WBEM – Web-based Enterprise Management
Outline

- What is Enterprise Management?
- What are the drivers in Enterprise Mgt
- Distributed Management Technology Forum (DMTF)
  - Web Based Enterprise Management (WBEM)
  - Common Information Model (CIM)
Enterprise Management

- Traditionally concentrated on managing IT infrastructure of an organisation:
  - Host management
  - Server management (key corporate applications)
  - Management of backbone network of organisation
- Has expanded into the management of the organisations:
  - Organisation’s network
  - Host management
  - Server management
  - Desktop Management
  - Service Management
  - Business-to-Business Management
# Traditional Enterprise Management

<table>
<thead>
<tr>
<th>Management areas</th>
<th>Traditional technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organisation’s network</td>
<td>SNMP</td>
</tr>
<tr>
<td>Host management</td>
<td>Operating systems mgt</td>
</tr>
<tr>
<td></td>
<td>Tools &amp; applications</td>
</tr>
<tr>
<td></td>
<td>3rd party enterprise mgt</td>
</tr>
<tr>
<td></td>
<td>platforms</td>
</tr>
<tr>
<td>Server management</td>
<td>as host mgt</td>
</tr>
</tbody>
</table>
Enterprise Management

- However in the late 80s – 90s
  As organisations increasingly become computer/network dependant

=> Greater importance on good IT infrastructure management
Today & Going Forward

- Organizations have:
  - Moved toward e-business
  - Moved toward more management of ‘business function’ or ‘business process’

- Current Trend
  - Virtualization of Business and Computing
  - Cloud Computing
Virtualization of Business and Computing

- The Internet has enabled the rise of virtual businesses and services
  - Restructuring of value chains as businesses exploit the economies of scale provided by new service providers
  - New entrants go after narrow value opportunities
    - Existing businesses are forced to create new eBusiness spin-offs to compete
    - Every business must decide where to focus and where it has unique competitive strengths
- Virtualization has been a real game changer!
  - Application Service Providers (ASPs) virtualize computing
  - Business Process Outsourcers (BPO) virtualize business
The Virtualization of Computing

- The web is a platform for business computing and inter-enterprise systems

- With the Internet, it provides a global addressing scheme spanning private and public networks
The Virtualization of Computing

- The web architecture separates information and services (logical) from infrastructure (physical)
  - Enables movement of servers out of local environment
    - Server farms
    - Use of remote services
  - Allows re-engineering of IT infrastructure for maximum cost-effectiveness and reliability
    - Using server-centric, thin-client application software architectures
    - Exploit the scale of broadband networks and high performance server clusters
What Happens to Enterprise IT

- Servers and operations centralized into a few large-scale server centres to gain economies of scale
- Infrastructure gradually turned into services managed by others
- Internal groups organized and measured as service providers
- Applications become services accessible via secure browser sessions
  - First generic applications, then industry applications
- Business runs securely on the internet as virtual private network (VPN) technology matures
- More applications and processes provisioned externally
  - More data, applications and processes are inter-enterprise
IT Becomes a Service Integrator

- IT becomes a service integrator, architect, and manager
  - Learns to integrate and manage a diverse set of internal and external services providers: the multi-service enterprise
  - Shifts headcount to higher value activities and out of IT operations

- Service management: manage services, not just boxes or systems, even when services provisioned externally
  - Move beyond component and technology silos to service-oriented end-to-end processes and management
  - Avoid being squeezed between greater business accountability and unresponsive service providers
Who will define interoperability in Management of Enterprises?

- Internet Management ---- IETF
- Telecom & Service Management ---- ITU-T, TMF
- Enterprise Management ---- DMTF
Distributed Management Task Force

- Industry organization that is seeking to lead the development, adoption and unification of management standards for desktop, enterprise, Internet environments.
- Composed of (predominantly) technology vendors and affiliated standards groups
- Claims to “...enable a more integrated, cost effective, and less crisis-driven approach to management through interoperable management solutions”.
DMTF Standards

- **WBEM** – Web based Enterprise Management
- **CIM** - Establishing a common model for the whole computing environment
- **DMI** - Instrumenting the Desktop and Server
What is WBEM

- Web-Based Enterprise Management is a set of management and Internet standard technologies developed to unify the management of enterprise computing environments.
What is WBEM (contd.)

