System Management Using WBEM

Konrad Rzeszutek

IBM Linux Technology Center
Agenda

• Introduction to WBEM
  – System Management problem and WBEM solution
  – WBEM Architecture
  – CIM Specification, CIM Schemata
• Installation of WBEM on Linux®
  – CIM Broker
    • OpenPegasus, OpenWBEM
  – CIM Instrumentation
    • SBLIM
• System Management using WBEM
  – Inventory, Administration, Monitoring
• Providers (Perl and C)
• Client (command line and Python)
• Question and Answer
• Appendix
  – Installing WBEM on Windows ®, AIX ®, etc. and building WBEM applications.
  – Resources
Introduction to WBEM
• Why is system administration important to everybody?
What does system administration consist of?

• Performing the tasks of:
  – Provisioning
  – Configuration
  – Backup
  – Change management
  – Inventory
  – User administration
  – Security policies
  – Performance monitoring
  – Problem determination
Problems facing System Administrators

• Heterogeneous environments with multiple vendors for:
  – Hardware (Networks, CPUs, Hard disks)
  – Operating Systems (Linux®, Windows®, AIX®)
  – Middleware (DB2®, J2EE™)
  – Applications (PeopleSoft, Print Manager)

• Increased number and diversity of machines compounds:
  – Problem determination
  – Problem resolution
  – Accessibility
How to manage heterogeneous environments?

• Telnet
  – Command-line interface
  – Universally supported

• Web front ends
  – Flashy and easy to use
  – Web browser already installed

• Simple Network Management Protocol (SNMP)
  – *de facto* system management tool
  – Supported by all major operating systems

• Proprietary end-to-end management
  – Manages almost everything on the supported system
Problems with existing solutions

• Resource (inventory) oriented, e.g.,- telnet/Web front end:
  – Unique tools from each vendor
  – Different data formats
  – Multiple tools required to manage various data

• Task oriented, e.g.,- SNMP:
  – Limited functionality
  – Limited expandability to new products
  – Insecure

• System Administrators need to juggle both partial solutions.
Problems with proprietary System Management Suites

• Lock in and dependence on single vendor
• Limited availability of experienced developers
• High cost for customization
• Little or no support for other vendors’ systems
Existing solutions

• Often have a different perception than the user:
  – Vendors like to make their products “unique” to differentiate themselves from competitors.
  – Lots of products to support: Hardware, software.
  – Have a limited view of only their products, while user has the “bigger picture” of many vendors and products.
  – Only provides problem resolution for the vendor’s products.

• Technology is:
  – Inadequate in many ways (Security)
  – Not interoperable (Data encoding)
  – Not responsive enough or even-engineered (Web frontend)

• Have no common standards for:
  – Modeling
  – Protocol
Summary of problems.

• Lack of scope
  – Customized for single vendor, product, or problem.

• Lack of interoperability
  – Cannot interact with other vendors products.
  – Data does not conform to standard model.

• Lack of expandability
  – Not designed for end-user customization or extensibility

WBEM aims to solve these issues.
Solution: Web-based enterprise management

• “Set of management and Internet standard technologies developed to unify the management of enterprise computing environments.” [from DMTF's site http://www.dmtf.org/standards/wbem]

• Designed with the following ideas in mind:
  – Common concepts
    • ComputerSystem, NetworkCard
  – Interoperability
  – Common modeling language
  – Extending existing instrumentation and management standards
    • SNMP, DMI, etc.

• One tool to manage many different vendor's products.
Common concepts

• Generalized view on information across all products, platforms, and problems.
• Provides an extension mechanism (inheritance) to support expandability:
  – Allows expansion of the existing information models to provide required new functionality.
  – All existing information models can be re-used or expanded.
Interoperability and Modeling

- Information model is independent of transport mechanism between client and server.
  - Protocol left to other standards
    - CIM/XML over HTTP
    - Other protocols available
- Modeling language described in the Managed Object Format (MOF), which is based on Interface Definition Language (IDL).
- Information model described in Unified Modeling Language (UML) to represent complex object relationships pictorially.
WBEM implementations

- Most vendors are shipping WBEM in their product offering (Novell, AIX ®, HP-UX, Microsoft, Solaris, etc.).
- WBEM implementations are three-tiered to facilitate a modular technology independent system:
  - Management Applications aka CIM Client
    - IBM® Director, Solaris Management Console
  - CIM Object Manager (CIMOM) aka CIM Server
    - Microsoft® WMI, Sun WBEM Services, SNIA CIMOM, OpenPegasus, OpenWBEM,
  - Providers for resource access
    - Standards Based Linux Instrumentation for Manageability (SBLIM)
WBEM Architecture

Management Application

CIM API

CIM Object Manager

CIM OM Repository

Provider

Provider

Resources
WBEM standard for interoperable Systems Management

• WBEM is governed by:
  – CIM specification:
    • CIM is the language and methodology for describing management data. It is based on object oriented paradigm and a hierarchical architecture. Allows integration with other management models.
  – DMTF Schema (aka CIM Schema)
    • Provides the actual model descriptions.
CIM specification: classes

- Classes (CIM_OperatingSystem)

```csharp
class CIM_OperatingSystem : CIM_EnabledLogicalElement {
  ..
  [Description ("Number of user sessions for which the OperatingSystem is "
                 "currently storing state information.")],
  Gauge,
  MappingStrings { "MIF.DMTF|Host System|001.4",
                 "MIB.IETF|HOST-RESOURCES-MIB.hrSystemNumUsers" }]
  uint32 NumberOfUsers;

  [Description ("Number of process contexts currently loaded or running on "
                 "the OperatingSystem.")],
  Gauge,
  MappingStrings { "MIF.DMTF|Host System|001.5",
                 "MIB.IETF|HOST-RESOURCES-MIB.hrSystemProcesses" }]
  uint32 NumberOfProcesses;
  ..
```
CIM specification: instances and object path

