

CIM – Common Information Model

Web-based Enterprise Management

Matthias Hölzer-Klüpfel <mhk@kde.org>

2004-08-22, aKademy, Ludwigsburg

> Overview

- What's the problem?
- Distributed Management Task Force
- Common Information Model
- Managed Object Format
- CIM Core Model
- CIM Common Models
- Implementing WBEM
- Why should we care?

> Enterprise Systems Management

What is the problem?

> What's the problem?



> Enterprise-wide systems management

- Many computer systems
 - Servers (x86, SPARC, etc.)
 - Desktops
 - Laptops, PDAs
- Different operating systems
 - UNIX
 - Linux
 - even Windows!
- A lot of users
- A wide range of peripherals
 - Printers
 - Routers
 - Storage devices

> Typical situations

■ Anarchy

- use many different systems
- use many different tools for administration

=> Chaos

■ Monarchy

- use only one tool for administration
- use a small number of systems

=> Dependency

■ Dictatorship

- use only one system
- use only one tool

=> Submission

> The goal

- heterogenous systems
 - exactly what you need
 - flexible
- unified administration
 - one tool for one task
 - integrated user management
- keep freedom
 - a free market for tools
 - open-source solutions possible

=> open standard for enterprise-wide systems management

> Open standards for management

Distributed Management Task Force

> DMTF



■ Participants:

- the usual suspects (Sun, IBM, MS, HP, Intel, Cisco, Novell, Oracle and many others)

■ Mission:

- „To lead the development of management standards for distributed desktop, network, enterprise and Internet environments.”

■ Goals:

- define management standards
- ensure interoperability of implementations

> DMTF-Standards

- CIM – Common Information Model
 - a common data model to represent systems management information in enterprise computer systems
- WBEM – Web-Based Enterprise Management
 - Unification of enterprise management by using internet technologies
- DMI – Desktop Management Interface
 - framework for asset management
- DEN – Directory Enabled Network
 - building blocks for “intelligent” management by mapping CIM/WBEM to directories
- SMBIOS – System Management BIOS
 - standard for embedding management information into the system BIOS

Web-based Enterprise Management

> CIM/WBEM – Basic idea

- Build a common model of enterprise computer systems
 - Model hardware and network structure
 - Model software elements
 - Model services
- Provide a mapping between the model and the real system
 - Dynamically build instances of the model
 - Map modifications of the model back to the real system
- Access the model via a standard protocol
 - Mainly used: XML via HTTP
- Provide management applications based on the model (not the system)

> Basic Architecture

 file:///home/tom/Hacking/cvs/www/areas/events/info/conference2004/slides/Model/Basic_Architecture.wmf

> Modeling Enterprise Systems

The Common Information Model

> Common Information Model

- An object-oriented modeling approach for computer systems
- Basic elements:
 - CIM Meta model
 - A “model of the model”
 - Defines the basic framework
 - CIM Core model
 - Abstract elements common to all models
 - Defines the basic usage of the meta model
 - CIM Common models
 - Models for common application domains
- CIM Extension models
 - allows to provide application specific (i.e. proprietary) extensions
- All basic elements available as specifications from the DMTF

> CIM Meta Model

 file:///home/tom/Hacking/cvs/www/areas/events/info/conference2004/slides/Model/CIM_Meta_Schema.wmf

> CIM – Basic Elements

■ NamedElement

- Basic element in all CIM models
- Simply provides a name for all elements
- Convention: All CIM specified elements are prefixed with “CIM_”

■ Schema

- A collection of named elements
- Used to organize model elements into packages
- (Do not confuse with namespaces)

■ Class

- The most central element
- Provides the definition of a class
- Can build inheritance hierarchies

> CIM – Basic Elements (2)

■ Property

- A named value
- Defines a typed element
- Classes define properties
- Properties can be overridden in subclasses

■ Method

- Defines the signature of a class method
- Does NOT define any behavior – we are in the model space!
- (Behavior can only be defined in an Object Manager)