● **WBEM** is:
  – Web-Based Enterprise Management
  – A DMTF standard
  – Founded in 1996 by: BMC Software, Cisco, Compaq, Intel, Microsoft
  – A standard way to get management data:
    ● Schema, Transport, Protocol
What is WBEM (contd.)

- **CIM**
  - provides a data modelling process and language (Managed Object Format). Includes standard models (schemata) for systems, applications, networks, devices, etc. Enables description of management data in a standard way.

- The xmlCIM Encoding Specification which encodes commands and responses which can be used to represent WBEM entities:
  - The definition of XML elements in DTD
  - The representation of CIM in XML specification

- The ‘CIM Operations over HTTP’ specification
  - HTTP access, the HTTP encapsulation (CIM-XML), the transport mechanism for carrying commands and responses across a network, including the ‘CIM operations over HTTP’
Consistent view of the managed environment results in an ability to manage the business rather than just its components.
Why Implement WBEM?

- WBEM is extensible, facilitating the development of platform-neutral, reusable infrastructure, tools and applications.
- In addition to its use by vendors, end users and the open source community
  - WBEM is enabling other industry organizations to build on its foundation in areas including Web services, security, storage, grid and utility computing.
- Reduced Development Cost using and re-using existing standards models, no need to “re-invent the wheel” every time
What is CIM?

- A model for describing the overall management information in a network/enterprise environment
- Provides a consistent definition and structure of data, using object oriented techniques
- Includes expressions for common elements that must be presented to management applications
  - E.g. object classes, properties, methods and associations
- Uses the Management Object Format (MOF) to define elements
CIM Overview

- CIM is:
  - Hierarchical, object oriented architecture
    - Supports tracking and depicting of the complex interdependencies and associations among objects
  - An information model - a conceptual view of the managed environment that attempts to unify and extend the existing standards (SNMP, CMIP, etc.) using OO constructs and designs
CIM Overview

- CIM Specification:
  - Includes the meta schema and the meta schema elements
  - The Managed Object Format (MOF)
  - How UML is used to diagram CIM Models.

- CIM schema:
  - Core and common models
CIM Object Oriented Overview

- CIM OO Modelling is a formal way of representing something in the real world, based upon set and classification theory.
  - **Classes** are types of things
  - **Subclasses** are subtypes of things
  - **Properties** are attributes
  - **Relationships** are pairs of attributes
  - **Instances** are things
CIM Object Oriented Overview

- **Abstraction**: The essential characteristics of an object
- **Modularity**: Decomposition of abstractions into discrete units
- **Encapsulation**: Compartmentalization of elements of abstraction that constitute its structure and behavior
- **Hierarchy**: Ranking order of abstractions
CIM Specification

- Defines syntax and rules
  - Meta Schema
  - Meta Schema Elements
  - Rules for each element
  - Syntax based on MOF
  - CIM naming mechanism
- Does NOT describe
  - CIM implementations, APIs or communication protocols
  - Or the core and common models
CIM Managed Object Format (MOF)

- Many ways CIM information can be represented to exchange information.
- MOF is based on the Interface Definition Language (IDL).
- The MOF syntax is a way to describe object definitions in textual form.
  - It establishes the syntax for writing definitions.
- The main components of a MOF specification are textual descriptions of classes, associations, properties, references, methods and instance declarations and their associated qualifiers.
  - Comments are permitted.
- A MOF file can be encoded in either Unicode or UTF-8.
CIM Managed Object Format (MOF)

- Some rules apply:
  - Comments take the C++ and C forms // or /* */
  - Names are case sensitive
  - Datatypes are better defined than C/C++ being signed or unsigned 8, 16, 32 and 64 bit integers, and 4 and 8 byte floating point numbers. Supports Booleans, strings, and datetimes as basic datatypes.
  - As with C++, constant strings can be continued over a line by closing quotation marks and re-opening them at the beginning of next line.
The MOF file is basically made up of a series of class and instance declarations.