- Instances contain the "live" data

```csharp
Instance of class CIM_OperatingSystem {
    string CSCreationClassName = "CIM_UnitaryComputerSystem";
    string CSName = "localhost";
    string CreationClassName = "CIM_OperatingSystem";
    string Name = "SuSE Distribution";
    uint16 OSType = 36;
    string Version = "SUSE LINUX Enterprise Server 9 (i586)";
    datetime LastBootUpTime = "20041129071628.000000-480";
    sint16 CurrentTimeZone = -480;
    uint32 NumberOfUsers = 8;
    uint32 NumberOfProcesses = 80;
    ..
};
```

- Object paths contain a set of key properties (hostname and namespace) that makes the instance unique:

```csharp
bender:5988/root/cimv2:PG_OperatingSystem.CreationClassName="CIM_OperatingSystem",CSCreationClassName="CIM_UnitaryComputerSystem",CSName="localhost",Name="SuSE Distribution"
```
CIM specification: associations

- Associations are a key concept of CIM models, and are similar to database references:
  - 1-to-1, 1-to-M, M-N relationships

- Associations provide:
  - Interrelationships of classes,
  - Facilitation of extensions.

- Example of an association class, CIM_OSProcess:
  - 1-to-M relationship

```cpp
[Association]
class CIM_OSProcess : CIM_Component {

  [Aggregate, Override ( "GroupComponent" ), Min ( 1 ), Max ( 1 ),
   Description ( "The OperatingSystem."))]
  CIM_OperatingSystem REF GroupComponent;

  [Override ( "PartComponent" ), Weak, Description ( "The Process running in the context of the OperatingSystem."))]
  CIM_Process REF PartComponent;
}
```
CIM_OSProcess

<table>
<thead>
<tr>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreationClassName : string {key}</td>
</tr>
<tr>
<td>Handle : string {key}</td>
</tr>
<tr>
<td>Priority : uint32</td>
</tr>
<tr>
<td>ExecutionState : uint16 {enum}</td>
</tr>
<tr>
<td>OtherExecutionDescription : string</td>
</tr>
<tr>
<td>CreationDate : datetime</td>
</tr>
<tr>
<td>TerminationDate : datetime</td>
</tr>
<tr>
<td>KernelModeTime : uint64 {units}</td>
</tr>
<tr>
<td>UserModeTime : uint64 {units}</td>
</tr>
<tr>
<td>WorkingSetSize : uint64 {units}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OperatingSystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>(See System Model (Operating System))</td>
</tr>
</tbody>
</table>

1

OSProcess

w *
Properties, methods, qualifiers

class CIM_OperatingSystem : CIM_EnabledLogicalElement {

    [Description (    
        "Boolean indicating whether the OperatingSystem is " 
        "distributed across several ComputerSystem nodes. If so, " 
        "these nodes should be grouped as a Cluster.")]
    boolean Distributed;

    [Description (    
        "A value that indicates the maximum processes" 
        "that a user can have associate with it.")]
    uint32 MaxProcessesPerUser;

    [Description (    
        "Requests a reboot of the OperatingSystem. The return value " 
        "should be 0 if the request was successfully executed, 1 if " 
        "the request is not supported and some other value if an " 
        "error occurred. In a subclass, the set of possible return " 
        "codes could be specified, using a ValueMap qualifier on the " 
        "method. The strings to which the ValueMap contents are " 
        "'translated' may also be specified in the subclass as a " 
        "Values array qualifier.")]
    uint32 Reboot();
}
Indications

- Also known as Events, Traps, or Notifications.
- Indications behave exactly like instances, but are not persistent.
- Are part of the Common Schema,
- Indications are generated by lifecycle and process events:
  - Lifecycle events cover creation, deletion, modification of elements (classes or instances).
  - Process events cover generating when an filter is matched.
- Indications require a filter, handler, and subscriptions to be specified:
  - Filters are query language constructs:
    - SELECT * FROM CIM_SNMPTrapIndication WHERE AgentAddress = '127.0.0.1'
  - Handlers specify destination
    - http://localhost:3232
  - Subscriptions tie filter and handle together.
Example of an indication class

[Indication, Version ("2.7.0"), Description ("A concrete class for mapping an SNMP Trap to CIM based on the "IETF RFC 1157. The usefulness of this class is to describe "common trap semantics. But, a complete understanding of any "trap data received relies on the Indication recipient having "access to the sender's MIB. Understanding can be improved by "mapping the SNMP domain to CIM, and using CIM LifeCycle and "standard subclasses of CIM_ProcessIndication."))]
class CIM_SNMPTrapIndication : CIM_ProcessIndication {
    [Description ("Type of object generating the trap.")],
    MappingStrings { "PDU.IETF|RFC1157-TRAP-PDU.enterprise" }]
    string Enterprise;

    [Description ("Address of the object generating the trap.")],
    MappingStrings { "PDU.IETF|RFC1157-TRAP-PDU.agent-addr" }]
    string AgentAddress;

