com/pub/WebServices/Manuals/Lifecycle_Controller_Web_Services_for_Linux_05a.pdf

http://msdn.microsoft.com/en-us/library/aa384470%28v=vs.85%29.aspx

http://www.dell.com/downloads/global/power/ps3q06-20060111-dasari-oe.pdf

http://dmtf.org/documents/cim/cim-indications-events-white-paper-210

http://www.openwsman.org/files/MDCEventing.pdf

http://www.dmtf.org/standards/cim

http://www.openwsman.org/project/openwsman

http://doc.opensuse.org/products/draft/SLES/SLES-admin/cha.wbem.html

Appendix

Working SFCB configuration:

The following configuration enables SFCB to respond to http (5988), https(5989)requests and disables all authentication. This configuration can be used as a base to start setting up the SFCB server.

```
httpPort:       5988
enableHttp:     true
enableUds:      true
httpProcs:      8
httpsPort:      5989
enableHttps:    true
httpsProcs:     8
provProcs:      32
```

```
doBasicAuth:    false
doUdsAuth:      true
basicAuthLib:   sfcBasicPAMAuthentication
useChunking:    true
keepaliveTimeout: 1
keepaliveMaxRequest: 10
sslKeyFilePath: /etc/sfcb/file.pem
sslCertificateFilePath: /etc/sfcb/server.pem
sslClientTrustStore: /etc/sfcb/client.pem
sslClientCertificate: ignore
certificateAuthLib:   sfcCertificateAuthentication
registrationDir: /var/lib/sfcb/registration
providerDirs: /usr/lib64 /usr/lib64/cmpi
enableInterOp:  true
```

FollowingFollwing are some sample commands from the remote client:

i) **`wbemcli ec http://my_user:my_password@10.208.46.57:5988/root/cim2`** (http with authentication)

ii) **`wbemcli -noverify ec https://my_user:my_password@10.208.46.57:5989`** (https with authentication but no cert verification)

iii) **`wbemcli ec http://10.208.46.57:5988/root/interop`** (http no authencation)

The sample openwsman configuration provided below will get wsman server to listen 443 port by default. The authentication is configured suing pam. The wsman server will contact the local SFCB CIMOM for all the management information.

```
[server]
port = 8889
ssl_port = 443
ssl_cert_file = /etc/openwsman/servercert.pem
ssl_key_file = /etc/openwsman/serverkey.pem
#digest_password_file = /etc/openwsman/digest_auth.passwd
#basic_password_file = /etc/openwsman/simple_auth.passwd
min_threads = 4
max_threads = 10
basic_authenticator = libwsman_pam_auth.so
basic_authenticator_arg = openwsman

#[client]
#port = 8889
#agent = openwsman 2.0.0
#
# settings for the CIM plugin
#
# can also be SfcbLocal for local connection with sfcb CIMOM running on same system
[cim]
cim_client_frontend = SfcbLocal
```

default_cim_namespace = root/cimv2
#indication_source_namespace is used to define namespace where the Indications originate
indication_source_namespace = root/cimv2

# The following are in part fake namespaces for some publicly available CIM implementations.
vendor_namespaces = OpenWBEM=http://schema.openwbem.org/wbem/wscim/1/cim-schema/2,Linux=http://sblim.sf.net/wbem/wscim/1/cim-schema/2,OMC=http://schema.omc-project.org/wbem/wscim/1/cim-schema/2,PG=http://schema.openpegasus.org/wbem/wscim/1/cim-schema/2
# CIMOM host, default is localhost
# host = localhost
# CIMOM port, default is 5988
# port = 5988

Some more example commands:

1) **wsman enumerate http://sblim.sf.net/wbem/wscim/1/cim-schema/2/Linux_CSProcessor --port 8889 --username root -p Password1 -h 10.208.46.57 --namespace root/cimv2** (using http with authentication)

2) **wsman enumerate http://sblim.sf.net/wbem/wscim/1/cim-schema/2/Linux_CSProcessor --port 443 --username root -p password -h localhost --namespace root/cimv2 --cacert=/etc/openwsman/servercert.pem** (using https along with a certificate and authentication.)

3) **wsman enumerate http://sblim.sf.net/wbem/wscim/1/cim-schema/2/Linux_CSProcessor --port 443 --username root -p password -h localhost --namespace root/cimv2 --cacert=/etc/openwsman/dummy -noverifypeer** (using https with authentication and without certificate verification.)

Disclaimer