Example: MOF definition of class CIM_ManagedElement

```
[Abstract, Version ("2.7.0"), Description ( 
    "ManagedElement is an abstract class that provides a common superclass" 
    "(or top of the inheritance tree) for the non-association classes in the CIM Schema.") ]
class CIM_ManagedElement {

    [MaxLen (64), Description ( 
        "The Caption property is a short textual description (one-line string) of the object.") ]
    string Caption;

    [Description ( 
        "The Description property provides a textual description of " 
        "the object.") ]
    string Description;

    [Description ( 
        "A user-friendly name for the object. This property allows " 
        "each instance to define a user-friendly name IN ADDITION TO its " 
        "key properties/identity data, and description information. \n" 
        "Note that ManagedSystemElement's Name property is also defined " 
        "as a user-friendly name. But, it is often subclassed to be a " 
        "Key. It is not reasonable that the same property can convey " 
        "both identity and a user friendly name, without inconsistencies. " 
        "Where Name exists and is not a Key (such as for instances of " 
        "LogicalDevice), the same information MAY be present in both " 
        "the Name and ElementName properties.") ]
    string ElementName;
```
Schemas, Classes, Properties and Methods
Each CIM class is a blueprint for a type of managed element.

Classes contain properties
- which describe the data of the class

Methods
- which describe the behavior of the class.

Scoped by the schema to which it belongs.
- must belong to only one schema and the class name must be unique within that schema.

A fully qualified class name includes the schema name using the following format:
SchemaName_ClassName.
MOF Definition of Class CIM_ManagedElement

[Abstract, Version ("2.7.0"), Description ( "ManagedElement is an abstract class that provides a common superclass" ")(or top of the inheritance tree) for the non-association classes in the CIM Schema.") ]
class CIM_ManagedElement
{
[MaxLen (64), Description ( "The Caption property is a short textual description (one-line string) of the object.") ]
string Caption;

[Description ( "The Description property provides a textual description of the object.") ]
string Description;

[Description ( "A user-friendly name for the object. This property allows each instance to define a user-friendly name IN ADDITION TO its key properties/identity data, and description information. \nNote that ManagedSystemElement's Name property is also defined as a user-friendly name. But, it is often subclassed to be a Key. It is not reasonable that the same property can convey both identity and a user friendly name, without inconsistencies. Where Name exists and is not a Key (such as for instances of LogicalDevice), the same information MAY be present in both the Name and ElementName properties.") ]
string ElementName;
};
CIM Meta Schema: Property

- A property is a value used to denote a characteristic of a class.
- A property is scoped by the class in which it is defined and must be unique within the class.
- A property has a name, data type, value and optionally a default value.
- A property that does not have a default value is initialized to null.
MOF Definition of Class CIM_ManagedElement

- Property data types are limited to the following intrinsic data types or arrays of these datatypes

<table>
<thead>
<tr>
<th>INTRINSIC DATA TYPE</th>
<th>INTERPRETATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8</td>
<td>Unsigned 8-bit integer</td>
</tr>
<tr>
<td>sint8</td>
<td>Signed 8-bit integer</td>
</tr>
<tr>
<td>uint16</td>
<td>Unsigned 16-bit integer</td>
</tr>
<tr>
<td>sint16</td>
<td>Signed 16-bit integer</td>
</tr>
<tr>
<td>uint32</td>
<td>Unsigned 32-bit integer</td>
</tr>
<tr>
<td>sint32</td>
<td>Signed 32-bit integer</td>
</tr>
<tr>
<td>uint64</td>
<td>Unsigned 64-bit integer</td>
</tr>
<tr>
<td>sint64</td>
<td>Signed 64-bit integer</td>
</tr>
<tr>
<td>string</td>
<td>UCS-2 string</td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>real32</td>
<td>IEEE 4-byte floating-point</td>
</tr>
<tr>
<td>real64</td>
<td>IEEE 8-byte floating-point</td>
</tr>
<tr>
<td>datetime</td>
<td>A string containing a date-time</td>
</tr>
<tr>
<td>&lt;classname&gt; ref</td>
<td>Strongly typed reference</td>
</tr>
<tr>
<td>char16</td>
<td>16-bit UCS-2 character</td>
</tr>
</tbody>
</table>
A method is an operation that can be invoked.
- scoped by the class in which they are defined and must be unique within the class.
- A class can have zero or more methods.