};
Inheritance

• Used to expand upon a class model.
• Inheritance terms:
  – super class = the parent class
  – sub-class = the child class
• All properties and methods can be inherited or overridden.
• For example: CIM_UnixDeviceFile
  • CIM_UnixDeviceFile superclass is CIM_DeviceFile.
  • CIM_DeviceFile superclass is CIM_LogicalFile.
  • One of CIM_LogicalElement subclasses is CIM_LogicalFile.
Example of inheritance

class CIM_LogicalFile : CIM_LogicalElement {
    ..
    [Description (
        "Size of the File in bytes."),
        Units ( "Bytes" ), Gauge]
    uint64 FileSize;
    ..
};

class CIM_DeviceFile : CIM_LogicalFile {
};

Description (
    "DeviceFile is a special type of LogicalFile that represents a "
    "Device. This class is a specialization of DeviceFile for a "
    "Unix environment.")

class CIM_UnixDeviceFile : CIM_DeviceFile {
    ...
    [Required, Description (  
        "The Device's Major Id.")]  
    string DeviceMajor;
    
    [Required, Description (  
        "The Device's Minor Id.")]  
    string DeviceMinor;
};
CIM Schema

• Provides the actual model descriptions.
• “The CIM Schema captures notions that are applicable to all common areas of management, independent of implementations.” [from DMTF CIM Tutorial http://www.wbemsolutions.com/tutorials/CIM/cim-overview.html]

• Three schemata:
  • **Meta Model**, describes the object oriented modeling and composition features.
  • **Core Schema**, contains the essential classes for Systems Management.
  • **Common Schema**, contains the most important classes for various disciplines.

• Encodings of CIM Schema can be in
  • MOF, UML, XML
Meta Model

• Provides the basic vocabulary and concepts that describe the managed systems.
  – Simple elements
    • Schema, Classes, Associations
    • Instances, Properties, Methods
    • Qualifiers (Meta-Attributes)
  – Modeling features
    • Inheritance
    • Overriding of properties and methods
  – Associations
    • Logical grouping of classes via Schema
Core Schema

• The Core Schema defines the essential classes for system management, independent of any technology or implementation.
  – ComputerSystem
    • Name, PrimaryOwnerName, ResetCapability, etc.
  – Operating System
    • OSType, Version, CurrentTimeZone, NumberOfUsers, etc.
  – FileSystem
    • Root, BlockSize, FileSystemSize, AvailableSize, CodeSet, etc.
  – .. etc ..
Common Schema

• Technology-specific extensions of the common models in the Core Schema.

• Disciplines:
  – Applications
    • CIM_BIOSElement (Manufacturer, Version, ReleaseDate)
    • CIM_J2eeServer (Name, Vendor, Version)
  – Event
  – Network
  – Support
  – Database
  – Interop
  – Physical
  – Systems
  – Devices
  – Metrics
  – Policy
  – User
Why are these schema so important?

- WBEM, with built-in support for inheritance and the schemata to encompass many disciplines, gives two major benefits:
  - **Extensibility.** Easy to add on features. For example, one can expand the OperatingSystem class to add OS-unique properties.
  - **Re-usability.** Providers inherit values from other providers without requiring everything to be re-written or duplicated. For example, if an implementation of OperatingSystem class already exists, you can query it for values and then add new unique properties.
Installation of WBEM
Installation of WBEM on Linux ®

• The demonstration will require either CIMOM:
  – OpenPegasus
  – OpenWBEM

• And providers:
  – Standards Based Linux Instrumentation for Manageability (SBLIM)

• All three are included on SLES 9, and can also be downloaded from the Internet:
  – OpenPegasus
    • http://www.openpegasus.org
  – OpenWBEM
    • http://www.openwbem.org
  – SBLIM
    • http://www-124.ibm.com/sblim
OpenPegasus

- OpenPegasus is on the SLES 9 SDK CDs (Binaries SDK) which can be easily downloaded from:
- Source packages are available at:
  - http://www.openpegasus.org
- Install the OpenPegasus WBEM:
  - rpm -i pegasus-wbem-*-rpm
- Configure to enable HTTP connection (by default it is turned off)
  - cimconfig -s enableHttpConnection=true -p
OpenWBEM

- OpenWBEM is on the SLES 9 CDs, which can be downloaded from:
- Source packages are available at:
  - http://www.openwbem.org
- Installation is similar to OpenPegasus:
  - rpm -i openwbem-*.*.rpm
SBLIM providers

- The CIMOMs are just brokers, and without any providers they have limited useful functionality.
- The “Standards Based Linux Instrumentation for Manageability” providers address this issue.
- Source packages are available at:
- SBLIM RPM packages are on
  - SLES 9 CDs
  - Installation of RPM files from SLES9:
    - rpm -i sblim-*.rpm
SBLIM providers instrument

- Operating System, Processes
- File Systems
- Network
- NFSv3, NFSv4
- OpenLDAP
- Kernel parameters, SysFS
- SMBIOS (DMI)
- Performance matrix
- Syslog
- Indications:
  - Translate SNMP traps to CIM indications.
  - Event Log Indication
SBLIM in action
SBLIM website has additional tools

- WBEM CLI + Python script support
- Provider skeleton for CMPI
- Extensible CIM UML Tooling Environment (ecute)
- SBLIM TestSuite
- CMPI Perl stack
- CMPI Indication helper library.
- CIM JDBC Interface: “The Common Information Model is transformed to a relational view so that all informations are presented in two dimensional tables.”
System management using WBEM
Tasks demonstrated

- Inventory
- Administration
- Monitoring
WBEM applications used:

- Command line Interface:
  - CLI
  - WBEM CLI
- Graphical CIM browser
  - OpenPegasus Java™ Client
- Java Servlet Page (JSP) using CIM JDBC
- OpenOffice using CIM JDBC
- Event Subscription Application

Look in the Appendix for details on where to get these applications and how to compile them.
Inventory

- CIM_OperatingSystem
- CIM_Process
- CIM_FileSystem
- CIM_Setting
  - Linux_FileSystemParameter
  - Linux_NFSv3SystemSetting
  - Win32_Environment
  - Win32_NetworkAdapter
- CIM_Product
  - Linux_BIOSElement
- CIM_Controller
// Instance of Class PG OperatingSystem
instance of class PG_OperatingSystem:
{
    string CSCreationClassName = "CLM_UnitaryComputerSystem";
    string CName = "localhost";
    string CreationClassName = "CLM_OperatingSystem";
    string Name = "SuSE_Distribution";
    string Caption = "The current Operating System";
    string Description = "This instance reflects the Operating System on which the CIMOM is executing (as distinguished from instances of other installed operating systems that could be run).";
    string Status = "Unknown";
    uint16 OSType = 36;
    string OtherTypeDescription = "2.6.5-7.97-deb4_57 SMP Fri Jul 2 14:21:59 " "UTC 2004";
    string Version = "SUSE LINUX Enterprise Server 9 (i386)";
    datetime LastBootTime = "20041226071828.000000 480";
    datetime LocalDateTime = "20041226071828.000000-480";
    uint16 CurrentTimeZone = -480;
    uint32 NumberOfLicensedUsers = 0;
    uint32 NumberOfUsers = 9;
    uint32 NumberOfProcesses = 87;
    uint32 MaxNumberOfProcesses = 6054;
    uint64 TotalSwapSpaceSize = 1028192;
    uint64 TotalVirtualMemorySize = 1529460;
    uint64 FreeVirtualMemory = 1022392;
    uint64 FreePhysicalMemory = 6492;
    uint64 TotalVisibleMemorySize = 301308;
    uint64 MaxProcessMemorySize = 1028192;
    boolean Distributed = FALSE;
    uint32 MaxProcessesPerUser = 8054;
    uint64 SystemUpTime = 13611110;
    string OperatingSystemCapability = "32 bit";
};
[konrad@localhost konrad]$ wbemcli ei http://bender/root/cimv2:CIM_OperatingSystem

bender:5938/root/cimv2:PG_OperatingSystem.CreationClassName="CIM_OperatingSystem",CSCreationClassName="CIM_UnitaryComputerSystem",CSName="localhost",Name="SuSE Distribution" CSCreationClassName="CIM_UnitaryComputerSystem",CSName="localhost",CreationClassName="CIM_OperatingSystem",Name="SuSE Distribution",Caption="The current Operating System",Description="This instance reflects the Operating System on which the CIMOM is executing (as distinguished from instances of other installed operating systems that could be run)",Status="Unknown",OSType=36,OtherTypeDescription="2.6.5-7.97-debug #1 SMP Fri Jul 2 14:21:59 UTC 2004",Version="SUSE LINUX Enterprise Server 9 (i586)&#10;",LastBootUpTime=20041129071628.000000-480,LocalDateTime=20041215012218.000000-480,CurrentTimeZone=-480,NumberOfLicensedUsers=0,NumberOfUsers=9,NumberOfProcesses=87,MaxNumberOfProcesses=8054,TotalSwapSpaceSize=1028152,TotalVirtualMemorySize=1529460,FreeVirtualMemory=1022392,FreePhysicalMemory=7784,TotalVisibleMemorySize=501308,MaxProcessMemorySize=1028152,Distributed=FALSE,MaxProcessesPerUser=8054,SystemUpTime=1361150,OperatingSystemCapability="32 bit"

[konrad@localhost konrad]$
### bender:CIM_BIOSElement

<table>
<thead>
<tr>
<th>CIM_BIOSElement Name</th>
<th>BIOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIM_BIOSElement SoftwareElementState</td>
<td>2</td>
</tr>
<tr>
<td>CIM_BIOSElement SoftwareElementID</td>
<td>0000</td>
</tr>
<tr>
<td>CIM_BIOSElement TargetOperatingSystem</td>
<td>36</td>
</tr>
<tr>
<td>CIM_BIOSElement Version</td>
<td>2AKT34AUS</td>
</tr>
<tr>
<td>CIM_BIOSElement Manufacturer</td>
<td>IBM</td>
</tr>
<tr>
<td>CIM_BIOSElement PrimaryBIOS</td>
<td>true</td>
</tr>
<tr>
<td>CIM_BIOSElement LoadedStartingAddress</td>
<td>935632</td>
</tr>
<tr>
<td>CIM_BIOSElement LoadedEndingAddress</td>
<td>1048576</td>
</tr>
<tr>
<td>CIM_BIOSElement ReleaseDate</td>
<td>20031121110000.000000-480</td>
</tr>
</tbody>
</table>
Administration

• Methods to operate the Syslog provider:
  – Start
  – Stop
  – Restart
  – ...

• Get the Syslog_Service instance for the full list of methods.