■ Indication

- Special class that is instantiated when an event happened

> CIM - Basic Elements (3)

■ Association

- Associations are special classes
- Contains at least two properties that are references
- Modeling associations as classes allows to add associations by introducing new classes

■ Reference

- A special property that “points” to another instance of a class

■ Qualifier

- Represents meta-data
- Can be attached to classes, methods, properties
- A useful extension mechanism

> CIM Classes

- Classes are the basic modeling element
- Classes consist of
 - Properties
 - Methods
 - Qualifiers
- Classes can inherit properties and methods from a superclass
- Classes take part in associations
- Classes can be instantiated at runtime to build a representation of the actual system

> CIM Properties

- Properties have a name and a type
- Possible data types:

Datatype	Values
[u s]int[8,16,32,64]	unsigned signed integer value with 8,16,32 or 64 bit length
string	UCS-2 character string
boolean	Boolean value
real[32 64]	Floating point value with 32 or 64 bit length
datetime	a value containing a date and a time
ref	A reference to a CIM class
char16	UCS-2 character

> CIM Qualifiers

- Qualifiers provide meta-data to model elements
- Qualifiers can be attached to
 - Classes (e.g. abstract, association)
 - Properties (e.g. read-only)
 - Methods (e.g. static)
 - Parameters (e.g. in, out)
- You can define your own qualifiers
- Standard qualifiers that can be attached everywhere:

Name	Datatype	Default value	Explanation
Description	String		A string describing the meaning of the element.
Displayname	String		A string that shall be shown to the user instead of the name of the element.

> Class qualifiers

■ Qualifiers that can be attached to classes:

Name	Datatype	Default value	Explanation
Abstract	boolean	false	Declares that the class is abstract and can not be instantiated.
Terminal	boolean	false	Declares that the class can not have subclasses.
Version	string		A string containing the major version of the class. Increment the version when incompatible changes to the class definition have been made.
Revision	string		A string containing the minor version of the class.

> Property qualifiers

■ Qualifiers that can be attached to properties:

Name	Datatype	Default value	Explanation
Alias	string		An alternative name for the property.
Counter	boolean	false	Indicates that the property is a counter that will not decrease its value (only if it wraps to zero again). Can be applied to all unsigned integer values.
Key	boolean	false	Defines this property to be part of the keys that uniquely identify an instance of the class. Keys are used in the handle for an instance.
Maxlen	int	NULL	Defines the maximum length of a string property. NULL means no restriction on the length.
Maxvalue	int	NULL	Defines the maximum value of an integer property. NULL means there is no maximum defined.
Minlen	int	NULL	Defines the minimum length of a string property. NULL means no restriction on the length.
Minvalue	int	NULL	Defines the minimum value of an integer property. NULL means there is no minimum defined.
Read	boolean	false	Indicates that the property is readable.
Required	boolean	false	Defines that the property has to have an value that is not NULL.
Static	boolean	false	Declares that this is a class property, not an instance property.
Units	string		Names the units associated with the value of this property.
Values	string[]		strings that shall be displayed to the user instead of the integer value.
Write	boolean	false	Tells that this value can be written by the user of the class.

> Method and parameter qualifiers

- Qualifiers that can be attached to a method:

Name	Datatype	Default value	Explanation
Static	boolean	false	Declares that this is a class method, not an instance method.

- Qualifiers that can be attached to a parameter:

Name	Datatype	Default value	Explanation
In	boolean	false	input.
Out	boolean	false	Indicates that this parameter will be changed by the method called.

- In addition, property qualifiers can be attached to parameters as well

Managed Object Format

> Managed Object Format

- The Managed Object Format (MOF) is used to define the models
- MOF is based in the OMG's Interface Description Language (IDL)
- MOF describes only interfaces, no behavior
- Additionally, MOF can be used to define instances

- Do not confuse the DMTF MOF with the OMG one...