A method signature includes a name, return type, optional input parameters and optional output parameters.

The method return type must be one of the CIM supported data types.
- Return types must not be arrays.

A method parameter must be one of the CIM supported data types, fixed or variable length array of one of those types, or an object reference or array of object references.
Qualifiers provide additional information about classes, associations, indications, methods, method parameters, properties or references.

- E.g. maximum length of a string property, maximum value of numeric property, whether a property may be modified

All qualifiers have a name, type, value, scope, flavour and optionally a default value.

Examples of qualifiers include:

- Abstract (only be a superclass of other classes)
- Terminal (may not be sub-classed)
- Description
- Key – indicates that the value of a property (or reference) allowing it to be uniquely identified
- Read/Write (can be accessed/changed by operator)
- Etc.
CIM Meta Schema: Qualifier Flavors

• The flavor defines additional behavior for qualifiers.
  • For example, qualifiers can be transmitted automatically from classes to derived classes or restricted to the class in which it was defined.
  • Qualifiers can also specify whether or not derived classes can override the qualifier value, or whether it must be fixed for an entire class hierarchy.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnableOverride</td>
<td>The qualifier can be overridden</td>
<td>yes</td>
</tr>
<tr>
<td>DisableOverride</td>
<td>The qualifier can not be overridden</td>
<td>no</td>
</tr>
<tr>
<td>ToSubClass</td>
<td>The qualifier is inherited by any subclass</td>
<td>yes</td>
</tr>
<tr>
<td>Restricted</td>
<td>The qualifier applies to the class in which it is declared</td>
<td>no</td>
</tr>
<tr>
<td>Translatable</td>
<td>Indicates the value of the qualifier can be specified in multiple locales</td>
<td>no</td>
</tr>
</tbody>
</table>
CIM Meta Schema: Qualifier Flavors

- The scope defines the meta elements to which the qualifier can be applied.
- The scope must contain at least one meta element, but can contain a combination of meta elements or the keyword any to imply that the qualifier can be applied to all meta elements.
- The scope can include the following meta elements: Class, Association, Indication, Property, Reference, Method, Parameter

**Example Qualifier Type Definitions**

```
Qualifier Abstract : boolean = false,
    Scope(class, association, indication),
    Flavor(Restricted);

Qualifier Description : string = null,
    Scope(any),
    Flavor(Translatable);

Qualifier Version : string = null,
    Scope(class, association, indication),
    Flavor(Translatable);
```

**Example Qualifier Usage**

```
[Abstract, Version ("2.7.0"),
Description ( "ManagedElement is an abstract class that provides a common " "superclass (or top of the inheritance tree) for the " "non-association classes in the CIM Schema.") ]
class
CIM_ManagedElement {
```
CIM Meta Schema: Reference and Association

- **Reference**: Property data type that defines the role each object plays in an association.
- **Declared with REF key word indicating it is a pointer to other instances.**

```csharp
[Association, Version ("2.6.0"), Description ( "RunningOS indicates the currently executing OperatingSystem."
   "At most one OperatingSystem can execute at any time on a "
   "ComputerSystem. 'At most one' is specified, since the Computer"
   "System may not be currently booted, or its OperatingSystem"
   "may be unknown." ) ]
class CIM_RunningOS : CIM_Dependency {

[Override ("Antecedent"), Max (1), Description ( "The OperatingSystem currently running on the "
   "ComputerSystem.") ]
CIM_OperatingSystem REF Antecedent;

[Override ("Dependent"), Max (1), Description ( "The ComputerSystem.") ]
CIM_ComputerSystem REF Dependent;
};
```
CIM Meta Schema: Reference and Association

- Association: A type of class that contains two or more references. They represent relationships between classes.

