

November 15-18, 2010



Santa Clara Marriott  
Santa Clara, CA

# DMTF CIM v3 Architecture: A discussion on directions

George Ericson,  
([George.Ericson@EMC.com](mailto:George.Ericson@EMC.com))

Chair

DMTF Architecture Work Group

Last Updated: 21 November 2010

DMTF Work in Progress

# Disclaimer

- The information in this presentation represents work in progress within the DMTF.
- This information is subject to change. The Standard Specifications remain the normative reference for all information.
- For additional information, see the Distributed Management Task Force (DMTF) Web site.



The DMTF was formed to lead the development, adoption and unification of management standards and initiatives for desktop, enterprise and internet environments

# What

- Directions for CIM Architecture
  - Overview of Active CIM v3work
  - Managed Object Format language
  - *Caveat:*

*Directions may change as we further investigate issues and requirements*

*The information in these slides represents a snapshot of our direction...*

# Purpose

- Expose current directions
- Invite participation



# Active CIM v3 Documents

DSP#	Title	DMTF Version	WIP Target
DSP0004	CIM Infrastructure Specification	2.7.0	Q4 2010
		3.0.0a	Q1 2011
DSP0221	CIM Managed Object Format (MOF)	1.0.0a	Q4 2010



# CIM v3 Metamodel Work Items

- Meta Elements
  - QualifierTypes
  - Associations and AssociationClasses
  - Enumerations, Interfaces, Structures, Unions, and Signals
  - References in Non-Association Classes
  - Methods
    - Override to Add to Signature
    - Allow No Return Value (void)
    - Allow complex types and arrays
  - Clarify semantics of array = NULL vs. Array[] = NULL
  - Case sensitive identifiers – mixed case is cited as a performance issue
- Qualifiers
  - Refactor qualifiers
    - Remove meta model qualifiers (Association, Indication, Exception)
    - Remove or replace obsolete or badly formed qualifiers
    - Change default value for In qualifier
    - Remove implicit qualifiers and element level flavors
  - Qualifiers as attributes of all metamodel elements



# Work that hasn't yet started

- Updates to protocol binding specifications to support the introduction of new metaelements and new datatypes
  - AssociationClass, Enumeration, Interface, Structure, Union, ...
  - OctetString, complexTypes, translatable string, wider integers, ...
  - WS-MAN and CIM-XML workgroups
- Updates to the CIM Schema to adapt to new metaelements and datatypes
  - Mostly algorithmic transforms, but a careful review will be required
  - Schema subcommittee

# CIM v3: Principal Tenets

- Native support for commonly used data structures in XML, C, Java...
- Continued support for non-deprecated CIM v2 features
- v2 schema can be transformed into v3 schema algorithmically
  - This becomes a starting point for development of the CIM v3 schema that is ultimately published
    - Development happens in profile work groups and the Schema subcommittee

# CIM Myths and Monsters

Common Complaints	Mitigation/Resolution
Too complex	<p>Datastructures with higher-level semantics are allowed / encouraged. For instance: View classes.</p> <p>Use only the CIM model elements that you need. CIM makes very few requirements on usage.</p> <p>New v3 metaelements should help to simplify by allowing more natural representations and by allowing reuse, (enumerations, structures, unions).</p>
Too much to understand	<p>CIM covers a very broad set of uses.</p> <p>Start with profiles for the domain of interest</p> <p>Remember: CIM is a dictionary. Which elements are used depends on the use cases covered by a profile.</p> <p>Deprecated v2 elements are removed in v3.</p>

## Goals for CIM v3

- Enable lighter-weight and more adaptable classes, while continuing to foster re-use.
  - Lighter weight associations
  - Reuse via enumerations, structures and unions
  - Support for Interfaces allows a later binding of operations and properties in a class inheritance hierarchy
- Rationalize with UML
  - Enable use of standard UML based tools

# New: ComplexTypes

- Complex types include:
  - Class, AssociationClass, Enumeration, Signal, Structure, and Union
- By Value usage in declarations
  - Native support for specifying complex types in declaration of properties, parameters, and return values
  - Replaces EmbeddedObject and EmbeddedInstance
- By Reference usage in declarations
  - Native support for references to AssociationClass and Class by properties of complex types.