- MOF can be evaluated by CIMOMs to build up the repository
- CIM Schemas (Core & Common Models) are defined in MOF

> Class definition

- A typical example for a class definition in MOF:

```
[Description ("A Linux Process")]
class LinuxProcess : CIM_Process
{
    [Description("Send a signal to a running process") ]
    string SendSignal([IN] sint32 signal);

    [Description ("Virtual memory size in KBytes") ]
    sint32 VirtualMemorySize;

    [Description ("Percentage of CPU used by process") ]
    real32 PercentCPU;
};
```

> Association definition

- An association is a specialized class:

```
[Association]
class Test_Association : CIM_Dependency
{
    [Override ("Antecedent") ]
    CIM_A Ref Antecedent;

    [Override ("Dependent") ]
    CIM_B Ref Dependent;
};
```

> Instance definition

- MOF can be used to define instances
- These instances are “static”, only put into the repository

```
instance of Linux_Process
{
    VirtualMemorySize = 12032;
    PercentCPU = 10.0;
};
```

```
instance CIM_A as $instance_a
{
    some_property = "Some value";
};
```

```
instance CIM_B as $instance_b
{
    another_property = 5;
};
```

```
instance Test_Association
{
    Antecedent = $instance_a;
    Dependent = $instance_b;
};
```

> Object identity in CIM

- Objects in CIM are identical if the key properties are identical
- This allows to create references in MOF like this:

```
instance CIM_A
{
    some_property = "Some value";
};
```

```
instance CIM_B
{
    another_property = 5;
};
```

```
instance Test_Association
{
    Antecedent = "CIM_A.some_property = \"Some value\"";
    Dependent = "CIM_B.another_property = \"5\"";
};
```

> CIM Core Model

CIM Core Model

> CIM Core Model

- The CIM Core Model defines the most basic entities
- all other CIM models are based on the Core Model

- the Core Model is expected to be very stable
- (the Common Models evolve over time)

- Core Model classes are hardly ever used directly
- Usually, derived classes are defined

> CIM Core Model Diagram

 file:///home/tom/Hacking/cvs/www/areas/events/info/conference2004/slides/Model/CIM_Core_Model.wmf

> Core Model elements

■ ManagedElement

- The root of the CIM class hierarchy
- Should be considered to be abstract
- defines the basic associations common to all elements in the CIM world:
 - ManagedElements can have dependencies to all other elements
 - for each ManagedElement, statistics providing additional information about the runtime behavior of the element can be defined
 - ManagedElements can be part of a collection
- defines some common properties:
 - Caption
 - Description

> ManagedSystemElement

- The ManagedSystemElement (MSE) represents everything in the system to be managed
- defines additional properties:
 - Name
 - Install date
 - Status
- can form component structures (MSEs can be composed of other MSEs)

> Logical / Physical elements

- CIM splits system elements into LogicalElements and PhysicalElements
- both are derived from ManageSystemElements
- PhysicalElements
 - occupy physical space
 - can be touched and seen
 - have a manufacturer, model number etc.
- LogicalElements
 - define abstract services
 - e.g. system capabilities, software
- PhysicalElements can realize (one or more) LogicalElements, e.g. multi-purpose cards

> Systems and their components

- A System is a LogicalElement representing a system, e.g. a computer system
- The most commonly used is the ComputerSystem
- A ComputerSystem
 - provides capabilities
 - hosts services
 - aggregates devices
 - contains software
 - may be dedicated to a purpose (Router, PrintServer etc.)
- all this is expressed by associations to specialized MSEs

> CIM Common Models

- CIM Common Models define models for typical application areas

- Examples:
 - CIM Device Model
 - CIM Application Management Model
 - CIM User & Security Model
 - CIM Event Model
 - CIM System Diagnostic Model
 - CIM Policy Model
 - CIM Metrics Model
 - CIM System Diagnostics Model

- Models are defined in MOF
- Altogether, several hundred classes!