• Browse the Syslog_Setting for settings.
CIM im -l bender Syslog_Service.SystemCreationClassName=CIM_Enumeration, SystemName=bender, CreationClassName=CIM_Service, Name=syslog @ StopService

Return Value= Shutting down syslog services..done

StartService:

Starting syslog services..done
Monitoring

• Remotely gather performance data.
• Methods to operate the gatherer:
  – StartService
  – StopService
  – StartSampling
  – StopSampling
CNI im -1 bender Linux_MetricGatherer.SystemCreationClassName="Linux_ComputerSystem",SystemName="bender",CreationClassName="Linux_MetricGatherer",Name="gatherd" StartService
Return Value = 0

wbemcli cm http://bender/root/cimv2:Linux_MetricGatherer.CreationClassName="Linux_MetricGatherer",Name="gatherd",SystemCreationClassName="Linux_ComputerSystem",SystemName="bender" StartSampling
bender:5988/root/cimv2:Linux_MetricGatherer.CreationClassName="Linux_MetricGatherer",Name="gatherd",SystemCreationClassName="Linux_ComputerSystem",SystemName="bender" StartSampling: TRUE
<table>
<thead>
<tr>
<th>Metric Value</th>
<th>MeasuredElementName</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIM_BaseMetricValue</td>
<td>TimeStamp</td>
<td>20041214230708.000000 480</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>Duration</td>
<td>00000000000000.0000000000</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>MetricValue</td>
<td>1:0</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>InstanceId</td>
<td>CPUtilTime.135.27093.1103094387</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>MetricDefinitionId</td>
<td>CPUtilTime</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>MeasuredElementName</td>
<td>27093</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>TimeStamp</td>
<td>20041214230627.000000 480</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>Duration</td>
<td>00000000000000.0000000000</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>MetricValue</td>
<td>1:0</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>InstanceId</td>
<td>KernelModeTime.136.1.1103094488</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>MetricDefinitionId</td>
<td>KernelModeTime</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>MeasuredElementName</td>
<td>1</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>TimeStamp</td>
<td>20041214230808.000000 480</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>Duration</td>
<td>00000000000000.0000000000</td>
</tr>
<tr>
<td>CIM_BaseMetricValue</td>
<td>MetricValue</td>
<td>897580</td>
</tr>
</tbody>
</table>
Events (SNMP v1/v2 traps -> CIM Indications)

• Server side setup:
  – To receive indications - you must first set up a filter:
    – SELECT * FROM CIM_SNMPTrapIndication
  – Set up a handler with the destination:
    – http://localhost:34443
  – Finally, subscribe to the indication.
    • Provide the object path to handler and filter.

• Client side setup:
  – Set up a CIM Listener on the port number passed to the handler (34443).
  – Wait until the indication is received and then process it as desired.
Data can also be sent to pagers, applications, etc to for processing.
Events (ELA)

• Server side setup:
  – Similar to the previous example, except the filter is different:
    SELECT * FROM Ela_ReportIndication
Subject: ELA from localhost
From: root@sig-5-65-54-110.mts.ibm.com
Date: 04:02 AM
To: kcnracr@ts.ibm.com

Hostname: localhost
Datetine: 12/15/04 01:03:34
Srnr:
Srnnenerrorstex:
Menunumber: 1000
Menugualerror: e.h0 (e1000 0000:00:03.0) PROCBE: The EEPROM Checksum Is Not
ValidCheck for the following:Verify driver software levelExecute diagnosticsReplace
adapterEndText
Partnumber:
Location:
Errorlogseqno: 121
Writing Perl Providers
Perl providers

• Build the Perl interface from the sblim-cmpi-perl source package:

• Functions to implement in the Perl file:
  – initializeInstance
  – enumInstanceNames
  – enumInstances
  – getInstance
  – createInstance
  – setInstance
  – deleteInstance
  – execQuery
sub enumInstanceNames() {
    my ( $broker, $context, $result, $status, $op ) = @_;  
    print STDERR "Called PerlyFile.enumInstanceNames()\n";

    my $objectpath = CMPI::ObjectPath->new(
        $broker,
        perlCMPI::CMGetCharPtr(
            perlCMPI::CMGetNameSpace( $op, $status )
        ),
        perlCMPI::CMGetCharPtr(
            perlCMPI::CMGetClasName( $op, $status )
        ),
        $status
    );
    opendir( DIRHANDLE, FILEROOT ) || die "Cannot opendir FILEROOT: $!";
    my @files = grep -f FILEROOT . "/" . $_, readdir DIRHANDLE;
    closedir(DIRHANDLE);

    my $keys_ref = $objectpath->keys;
    print @files . "\n";
    foreach my $file (@files) {
        $keys_ref->{ 'Name' } = $file;
        perlCMPI::CMReturnObjectPath( $result, $objectpath->ref );
    }
    perlCMPI::CMReturnDone($result);
    print STDERR "Done with PerlyFile.enumInstanceNames()\n";
}
Perl Provider - Enumerate Instance Names Operation

- Use SWIG 1.3.22
- Source code located at:
Writing C Providers
C providers

- Functions to implement in the provider (library):
  - CMPIStatus GetInstance(…)
  - CMPIStatus EnumerateInstances(…)
  - CMPIStatus EnumerateInstanceNames(…)
  - CMPIStatus ModifyInstance(…)
  - CMPIStatus CreateInstance(…)
  - CMPIStatus SetInstance(…)
  - CMPIStatus DeleteInstance(…)
  - CMPIStatus ExecQuery (…)

- Compilation requires CMPI header files (cmpidt.h, cmpift.h, cmpimacs.h)
  - cc -Wall -g -I ./include -c –shared –o libcmpiProvider.so Provider.c