# New: AssociationClass metaelement

- UML AssociationClass: an association that is also a class
  - UML equivalent to a CIM v2 Association
  - References are properties of the AssociationClass
  - Must be explicitly declared

# AssociationClass vs. Ordinary class

- Both are independently declared and instantiated
  - Both support additional properties
  - Both support extrinsic methods and operations
- An associationClass defines information about a relationship between two or more resources
  - An associationClass instance shall not exist unless the relationship exists
- An ordinary class defines information about a resource
  - An ordinary class instance may be defined to exist regardless of other relationships

# Example: AssociationClass transformation

CIM v2



CIM v3

## [Association] Class

```

CIM_BasedOn : CIM_Dependency {

    [Override ( "Antecedent" ) ]
    CIM_StorageExtent REF Antecedent;

    [Override ( "Dependent" ) ]
    CIM_StorageExtent REF Dependent;

    uint64 StartingAddress;

    uint16 OrderIndex;
};

```

## AssociationClass

```

CIM_BasedOn : CIM_Dependency {

    [Override ( "Antecedent" ) ]
    CIM_StorageExtent REF Antecedent;

    [Override ( "Dependent" ) ]
    CIM_StorageExtent REF Dependent;

    uint64 StartingAddress;

    uint16 OrderIndex;
};

```

# New: Association metaelement


- Lighter-weight... Just a kind of a property...
- Defined by reference properties in associated classes
  - Each property name must be unique in the associating class.
- Associations are not independently instantiated
  - Not a class
  - No properties
  - No extrinsic methods or intrinsic operations
  - **Note: association traversal is an operation on the associating class**

# AssociationClass vs. Association

- Both represent a relationship between two or more resources
- An AssociationClass IS an ordinary Class
  - Supports intrinsic and extrinsic operations
  - May have non-reference properties
  - Reference properties are present in the AssociationClass
- An association is NOT an ordinary Class
  - Essentially a REFERENCE property
  - Reference properties are present in the associated classes
  - No instances
    - No intrinsic or extrinsic operations
    - No properties

# New: Enumeration metaelement

- Replaces ValueMap and Value qualifiers

CIM v2	CIM v3
<pre>[ ... ValueMap { "0", "5", "10", "15", "20", "25", "30", ".." }, Values {   "Unknown",   "OK",   "Degraded/Warning",   "Minor failure",   "Major failure",   "Critical failure",   "Non-recoverable error",   "DMTF Reserved" }] uint16 <b>HealthState</b>;</pre>	 <pre>[ ... ReservedRanges{ "..", "24..29" }, ReservedNames { "DMTF Reserved", "MyRange" }] enumeration <b>HealthState</b> : uint16 {   Unknown = 0,   OK = 5,   DegradedWarning = 10,   MinorFailure = 15,   MajorFailure = 20,   CriticalFailure = 25,   Non-RecoverableError = 30 };</pre>

# New: Structures and Unions

## – Structure

- Use like an EmbeddedInstance but without the qualifier.
- Lighter weight than a Class
  - No Instances
  - No 'Key' properties
  - No Methods
- A 'Sequence' in XML



## – Union

- Specialized structure
- Only one property can have a value at a time
- AKA: choice list in XML

```
struct Foo
{
    enum barType: uint16 {
        stringBar;
        intBar;
    };
    union Bar {
        string y;
        int16 x;
    };
    barType BarE = stringBar;
    Bar      BarU;
};
```

# Indication/Exception


- Never independently instantiated
  - Structure replaces Indication and Exception qualifiers
  - Does not support methods

CIM v2		CIM v3
<pre>[Indication, Abstract] Class CIM_Indication {   ... };</pre>		<pre><b>Structure</b> CIM_Indication {   ... };</pre>
<pre>[Indication, Exception] Class CIM_Error {   ... };</pre>		<pre><b>Structure</b> CIM_Error {   ... };</pre>

# New: Specification type

## – Specification

- Formalize 'EmbeddedObject', where the type is a declaration of a class, structure, union, signal...

CIM v2	 CIM v3
<pre>[Indication] class CIM_ClassIndication : CIM_Indication {     <b>[EmbeddedObject]</b>     <b>string</b> ClassDefinition; };</pre>	<pre>Signal CIM_ClassIndication : CIM_Indication {     <b>CIM_Element Class</b> ClassDefinition; };</pre>

# New: Value

## – Values

- Literal values for a class, structure, union, signal... with no intention of instantiation within a namespace.
- Used for specifying values of properties or parameters with complex types.

### struct Foo

```
{
  enum barType: uint16 {
    stringBar;
    intBar;
  };
  union Bar {
    string y;
    int16 x;
  };
  barType BarE = stringBar;
  Bar BarU;
};
```



### Value of Foo as \$Foo1

```
{
  barE = intBar;
  BarU.x=5;
};
```



### Class Zen {

```
{
  Foo Puk = $Foo1;
};
```

## New: Interface metaelement

- V2 schema generally pushes properties and methods into super classes when they are shared by two or more sub classes
  - This creates a problem when not all sub classes need the added elements
- Interfaces provide a v3 alternative to define a related set of properties and methods and then to “implement” the interface at the point of use.
  - This enables simpler base models with features added only where needed.