> CIM Common Models

CIM Common Models

> CIM Device Model

- The Device Model describes functionality of networking and computing devices
- Areas covered:
 - Processors
 - Controllers
 - Ports
 - Network Adapters
 - Storage Devices
 - Memory
 - Modems
 - Printers
 - Sensors
 - USB devices (Ports, Hubs, Controllers)

> CIM Application Management Model

- Describes software installation and management
- Allows to model:
 - software installed on a ComputerSystem
 - discover software that is installed
 - control software distribution
- Basic model elements:
 - SoftwareProduct
 - SoftwareFeature
 - SoftwareElement

> CIM User & Security Model

- The User and Security Model defines:
 - Principals, i.e. representation of users in a system
 - Groups
 - Accounts
 - Authentication
 - Authorization
 - Roles

> CIM Event Model

- Describes Indications, signaling events
- Indications are published by CIMOMs
- Clients may register Subscriptions by creating Filters to select events and Handlers to process the events
- Indications can form a hierarchy
- Standard indications include:
 - Class creation and deletion
 - Indications provided by the monitored system
- Standard handler provides a CIM-XML request via HTTP

> WBEM at runtime

Executing CIM/WBEM

> Executing CIM

- CIM Standard defines only the models and the interfaces
- No “official” standard for the software to bring the models “to life”
- De-facto standards established:
 - CIM Object Manager (CIMOM) is the basic element
 - Access to the CIMOM via CIM-XML/HTTP
 - Static information is contained in a repository (database)
 - System information is obtained via providers
 - Client applications access CIMOM via quasi-standard APIs (JAVA API will soon be standardized)

> CIM Object Manager

 <file:///home/tom/Hacking/cvs/www/areas/events/info/conference2004/slides/Model/CIMOM.wmf>

> Necessary Elements

■ To implement a CIM-base management system, you need:

- A CIM Object Manager
 - OpenPegasus (Unix, Linux, Win32)
 - OpenWBEM (Unix, Linux)

- Providers, providers, providers
 - SBLIM (Linux)

- Client applications
 - ?



- ... or you can use the Windows Management Interface (WMI)
- WMI is a complete CIM implementation by Microsoft
- available since Windows 95
- Providers are available for all system management tasks
- Management via “Computer Management” or MMC
- Remote management possible
- Of course, Microsoft uses DCOM instead of the standard CIM-XML/HTTP :-((

> OpenPegasus

- OpenPegasus is developed by The Open Group
- Active contributors mainly from
 - IBM
 - HP
 - Veritas
- Runs on
 - Linux, Unix
 - Windows
- Written in C++, provides client API in C++ and JAVA
- MIT open source license

> OpenWBEM

- OpenWBEM is an open-source CIMOM developed by
 - Caldera (in the good old times...)
 - Vintella
 - Novell
- Written in C++, very clean C++ API
- Runs on
 - Linux, Unix
- Provides provider interfaces in C, C++, Perl