  - E.g. The ProductSoftwareFeatures association identifies the SoftwareFeatures for a particular Product

```csharp
class CIM_ProductSoftwareFeatures {
    [Association, Aggregation, Version("2.6.0"), Description ( 
        "The ProductSoftwareFeatures association identifies the " 
        "SoftwareFeatures for a particular Product.") ]
    CIM_Product REF Product;

    [Key, Description ( 
        "The SoftwareFeature in a Product.") ]
    CIM_SoftwareFeature REF Component;
};
```
CIM Meta Schema: Indication

- An *indication* is the active representation of the occurrence of an event.
- Indications are classes that have the indication qualifier applied.
  - Since indications are types of classes, they can have properties and methods, and can be arranged in a hierarchy.
- Instances of an indication are transient and cannot be obtained by using CIM Operations
  - E.g. such as getInstance() or enumerateInstances()
- Indications can only be received by subscribing to them.
CIM Meta Schema: Indication

- Two types:
  - Life Cycle Indications - CIM class and instance life cycle events
    - Classes - class creation, deletion and modification
    - Instances - instance creation, deletion, modification, method invocation and read access
  - Process Indications - alert notifications associated with objects that may or may not be completely modeled in CIM or do not correspond to a simple life cycle event; like low-level instrumentation alerts, SNMP traps and TMN events

CIM Schema

• **Core Model:**
  • Set of classes, associations and properties that provide a basic vocabulary for describing managed systems

• **Common Models:**
  • Capture notions common to particular management areas, independent of a particular technology or implementation
  • E.g. systems, applications, networks and devices

• **Extension Schema**
  • Technology specific extensions of the common models
  • Specific to environments such as operating systems
CIM Schema: Core Model

- The core model establishes a basic classification of the elements and associations of the managed environment.
- Class hierarchy begins with the abstract Managed Element class, which has the following subclasses:
  - Managed System Element
  - Product related classes
  - Setting and Configuration, Collection
  - Statistical Data classes
  - etc.
CIM Schema: Core Model

- **Managed Element** class roots the CIM object hierarchy and acts as a reference for associations that apply to all entities in the hierarchy.

- **Managed System Elements** represent Systems, components of Systems, any kinds of services (functionality), software and networks.
  - The definition of "System" in the CIM context is quite broad, ranging from computer systems and dedicated devices, to application systems and network domains.
  - E.g. software modules, cards, integrated circuits, racks, frames, etc.

- **Core Elements**
  - PhysicalElements & Location
  - SoftwareIdentity
  - Devices
  - The "logical" function, configuration and state of hardware
  - StorageExtents (subclass of LogicalDevice)
  - Collections
  - Redundancy Information (a subclass of LogicalDevice generalizing the management of storage with more specific classes and services in the Device Model)
  - Collections
  - Redundancy Information
  - Both sparing and load balancing
  - Product and Filed Replaceable Units (FRUs)
  - Statistics
  - Capabilities - The superclass to describe the various capabilities of specific ManagedElements.
  - Settings and Profiles
  - Method Parameters
  - Power Management
  - Qualifiers - The meta and standard qualifier type definitions.
CIM Schema: Core Model

- Both **Logical** and **Physical Elements** are subclasses of Managed System Element.
- Further definition and specification of these subclasses are provided in the Core and Common Models.
  - For example, Logical Device objects are subclasses of Logical Element, defined in the Core Model.
**CIM Schema: Core Model**

- **Physical Element** is a subclass of Managed System Element
  - Provides properties such as tag (key), manufacturer, model number stock-keeping code, serial number version, part number and date of manufacture.
  - SNMP, being primarily interested in physical devices, map to this class.