- Lots of example source code in SBLIM site

- Example from sblim-cmpi-base provider (next page)
C Provider – Enumerate Instance Operation

```c
CMPIStatus OBase_OperatingSystemProviderEnumInstances( CMPIInstanceMI * mi, 
    CMPIContext * ctx, 
    CMPIResult * rslt, 
    CMPIOBJECTPATH * ref, 
    char ** properties) {
    CMPIInstance * ci = NULL;
    CMPIStatus rc = {CMPI_RC_OK, NULL};

    _OSBASE_TRACE(1,("--- %s CMPI EnumInstances() called",_ClassName));

    ci = _makeInst_OperatingSystem(_broker, ctx, ref, properties, &rc);
    if( ci == NULL ) {
        if( rc.msg != NULL )
            { _OSBASE_TRACE(1,("--- %s CMPI EnumInstances() failed : %s", 
                        _ClassName,CMGetCharPtr(rc.msg))); }
        else
            { _OSBASE_TRACE(1,("--- %s CMPI EnumInstances() failed",_ClassName)); }
        return rc;
    }

    CMReturnInstance( rslt, ci );
    CMReturnDone( rslt );
    _OSBASE_TRACE(1,("--- %s CMPI EnumInstances() exited",_ClassName));
    return rc;
}
```
C Provider – Enumerate Instance Operation, #2

```c
CMIInstance * _makeInst_OperatingSystem( CMPIBroker * _broker,
   CMPIContext * ctx,
   CMPIObjectPath * ref,
   const char ** properties,
   CMPIStatus * rc) {
   CMIInstance               * ci   = NULL;
   struct cim_operatingsystem * sptr = NULL;
   int                          frc  = 0;

   _OSBASE_TRACE(2,("--- _makeInst_OperatingSystem() called"));

   frc = get_operatingsystem_data(&sptr);
   if (frc==0)
      ci = _makeOS( _broker, ref, properties, sptr, rc );
   else {
      CMSetStatusWithChars( _broker, rc,
                           CMPI_RC_ERR_FAILED, "Could not get OS Data." );
      _OSBASE_TRACE(2,("--- _makeInst_OperatingSystem() failed : %s",
                       CMGetCharPtr(rc->msg)));
   }

   free_os_data(sptr);

   _OSBASE_TRACE(2,("--- _makeInst_OperatingSystem() exited"));
   return ci;
}
```
C Provider – Enumerate Instance Operation, #3

```c
static CMPIInstance * _makeOS( CMPIBroker * _broker,
                                CMPIObjectPath * ref,
                                const char ** properties,
                                struct cim_operatingsystem * sptr,
                                CMPIStatus * rc) {

  CMPIObjectPath *   op        = NULL;
  CMPIInstance   *   ci        = NULL;
  CMPIDateTime   *   dt        = NULL;

  _OSBASE_TRACE(2,("--- _makeOS() called");

  /* the sblim-cmpi-base package offers some tool methods to get common
   * system data
   */
  if( !get_system_name() ) {
    CMSSetStatusWithChars( _broker, rc,
                              CMPI_RC_ERR_FAILED, "no host name found" );
    _OSBASE_TRACE(2,("--- _makeOS() failed : %s",CMGetCharPtr(rc->msg)));
    goto exit;
  }
  if( !get_os_name() ) {
    CMSSetStatusWithChars( _broker, rc,
                              CMPI_RC_ERR_FAILED, "no OS name found" );
    _OSBASE_TRACE(2,("--- _makeOS() failed : %s",CMGetCharPtr(rc->msg)));
    goto exit;
  }
```
C Provider – Enumerate Instance Operation, #4

```c
op = CMNewObjectPath( _broker, CMGetCharPtr(CMGetNameSpace(ref,rc)),
                      _ClassName, rc );
if( CMIsNullObject(op) ) {
    CMSSetStatusWithChars( _broker, rc, CMPI_RC_ERR_FAILED, "Create CMPObjecPath failed." );
    _OSBASE_TRACE(2,("--- _makeOS() failed : %s",CMGetCharPtr(rc->msg)));
    goto exit;
}

Ci = CMNewInstance( _broker, op, rc);
if( CMIsNullObject(ci) ) {
    CMSSetStatusWithChars( _broker, rc, CMPI_RC_ERR_FAILED, "Create CMPIInstance failed." );
    _OSBASE_TRACE(2,("--- _makeOS() failed : %s",CMGetCharPtr(rc->msg)));
    goto exit;
}

....
CMSSetProperty( ci, "CSCreationClassName", CSCreationClassName, CMPI_chars );
CMSSetProperty( ci, "CSName", get_system_name(), CMPI_chars );
CMSSetProperty( ci, "CreationClassName", _ClassName, CMPI_chars );
CMSSetProperty( ci, "Name", get_os_name(), CMPI_chars );
CMSSetProperty( ci, "Status", "NULL", CMPI_chars );
CMSSetProperty( ci, "Caption", "Operating System", CMPI_chars );
CMSSetProperty( ci, "Description", "A class derived from OperatingSystem " \
               to represents the running Linux OS.", CMPI_chars );
```
C Provider – Enumerate Instance Operation, #5

```c
CMSetProperty( ci, "OSType", (CMPIValue*)&(sptr->osType), CMPI_uint16);
CMSetProperty( ci, "OtherTypeDescription", "NULL", CMPI_chars);

CMSetProperty( ci, "Version", sptr->version, CMPI_chars);

CMSetProperty( ci, "NumberOfUsers", (CMPIValue*)&(sptr->numOfUsers),
               CMPI_uint32);
CMSetProperty( ci, "NumberOfProcesses", (CMPIValue*)&(sptr->numOfProcesses),
               CMPI_uint32);
CMSetProperty( ci, "MaxNumberOfProcesses", (CMPIValue*)&(sptr->maxNumOfProc),
               CMPI_uint32);
....

exit:
    _OSBASE_TRACE(2,("--- _makeOS() exited"));
    return ci;
```

- Above example code located at:
Python Client
WBEM CLI includes Python binding

• “Python is an interpreted, interactive, object-oriented programming language. It is often compared to Tcl, Perl, Scheme or Java.” (http://www.python.org/doc/Summary.html)

• WBEM CLI has a Python binding allowing Python calls to perform WBEM operations.

