WBEM – Web-based Enterprise Management
What is WBEM (revision)

- **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’
Schemas, Classes, Properties and Methods

- CIM model documents generally follow the convention of using **blue** lines for **inheritance**, **red** lines for **associations** and **green** lines for **aggregation**.
CIM Meta Schema: Class

- 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 behaviour 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. \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;
};
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.
**CIM Meta Schema: Qualifier**

Qualifiers provide additional information about classes, etc.
- E.g. maximum length of a string
- 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 the value of a property allow 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 - Scope

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

**Example Qualifier Type Definitions**

```java
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.**

```cim
[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.

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

  [Key, Min (1), Max (1), Aggregate, Description ( 
    "The Product that scopes the 
    SoftwareFeatures.") ]
  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.
  - Can be types of classes, have properties and methods, and 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, interoperability, devices etc.

- **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
  - 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
  - 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.

- **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.
CIM Schema: Core Model

- **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.

- **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.
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 modelled as a collection. A system is more than the sum of its parts.
  - Systems are top-level objects frequently used to scope their aggregated entities.
CIM Schema: System Model

- Also addresses computer components and functionality, associated with most computer systems, e.g:
  - file systems and files, operating systems, jobs, processes and threads, and diagnostics.

- Both general purpose (multi-task) and 'dedicated' systems (single task) can be described.

- There are no specific subclasses to describe system functionality (i.e. routing, storage, storage array, and etc.).
  - These are described by the services that are hosted or are capable of being hosted.
CIM Schema: System Model – Server Example

- **Server logical elements:**
  - Several installed operating systems, one running operating system which has jobs and processes
  - Local and remote file System that are composed of directories and files.
  - Logical devices such as a Monitor, Keyboard, Mouse, Hard Disk, Processor, Power Supply, Fan, and etc.
  - Services that are hosted on the server itself, such as spell checker or a diagnostic service.
  - Services that are available to the server via service access points, such as a print service.

- **Server physical elements:**
  - The server is contained in chassis or multiple chassis which may be mounted in a rack
  - The server consists of cards and components (chips)
  - The server occupies space at a known location
CIM Schema: Devices Model

- Addressed low-level concepts such as sensors, batteries and fans, and high-level abstractions such as Storage Volumes.
- **CIM_LogicalDevice:**
  - Represent abstract concepts of the functionality, configuration and state of hardware.
  - Typically a single hardware component provides multiple functionalities
  - Configuration of the underlying hardware and software is critical to managing the device.
  - The interaction between devices plays a crucial role in managing the device
  - Described as components of a containing CIM_System
    - described by the mandatory SystemDevice relationship
CIM Schema: Devices Model - Disk Drive Example

- The Device Model can be broken down to individual components (cooling and power, processors, storage, etc.) that are managed individually.

- Functionality:
  - **PhysicalPackage** – the drive mechanism you can see and touch
  - **DiskDrive** – the functionality to read/write data from the medium, realized as a MediaAccessDevice
  - **StorageExtent** – functionality of the medium used for storage (removable yes/no)
  - **Memory** – the internal cache buffers
  - **SoftwareIdentity** – firmware and device driver code for the drive
**CIM Schema: Devices Model - Disk Drive Example**

- The Device Model can be broken down to individual components (cooling and power, processors, storage, etc.) that are managed individually.

**Associations:**
- **MediaPresent** – tie StorageExtent to DiskDrive
- **AssociatedMemory** – tie the memory to its DiskDrive
- **DeviceSoftwareIdentity** – tie SoftwareIdentity to DiskDrive
- **Realizes** – tie DiskDrive, StorageExtent, and Memory to the PhysicalPackage
CIM Schema: Devices Model - Disk Drive Example

- The ComputerSystem has a SystemDevice relationship to:
  - PowerSupply
  - Memory (for the computer system)
  - DiskDrive
  - StorageExtent
  - Memory (for the disk drive)*

- The DiskDrive has the following associations:
  - SystemDevice to describe its component relationship to the ComputerSystem
  - MediaPresent to describe the dependent relationship with StorageExtent (its medium).
  - AssociatedMemory to describe its usage of Memory.
  - Realizes to tie to the PhysicalPackage (hardware).

*Memory for the disk drive is a component of the ComputerSystem versus a component of the disk drive. However, the Memory is associated to the DiskDrive using the AssociatedMemory relationship. This indicates that the Memory is indeed "dedicated" to the drive.
CIM Schema: Application Model

- The CIM Application Management Model describes the information commonly required to deploy and manage software products and applications.
  - standalone desktop applications to a sophisticated, multi-platform distributed, Internet-based application
  - Single software, or suite of software products
- **Schema incorporates three major concepts**
  - Structure of an application.
  - Lifecycle of an application.
  - The transition between states in the lifecycle of an application.
CIM Schema: Application Model - Structure

- **Software Product**
  - A collection of software features that can be acquired by a customer

- **Software Feature**
  - A collection of software elements that performs a particular role or function

- **Software Element**
  - A collection of one or more files and associated details deployed and managed on a particular platform