- **System** and **Service** are both subclasses of Logical Element
  - System is a collection of elements working together to provide a particular functionality
  - Service is a function a device offers, think in terms of what a client would expect.
**CIM Schema: Core Model**

- **Settings** define specific, pre-configured parameter data to be "applied" (loosely transitionally) to one or more Managed System Elements.
  - Their definition is very much tied to the properties of existing objects through the Element Setting association.
  - Configurations aggregate Settings and Dependencies, representing a certain behavior or desired functional state for Managed System Elements.

- **Products** represent contracts between vendors and consumers, and capture information about how the Product was acquired, how it is supported, and where it is installed.
CIM Schema: Core Model - Significant Classes and Associations

- The **Statistical Information** class is the abstract super class for any kind of statistical data for a Managed Element.
  - The Element to which the Statistical Information applies is indicated via the Statistics association.
- **Collections** represent arbitrary "bags" that group Managed Elements together.
  - For example the DMTF User Model. A user can be modelled as a member of various groups: department, group carrying out same role, etc. these can be modelled as a collection.
- **Component associations** establish 'part of' relationships between Managed Elements.
- **Dependency associations** describe functional dependencies (one object cannot function without the other) or existence dependencies (the object cannot exist without the other) between Managed Elements.
CIM Schema: Common Models

- Common models are information models that capture notions that are common to particular management areas, but independent of a particular technology or implementation.
- Extensions can be added for platform-specific additions that supply concrete classes and implementations of common models' classes.
- The common models for CIM Schema 2.7 are:
  - Applications
  - Databases
  - Devices
  - Event
  - Interop
  - Metrics
  - Network
  - Physical
  - Policy
  - Support
  - Systems
  - User
CIM Schema: System Model

- **CIM_System** describes the aggregation of 'parts' (or components) into a single manageable 'whole' (the system).
  - Typically a system will have a name, for example, a computer has a name but its keyboard does not.
- **CIM_System** is a LogicalElement, so we do not talk about the physical computer but rather the logical functioning of computing.
- **Important concepts related to a CIM_System are:**
  - Systems act as aggregation entities.
  - Systems are not modeled as a collection. A system is more than the sum of its parts. Systems have status and they host services and access points.
  - Systems are top-level objects that are frequently used to scope their aggregated entities.
WBEM Architecture

- WBEM Server
  - Client Interface
  - Indication Handler
  - CIM Object Manager
  - Repository Interface
  - Provider Interfaces

- WBEM Clients
  - Operator
  - Alarm Operator

- Indications

- Repository

- Providers
WBEM Interfaces

- **CIM Operations over HTTP**
  - Defines both the client/server and the server/listener interfaces. It defines the operations clients can ask a WBEM server to perform and the format of messages exported by the WBEM server to listeners.

- **Representation of CIM in XML**
  - Defines the precise XML syntax of the messages a client would use to invoke a function on a WBEM server (xmlCIM). The document defines an XML grammar written in DTD.

- **XML Document Type Definition**
  - Contains the DTD grammar extracted from the xmlCIM
The Client/Server Interface

- Operator uses a graphical screen, command line interface, etc.
- Mapping to abstract model
  - Creation of new service results in creation of instances of various CIM classes
- Message encapsulated in CIM-XML and passed to HTTP client

Graphical representation:

- WBEM Client
- WBEM Server
- Managed Resource
- Application Logic
- Object Abstraction
- CIM-XML Encoding/Decoding
- HTTP Client
- CIM Object Manager
- CIM-XML Decoding/Encoding
- HTTP Server
- Application-Specific Protocol
- xmlCIM
- HTTP(S)
- Transmission Link
- Port 5988 (HTTP)
- Port 5989 (HTTPS)
CIM Message Transfer

- WBEM Client prepares request and commands in xmlCIM and logically exchanges with the WBEM server
- In practice the WBEM client passes the xmlCIM to a CIM-XML client which inserts the request or command into an HTTP message
- Once received the server unpacks and passes an xmlCIM packet to the CIM Object Manager (CIMOM)
- The request, command or response is known as a CIM Operation Message
The goal of xmlCIM is:

- to create an XML grammar which can be written in DTD (Document Type Definition)
- used both to represent CIM declarations (Classes, Instances and Qualifiers) and CIM Messages for use by CIM protocols.