>SBLIM

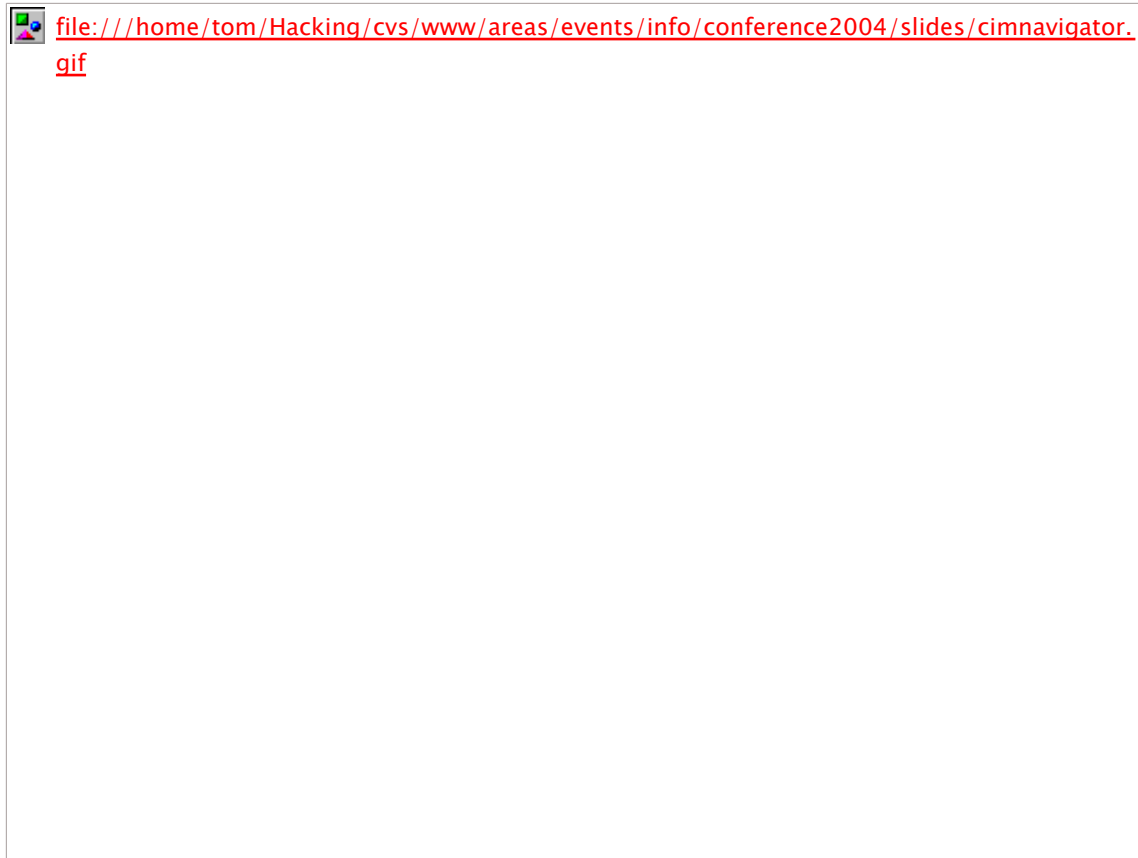
- Standards Based Linux Instrumentation for Manageability
- IBM-headed project to develop a complete set of providers
- Currently available:
 - Networking
 - Filesystems
 - SMBIOS
 - Syslog
 - Kernel parameters
 - NFS
- More coming soon...

> Client applications

- Open-source client applications:
 - severely missing!
- Until now, only developer-targeted tools available:
 - CIM Navigator
 - SBLIM Reference Implementation
 - KDE-CIM-Browser