Download the sblim-wbemcli source package from:
  - Follow the README.pycli
Python retrieving FileSystem information

```python
#!/usr/bin/python
execfile("cimcli.py")
hosts = ["konrad:sekret@127.0.0.1", "bender"]
for host in hosts:
    setmachine(host, locals())
    FSs = ei("CIM_FileSystem")
    print "On host: %s" % host
    for instances in FSs.ia:
        fsprops = instances.getProperties()
        print "Filesystem on %s is %s%% full (Total size: %d GB)" %
            (fsprops["ElementName"],
             fsprops["PercentageSpaceUse"],
             long(fsprops["FileSystemSize"])/
             (1024*1024*1024))
```

Filesystem on /boot is 22% full (Total size: 0 GB)
Filesystem on / is 94% full (Total size: 33 GB)
#!/usr/bin/python

execfile("cimcli.py")
hosts = ["konrad:sekret@127.0.0.1"]
for host in hosts:
    setmachine(host, locals())
OSes = ei("CIM_OperatingSystem")
for instances in OSes.ia:
    os_props = instances.getProperties()
    print "Host: %s Name: %s (%s), users: %s " % \
            (os_props["CSName"], \
             os_props["Name"], \
             os_props["OtherTypeDescription"],\n             os_props["NumberOfUsers"])

BIOSes = ei("CIM_BIOSElement")
for instances in BIOSes.ia:
    props = instances.getProperties()
    print "Host: %s BIOS Manufacturer: %s, version: %s" % \
            (os_props["CSName"], props["Manufacturer"], \n             props["Version"])

Python retrieving Operating System and BIOS
Appendix
Appendix

- Installation of WBEM on Windows ®.
- Installation of WBEM on various other UNIX® platforms.
- Installing and building WBEM applications.
- Resources
Installation of WBEM on Windows®

• Windows® has “Windows Management Instrumentation” (WMI), but their WBEM implementation does not have an HTTP over CIM interface. Hence you cannot “out-of-box” manage a Windows machine. But there are solutions.

• The OpenPegasus source includes a WMI Mapper, but you must compile it from scratch. Use Visual C++ 6.0 or later.
  – Instructions for building OpenPegasus under Windows:
    • http://cvs.opengroup.org/cgi-bin/viewcvs.cgi/pegasus/readme.html?rev=HEAD
  – Instructions for building the WMI mapper:
    • http://cvs.opengroup.org/cgi-bin/viewcvs.cgi/pegasus/src/WMIMapper/Doc/Readme.htm?rev=HEAD

• To enable:
  – Start|Settings|Control Panel|Administrative Tools|Services|Pegasus WMI Mapper
Installation of WBEM on Windows®: Binaries

- IBM Director 4.2 Agent.
- Installs a HTTP-to-WMI mapper. To enable, if it is not started automatically:
  - Start\Settings\Control Panel\Settings\Administrative Tools\Services\IBM Director Agent WMI CIM Server
- Or Hewlett-Packard WMI Mapper, which is actually the compiled version of The OpenPegasus WMI mapper.
  - Based on OpenPegasus 1.10
  - Caveat with the OpenPegasus WMI mapper:
    - Need Windows 2000 SP 4 or
    - To enable this code on locked down NT based systems (Win2k and XP) you will have to enable "Act as Operating System" and grant rights to the appropriate user group (run "secpol.msc" and select Local Policies, then User Rights (Assignments) that you want to log in as.
Installation of WBEM on other UNIX® platforms

• Installation of WBEM on AIX®
  – Use the OpenPegasus CIMOM.
  – Use the SBLIM providers.

• Installation of WBEM on HP-UX, Tru64, OpenVMS
  – Use the OpenPegasus CIMOM.
    • http://www.hp.com/large/infrastructure/management/wbem/

• Installation of WBEM on MacOS X
  – Use the OpenPegasus CIMOM
    • http://spsoftindia.com/

• Installation of WBEM on Solaris
  – Solaris WBEM (Java™)
    • http://wwws.sun.com/software/solaris/wbem/
    • http://wbemservices.sourceforge.net/
  – OpenPegasus
    • http://www.openpegasus.org
Installing and building WBEM applications: SBLIM WBEMCLI

• SBLIM WBEMCLI
  – Located on SLES 9 CD

• Or it can be downloaded from the SBLIM Web site:

• If you have an rpm:
  – rpm –i sblim-wbemcli-*.*.rpm

• If you have a tar-ball (extension .tgz):
  – Uncompress it
    • tar –zxvf sblim-wbemcli*.tgz
  – Change directory to the newly extracted directory and from there:
    • ./configure
    • make && make clipy && make install
Installing and building WBEM applications: SBLIM CMPI Perl stack

- SBLIM CMPI Perl stack
- Located on the SBLIM Web site:
- If you have a tar-ball (extension .tgz):
  - Uncompress it
    - tar -zxvf sblim-cmpi-perl*.tgz
  - Change directory to the newly extracted directory and from there:
    - ./configure
    - make && make install
  - Run CIM Server and perform CIM operations against the examples (PerlyProvider, PerlyFile, PerlyDirectory)
- If you get an error that CIM Server can’t find the providers, run the cimserver from the directory where the Perl providers are installed.
Installing and building WBEM applications: Java™ CIMBrowser