- **Application System**
  - A collection of software features that can be managed as an independent unit that supports a business function
    - E.g. sales, marketing, etc.
CIM Schema: Application Model – Transitions

1. **Deployments**
   - The deployable state describes elements in its distributable form, e.g. software repository + operations to move to next state

2. **Installation and configuration**
   - The **installable state** describes the element as ready for installation (for example, as a zip file that can be decompressed and installed) + operations to move to next state

3. **Startup**
   - The **executable state** describes the element as ready to start/run, as well as the details + operations to move to next state or back to previous

4. **Operation including monitoring**
   - The running state describes the element as it is configured and running.
CIM Schema: Interop Model

- The CIM Interop Model defines the management components that describe the WBEM infrastructure

- CIM Client
  - interacts with a CIM Server by issuing **CIM Operation Message Requests** and receives and processes **CIM Operation Message Responses**

- CIM Server
  - a server that receives and processes **CIM Operation Message Requests** and issues **CIM Operation Message Responses**

- CIM Object Manager
  - The central component of the CIM Server responsible for the communication between the CIM Server components
The CIM Interop Model defines the management components that describe the WBEM infrastructure.

**Provider**
- instruments one or more aspects of the CIM Schema
- interacts with a CIM Server by issuing CIM Operation Message Requests and receives and processes CIM Operation Message Responses
CIM Schema: Interop Model – sub models

- **CIM Object Manager Model describes**
  - the WBEM infrastructure and its relationships
  - the access mechanisms that the CIMOM supports
  - the capabilities of CIMOM
  - provides basic statistics data based on CIM Operations.

- **Namespace Model defines**
  - the namespaces that are supported by a CIMOM

- **Provider Model describes**
  - The capabilities of the class, the properties and/or methods a provider supports.
  - the mechanism in which a provider is required to register with the CIMOM.

- **Protocol Adapter Model**
  - A protocol adapter is something that accepts information using a particular protocol and converts that information so that it can be used natively, for example CIM-XML or CIM-SOAP.
WBEM Interfaces

• CIM Operations over HTTP defines
  • both the client/server and the server/listener interfaces
  • 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

```
Operator or high Level Management System

Application Logic
- Object Abstraction
- CIM-XML Encoding/Decoding
- HTTP Client

Application-Specific Protocol
- xmlCIM
- CIM-XML
- HTTP(S)

Provider
- CIM Object Manager
- CIM-XML Decoding/Encoding

HTTP Server

Transmission Link
- Port 5988 (HTTP)
- Port 5989 (HTTPS)

WBEM Client

WBEM Server
```
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 package 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.
CIM Message Transfer

- 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>
```
WBEM Operations

• 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.
WBEM Operations

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, EnumerateClasses, EnumerateClassName, GetInstance, EnumerateInstances, EnumerateInstanceName, 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, ModifyInstance, DeleteInstance</td>
</tr>
<tr>
<td>Schema Manipulation</td>
<td>Instance Manipulation</td>
<td>CreateClass, ModifyClass, DeleteClass</td>
</tr>
<tr>
<td>Association Traversal</td>
<td>Basic Read</td>
<td>Associators, AssociatorNames, References, 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, SetQualifier, DeleteQualifier, EnumerateQualifiers</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

Managed Systems

Management Systems

SNMP
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
## SNMPv CMIPv WBEM

<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 Language</td>
<td>SMI</td>
<td>GDMO</td>
<td>CIM(MOF)</td>
</tr>
<tr>
<td>Operations</td>
<td>GET, SET, TRAP</td>
<td>M-GET, M-SET, M-CREATE, M-DELETE, M-ACTION, M-EVEN REPORT, M-CANCEL-GET</td>
<td>GetClass, EnumerateClasses, GetProperty, EnumerateClassNames, GetInstance, EnumerateInstance, EnumerateInstanceNames, SetProperty, CreateInstance, ModifyInstance, DeleteInstance, CreateClass, ModifyClass, DeleteClass, GetQualifier, SetQualifier, DeleteQualifier, EnumerateQualifier, Associators, AssociatorNames, References, ReferenceNames, ExecQuery</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>
CIM Schema: Database Model

- Conceptually, there are three major entities that are modelled.
  - The database system, which represents the software application aspects of the database environment
  - The common database, which is a logical entity that represents the unit of inter-related, organized data
  - The database service, which represents the process or processes that perform tasks for the database, such as providing user access.
- Also supportive classes that represent configuration parameters, resources, and statistics
- Uses IETF's RFC 1697 specification definition of a database as an inter-related unit of data that is organized into a schema.
CIM Schema: Database Model

- The database system represents the software application aspects of the database environment.
- Database system software controls
  - the organization, retrieval, storage, security, and maintenance of a database
CIM Schema: Database Model

- It includes;
  - software inventory information for the database environment
  - software features that are meaningful from a user's perspective
  - the software elements that are part of the database software.

- Some database services perform other tasks, such as;
  - user authentication
  - authorization
  - concurrency control
  - data manipulation
  - integrity verification
  - data recovery.