Potentially many different ways in which CIM information could be represented within XML

In the interests of interoperability between different implementations of CIM there is an obvious requirement for standardization of this representation.
• While the DMTF makes no restrictions on the use of this mapping, it is recognized that a number of possible usage scenarios exist for which the mapping should cater:

1. XML documents conforming to this mapping that express CIM declarations should be capable of being rendered or transformed using standard techniques into other formats. In particular the mapping should contain sufficient information to be rendered into MOF (Managed Object Format) syntax.

2. The mapping should be applicable to the wire-level representation of CIM Messages defined by the CIM mapping onto HTTP.
A Simple CIM-XML Example:

This message contains a request to enumerate (i.e., list) the instances of the class CIM_LogicalDisk in the root/cimv2 namespace. In other words, a list of all CDROM drives being supported by a particular Linux operating system.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="87872" PROTOCOLVERSION="1.0">
    <SIMPLEREQ>
      <IMETHODCALL NAME="EnumerateInstanceNames">
        <LOCALNAMESPACEPATH>
          <NAMESPACE NAME="root"/>
          <NAMESPACE NAME="cimv2"/>
        </LOCALNAMESPACEPATH>
        <IPARAMVALUE NAME="ClassName">
          <CLASSNAME NAME="Linux_CDROMDrive"/>
        </IPARAMVALUE>
      </IMETHODCALL>
    </SIMPLEREQ>
  </MESSAGE>
</CIM>
```
All CIM Operation Message requests are defined as invocations of one or more methods. A method may be either:

- **Intrinsic**
  - Which means that it is defined by the CIM Operations over HTTP Specification for the purposes of modeling a CIM operation.
  - Oriented towards manipulating the model itself and includes methods to retrieve, delete, create, list and generally manipulate classes, instances associations and qualifiers.

- **Extrinsic**
  - Are operations carried out by the method provider which may do anything, e.g. shut down a system, bring it up or perform a complex action
WBEM Operations (Intrinsic)

**GetClass** - used to return a single CIM Class from the target Namespace.

**EnumerateClasses** - used to enumerate subclasses of a CIM Class in the target Namespace.

**EnumerateClassNames** - used to enumerate the names of subclasses of a CIM Class in the target Namespace.

**GetInstance** - used to return a single CIM Instance from the target Namespace.

**EnumerateInstances** - used to enumerate instances of a CIM Class in the target Namespace.

**EnumerateInstanceNames** - used to enumerate the names (model paths) of the instances of a CIM Class in the target Namespace.

**GetProperty** - used to retrieve a single property value from a CIM Instance in the target Namespace.

**SetProperty** - used to set a single property value in a CIM Instance in the target Namespace.

**CreateInstance** - used to create a single CIM Instance in the target Namespace. The Instance MUST NOT already exist.

**ModifyInstance** - used to modify an existing CIM Instance in the target Namespace. The Instance MUST already exist.

**DeleteInstance** - used to delete a single CIM Instance from the target Namespace.

**CreateClass** - used to create a single CIM Class in the target Namespace. The Class MUST NOT already exist.

**ModifyClass** - used to modify an existing CIM Class in the target Namespace. The Class MUST already exist.

**DeleteClass** - used to delete a single CIM Class from the target Namespace.

**Associators** - used to enumerate CIM Objects (Classes or Instances) that are associated to a particular source CIM Object.

**AssociatorNames** - used to enumerate the names of CIM Objects (Classes or Instances) that are associated to a particular source CIM Object.

**References** - used to enumerate the association objects that refer to a particular target CIM Object (Class or Instance).

**ReferenceNames** - used to enumerate the association objects that refer to a particular target CIM Object (Class or Instance).

**ExecQuery** - used to execute a query against the target Namespace.

**GetQualifier** - used to retrieve a single Qualifier declaration from the target Namespace.

**SetQualifier** - used to create or update a single Qualifier declaration in the target Namespace. If the Qualifier declaration already exists it is overwritten.

**DeleteQualifier** - used to delete a single Qualifier declaration from the target Namespace.
The table below partitions the intrinsic methods into functional groups.