# Interface Example

CIM v2	CIM v3
<pre> Class CIM_StorageSettings {     string InstanceID;     string ElementName;      uint16 CreateGoalSettings (         [IN, EmbeddedInstance (             "CIM_SettingData" )]         string TemplateGoalSettings[],          [IN, OUT, EmbeddedInstance (             "CIM_SettingData" )]         string SupportedGoalSettings[]     ); } </pre>	<pre> Interface CIM_CreateGoalSettings {     uint16 CreateGoalSettings (         [IN]         CIM_SettingData TemplateGoalSettings[],          [IN, OUT]         CIM_SettingData SupportedGoalSettings[]     ); }  class CIM_StorageConfigurationCapabilities :     CIM_Capabilities     IMPLEMENTS CIM_CreateSettings,     CIM_CreateGoalSettings {     [Override] string InstanceID; } </pre>
<p style="text-align: center;">Another interface</p>	

# v2 qualifier to v3 metaelement: Algorithmic transforms

- The semantics of the Qualifiers listed below are replaced by native support within the indicated CIM v3 metaelements.

CIM v2 qualifier	➔	CIM v3 metaelement
Association		AssociationClass
BitMap, BitValues		BitField in structure
EmbeddedInstance		ComplexType property
EmbeddedObject		Union of complexType properties
Indication		Structure
OctetString		OctetString data type
Reference		Reference property
Schema		schema prefix in classname
ValueMap / Values		Enumeration

# v2 qualifier to v3 qualifier: Algorithmic transforms

- The semantics of the qualifiers listed below are replaced by different qualifiers.

CIM v2 qualifier	→	CIM v3 qualifier
Aggregate / Aggregation / Composition		Replace by AggregationKind
DN		MappingStrings{ " <a href="http://www.ietf.org/rfc/rfc4514.txt">http://www.ietf.org/rfc/rfc4514.txt</a> "} }
Propagated / Weak		derivation PropertyConstraint.
Revision		Version qualifier
Units		Punits qualifier

# Qualifiers planned for Removal

- The semantics covered by the qualifiers listed below are better covered by profiles.

CIM v2 Qualifier	
Delete	Provider
Expensive / Large	Source / SourceType
Ifdeleted	Syntax / SyntaxType
Invisible	Terminal
NullValue	TriggerType
PropertyUsage	

# v2 qualifier to v3 qualifier: No Change

Unchanged qualifiers		
Abstract	IsPUnit	Override
Alias	Key	Out
ClassConstraint	MappingStrints	PropertyConstraint
Correlatable	Max	PUnit
Deprecated	MaxLen	Required
Description	MaxValue	ReservedNames
DisplayDescription	Min	ReservedValues
DisplayName	MinLen	Static
Experimental	MinValue	UMLPackagePath
Gauge	MethodConstraint	Version
In	ModelCorrespondence	XMLNamespace



# v2 qualifier to v3 qualifier: Proposed Additions

CIM v3 qualifier	Description
AggregationKind	For Associations, indicates no aggregation, shared aggregation, or composition
AssociationName	Allows a reference to name the association it belongs to
Implemented	Indicates this element is implemented
ManagedObjectDescription	Description of the resource in the managed environment
PropagationType	Qualifier propagation rule: "EnableOverride", "ChangeOnce", "Restricted", or "NoChange"
ReservedNames	Reserved ".." range names for enumerations
ReservedValues	Reserved ".." range domains for enumerations
ViewedClass	Indicates this class is a view of the named class.

# The Path to CIM v3 Schema

- First Step
  - Algorithmic Transforms produce a preliminary Schema

CIM v2	→	CIM v3
Association		AssociationClass
Class		Class
Exception		Structure
Indication		Structure
Qualifiers		Transformed or removed as indicated

- Second Step
  - Platform and Schema work groups review, apply further deprecations, and utilize Associations, Enumerations, Interfaces, Structures, and Unions to improve the CIM v3 Schema before its first release.

# Roadmap and Dependencies

DSP0221 CIM MOF Specification

DSP0004 CIM Infrastructure Specification ... (Metamodel)



**V3 Schema**

DSP1001 Profile User's Guide

DSP0223 Generic Operations

DSP1xxx Profile Specs

DSP0yyy Generic Operations Protocol Binding Specs

DSP0zzz Element Representation Specs



**Prototype Implementations**

External Standards

**End**