- Located in Pegasus source.
- Download .tgz or .zip file from the OpenPegasus Web site:
  - http://www.openpegasus.org
- You will also need the Xerces-J library and Java 1.3 or higher SDK.
  - http://xml.apache.org/xerces-j/
- Uncompress Pegasus
  - tar -zxvf pegasus-*.tgz
- Set environment variable PEGASUS_ROOT to the absolute path to the newly created directory:
  - export PEGASUS_ROOT=<some_path>/pegasus
- Create a src/Java/jars directory:
  - mkdir src/Java/jars
  - Extract Xerces-J in pegasus/src/jars directory
  - Follow directions outlined in pegasus/src/Java/README file
- Run ./cimBrowserSSL.sh or ./cimBrowserNoSSL.sh
Installing and building WBEM App: Event Subscription App

- Located on SBLIM website, called evsub:

- Requires:
  - Xerces
    - http://xml.apache.org/xerces-j/
    - Contains the xerces.jar file
  - SNIA CIMOM jar files
    - http://www.opengroup.org/snia-cimom/
    - Will generate two jar files: cimom.jar and httpc.jar.
  - JavaMail
    - Contains the javamail.jar file.
  - Java™ Activation Foundation
    - Contains the activation.jar file
Installing and building WBEM app: Event Subscription App, #2

• Extract all of the previously previously mentioned packages

• To build the SNIA CIMOM jar files:
  – set the environment variable EXTCLASSPATH to point to the xerces.jar file location:
    • export EXTCLASSPATH=~/xerces-1_4_4/xerces.jar
  – Run ‘make’ in the SNIA CIMOM directory.

• To build evsub:
  – Edit the 'makefile' in evsub directory to add the right paths to the previously mentioned .jar files. For example, assuming that all of the packages have been extracted in /home/konrad, it would look like this:
    • JCLASSPATH=.:~/home/konrad/cimom-2003-11-24/cimom.jar:~/home/konrad/xerces-1_4_4/xerces.jar:~/home/konrad/javamail-1.3.2/mail.jar
  – Build by running:
    • make
Installing and building WBEM app: Event Subscription App, #3

• Edit ./runit.sh to provide right paths to the mentioned jar files,
• Edit subscription.xml to change where the indication is going:
  – Sent via e-mail
  – Presented to a GUI
  – Passed in as parameters to an applications
  – Executing a method on the CIMOM.

• The sample subscription XML files and the README file give examples of how and what to change in these files.
Installing and building WBEM applications: JDBC CIM

- Located on SBLIM website, called cimjdbc:
- Requires:
  - Xerces
    - http://xml.apache.org/xerces-j/
    - The xerces.jar file
  - SNIA CIMOM jar files
    - http://www.opengroup.org/snia-cimom/
    - The httpc.jar file
  - Pegasus CIMClient jar file
    - http://www.openpegasus.org
    - The cimclient.jar file
  - Apache Ant to build the cimjdbc.jar file
    - http://ant.apache.org
Installing and building WBEM applications: **JDBC CIM #2**

- After uncompressed file, change paths in buildJarFile.xml and compile:
  - `ant -f ./buildJarFile.xml`
- The build should produce a cimjdbc.jar file.
- The package includes a sample JSP servlet. Follow the example README file for details on how to install.
- To use the JDBC CIM driver in OpenOffice, go to:
  - Next, set up a DataSource:
    - Tools|Data Sources|New Data Source|
    - Make Database Type “JDBC”
    - Hit JDBC tab and fill out the values
      - JDBC Driver class “com.ibm.wbem.jdbc.cim.cimDriver”
      - URL “cim:http:<hostname of the CIM Server>"
Installing and building WBEM applications: JDBC CIM #3
Installing and building WBEM applications: JDBC CIM #4

- And you should see CIM classes as tables:
Resources, Tutorials and White Papers

- **DMTF Schema**
  - [http://www.dmtf.org/standards/cim](http://www.dmtf.org/standards/cim)
- **DMTF WBEM FAQ**
  - [http://www.dmtf.org/about/faq/wbem/](http://www.dmtf.org/about/faq/wbem/)
- **DMTF CIM Tutorial**
  - [http://www.dmtf.org/education](http://www.dmtf.org/education)
- **SBLIM website**
- **SBLIM download packages website**
- **Common Manageability Programming Interface**
  - [http://www.wbems.sourceforge.org](http://www.wbems.sourceforge.org)
- **OpenPegasus Tutorial on C++ Providers**
- **OpenPegasus Tutorial on C++ WBEM Client**
Resources, Specifications

- **CIM Operations over HTTP, v1.1, DSP0200**

- **Representation of CIM in XML, v2.1, DSP0201**

- **Representation of CIM in XML (XML Mapping Specification), v2.0.0**
Resources, Python and Perl

- **Python**
  - http://www.python.org/
- **Python Beginners Guide**
- **Perl**
  - http://www.perl.org/
- **Perl Books**
  - http://www.perl.org/books.html
Legal Statement

- This work represents the view of the author and does not necessarily represent the view of IBM.
- IBM, the IBM (logo), e (logo) e-business, AIX, and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.
- Linux is a registered trademark of Linus Torvalds.
- Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks are registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Other company, product, and service names may be trademarks or service marks of others.