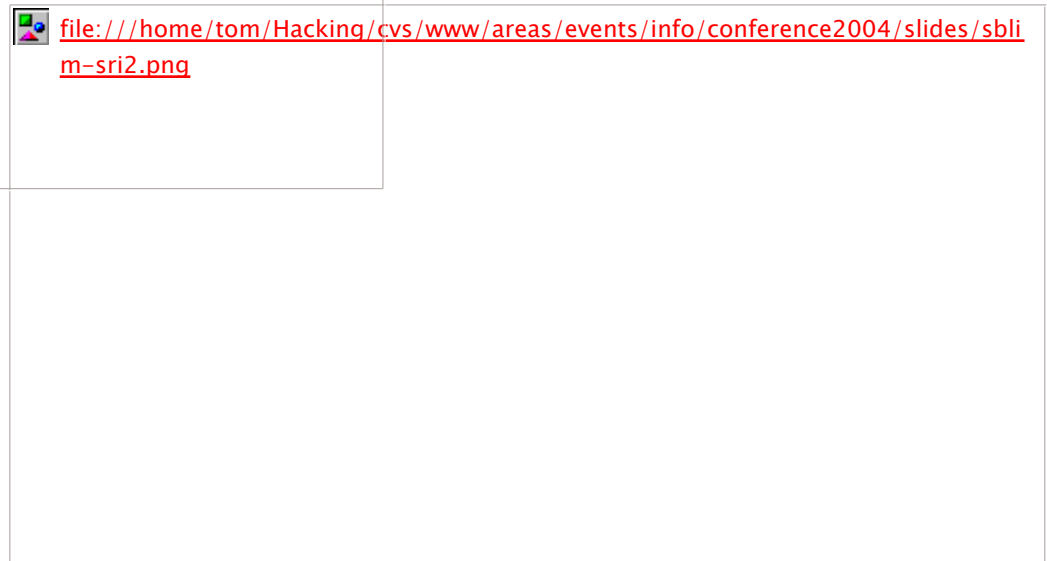
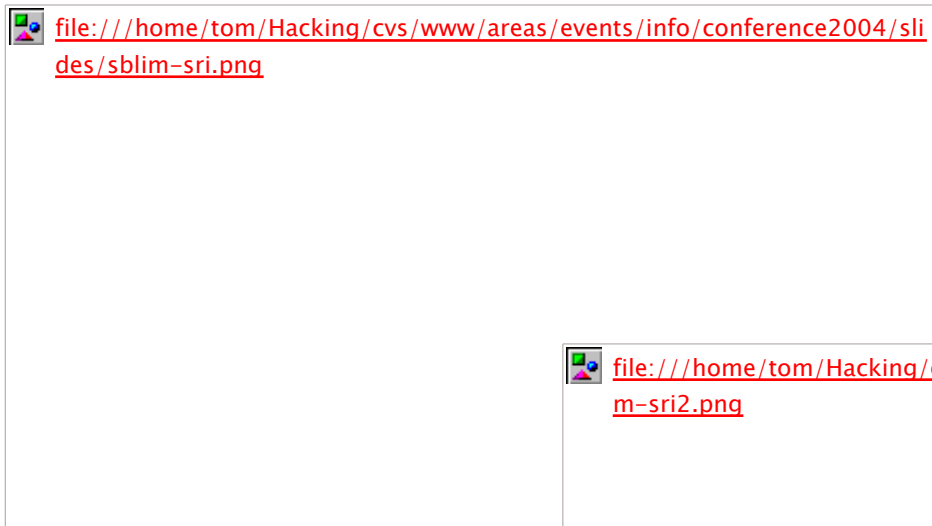
> CimNavigator

■ <http://www.cimnavigator.com>



> SBLIM Reference Implementation

- Management Demo provided by IBM




> KDE-CIM-Browser

- <http://kde-cim.sourceforge.net>
- A browser for
 - CIM Classes
 - CIM Instances
 - CIM Namespaces
- Right now (version 0.3): only read support
- Uses KDE libraries
- Makes use of the OpenWBEM-API (C++)
- Intention: get familiar with the technology

> KDE-CIM-Browser, Connection

 <file:///home/tom/Hacking/cvs/www/areas/events/info/conference2004/slides/kim1.png>


> KDE-CIM-Browser, Classview

 <file:///home/tom/Hacking/cvs/www/areas/events/info/conference2004/slides/kim2.png>

> KDE-CIM-Browser, Qualifiers

 <file:///home/tom/Hacking/cvs/www/areas/events/info/conference2004/slides/kim3.png>

> KDE-CIM-Browser, Instance

 <file:///home/tom/Hacking/cvs/www/areas/events/info/conference2004/slides/kim4.png>

> Conclusion

So what?

> Summary, so far

■ What is interesting about CIM?

- Enterprise-management is getting important
- Open standard
- Open-Source implementations
- remote administration built-in
- Growing support, especially on Linux
- No need to modify the systems

■ What is problematic about CIM?

- provider support still weak
- security issues “need to be addressed”
- no client applications available

> Why should we care?

- We can improve the manageability of KDE systems
 - by supporting the existing initiatives
 - by making KDE administrable via CIM

=> make KDE ready for the enterprise desktop
- We could get rid of the “root modules” in KControl
 - by cleanly separating the UI and the “core mechanics”

=> get rid of that ugly construction
- KDE is good at writing client applications!
 - improved KDE-CIM-Browser
 - KDE System Administration console
- Other ideas?

> Questions?

- Ask them now
- Mail mhk@kde.org
- Come to the BoF, 2004-08-24, 15:00, BoF-Room