<table>
<thead>
<tr>
<th>Functional Group</th>
<th>Dependency</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Read</td>
<td>None</td>
<td>GetClass</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EnumerateClasses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EnumerateClassNames</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GetInstance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EnumerateInstances</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EnumerateInstanceName</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GetProperty</td>
</tr>
<tr>
<td>Basic Write</td>
<td>Basic Read</td>
<td>SetProperty</td>
</tr>
<tr>
<td>Instance Manipulation</td>
<td>Basic Write</td>
<td>CreateInstance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ModifyInstance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DeleteInstance</td>
</tr>
<tr>
<td>Schema Manipulation</td>
<td>Instance Manipulation</td>
<td>CreateClass</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ModifyClass</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DeleteClass</td>
</tr>
<tr>
<td>Association Traversal</td>
<td>Basic Read</td>
<td>Associators</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AssociatorNames</td>
</tr>
<tr>
<td></td>
<td></td>
<td>References</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ReferenceNames</td>
</tr>
<tr>
<td>Query Execution</td>
<td>Basic Read</td>
<td>ExecQuery</td>
</tr>
<tr>
<td>Qualifier Declaration</td>
<td>Schema Manipulation</td>
<td>GetQualifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SetQualifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DeleteQualifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EnumerateQualifier</td>
</tr>
</tbody>
</table>
Basic HTTP Operations

- Works on existing web servers
- Does not mandate any new HTTP extensions
- Intuitively obvious to typical Web programmers
- Compatible with existing Web programming
- Simple in concept and implementation
Distributed Management using XML over HTTP

Managing System

Management Systems

Managed Systems

SNMP

CIM Schema

XML

XML
In order to be able to claim WBEM Compliance a product must conform to the following three requirements:

1. The DMTF CIM Compliance Specification
2. Compliance to xmlCIM Document Type Definition (DTD) encodings
3. Compliance to CIM Operations over HTTP
SNMPvCMIPvWBEM

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>SNMP</th>
<th>CMIP</th>
<th>WBEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Model</td>
<td>Object Based</td>
<td>Object Oriented</td>
<td>Object Oriented</td>
</tr>
<tr>
<td>Specification</td>
<td>SMI</td>
<td>GDMO</td>
<td>CIM(MOF)</td>
</tr>
<tr>
<td>Language</td>
<td>GET, SET, TRAP</td>
<td>M-GET, M-SET, M-</td>
<td></td>
</tr>
<tr>
<td>Operations</td>
<td></td>
<td>CREATE, M-DELETE,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>M-ACTION, M-EVEN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPORT, M-CANCEL-</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GetClass, EnumerateClasses, SetProperty, CreateInstance, ModifyInstance, DeleteInstance, CreateClass, ModifyClass, DeleteClass, GetQualifier, SetQualifier, DeleteQualifier, EnumerateQualifier, Associators, AssociatorNames, References, ReferenceNames, ExecQuery</td>
<td></td>
</tr>
<tr>
<td>Standard Body</td>
<td>IETF</td>
<td>ITU-T, ISO/OSI</td>
<td>DMTF</td>
</tr>
<tr>
<td>Addressing</td>
<td>MIT with OID at leaves of the Tree</td>
<td>MIT with OID Scoping/Filtering</td>
<td>Name and Association</td>
</tr>
</tbody>
</table>