Document Number: DSP0201

Date: 2009-07-29

Version: 2.3.1

Representation of CIM in XML

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Foreword

The *Representation of CIM in XML* (DSP0201) was prepared by the DMTF.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability.

Introduction

This document defines an XML grammar, written in document type definition (DTD), which can be used to represent both Common Information Model (CIM) declarations (classes, instances and qualifiers) and CIM messages for use by DSP0200 (*CIM Operations over HTTP)*.

For convenience, the complete unannotated DTD is available as a separate document (DSP0203).

CIM information could be represented within XML in many different ways. In the interest of interoperability between different implementations of CIM, there is an obvious requirement for standardization of this representation. The following criteria have been applied in the design of the representation presented here:

- Fully standardized technologies are used wherever possible, in preference to Working Drafts. Where use is made of a Working Draft, the intention is to track the changes to the Working Draft in this specification.
- Completeness is favored over conciseness (all aspects of CIM should be modeled).

Although this document makes no restrictions on the use of this mapping, a number of possible usage scenarios exist for which the mapping should provide:

- 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 Managed Object Format (MOF) syntax (DSP0004).
- The mapping should be applicable to the wire-level representation of CIM messages defined by DSP0200.

A Note on Rendering to MOF

The subset of the DTD for CIM presented in this specification that concerns object declarations (identified by the element DECLARATION) is intended to allow expression of CIM objects in XML sufficient for rendering into a number of formats, including MOF.

The semantic content of a MOF file is fully captured by the DTD presented herein, which makes it possible to express any MOF conformant to DSP0004 in an equivalent XML representation using this DTD. This includes the ability to express any of the standard MOF pragmas defined in DSP0004, with the exception of the locale and instancelocale pragmas (which are subjects for further study in the context of localization support within CIM).

Note that the Processing Instruction (PI) mechanism defined by XML is the means by which bespoke pragmas may be added to an XML document in an analogous manner to the #pragma extension mechanism defined for MOF. The format of such PIs is necessarily outside the scope of this document.

A Note on Mapping Choices

There are two fundamentally different models for mapping CIM in XML:

- A *Schema Mapping* is one in which the XML schema is used to describe the CIM classes, and CIM Instances are mapped to valid XML documents for that schema. (Essentially this means that each CIM class generates its own DTD fragment, the XML element names of which are taken directly from the corresponding CIM element names.)
- A *Metaschema Mapping* is one in which the XML schema is used to describe the CIM metaschema, and both CIM classes and instances are valid XML documents for that schema. (In other words, the DTD is used to describe in a generic fashion the notion of a CIM class or

instance. CIM element names are mapped to XML attribute or element values rather than XML element names.)

Although employing a schema mapping has obvious benefits (more validation power and a slightly more intuitive representation of CIM in XML), the metaschema mapping is adopted here for the following reasons:

- It requires only one standardized metaschema DTD for CIM rather than an unbounded number of DTDs. This considerably reduces the complexity of management and administration of XML mappings.
- An XML DTD does not allow an unordered list of elements. In a static mapping, this restriction would require one of the following actions:
  - Fixing an arbitrary order for property, method, and qualifier lists (making it harder for a receiving application to process)
  - Defining a very unwieldy mapping that accounts for all list orderings explicitly (and whose size would grow exponentially with the number of list elements)
- In a schema mapping, the names of CIM schema elements (class, property, qualifier, and method names) populate the XML element namespace. To replicate the scoping rules on CIM element names within an XML DTD, it would be necessary to employ XML namespaces to define XML schema to a per-property level of granularity. This would be extremely cumbersome to administer and process. A metaschema mapping introduces only a small, fixed number of terms into the XML element namespace (such as Class, Instance, Property, and so on). As an alternative to the introduction of additional XML namespaces, some renaming of CIM elements could be used (for example, prefixing a qualifier name with the name of its owning property and its owning class), but this would result in XML documents that are verbose and difficult to understand.
- Although a schema mapping could allow XML-based validation of instances against classes, this would be possible only if the entire class hierarchy were flattened prior to mapping the CIM class to an XML schema. If this flattening was not performed, inherited properties might be absent from the DTD, which would cause validation to fail against an instance that included the value of an inherited property.

122 # Representation of CIM in XML

123 ## 1 Scope

124 The Extensible Markup Language (XML) is a simplified subset of SGML that offers powerful and
125 extensible data modeling capabilities. An *XML document* is a collection of data represented in XML. An
126 *XML schema* is a grammar that describes the format of an XML document. An XML document is
127 described as *valid* if it has an associated XML schema to which it conforms.

128 The Common Information Model (CIM) is an object-oriented information model defined by the DMTF that
129 provides a conceptual framework for describing management data.

130 This document defines a standard for the representation of CIM elements and messages in XML.

131 ## 2 Normative References

132 The following referenced documents are indispensable for the application of this document. For dated
133 references, only the edition cited applies. For undated references, the latest edition of the referenced
134 document (including any amendments) applies.

135 DMTF DSP0004, *Common Information Model (CIM) Specification 2.5*,
136 http://www.dmtf.org/standards/published_documents/DSP0004_2.5.pdf

137 DMTF DSP0200, *CIM Operations over HTTP 1.3*,
138 http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf

139 DMTF DSP0203, *CIM XML DTD 2.3*,
140 http://www.dmtf.org/standards/published_documents/DSP0203_2.3.dtd

141 W3C Recommendation, Cascading Style Sheets, level 1, April 2008, http://www.w3.org/TR/REC-CSS1/

142 W3C Recommendation, Cascading Style Sheets, level 2, April 2009, http://www.w3.org/TR/CSS2/

143 W3C Recommendation, *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, September 2006,
144 http://www.w3.org/TR/2006/REC-xml-20060816/

145 W3C Recommendation, *Namespaces in XML 1.0 (Second Edition)*, August 2006,
146 http://www.w3.org/TR/REC-xml-names/

147 W3C Recommendation, *XML Linking Language (XLink) 1.0*, June 2001, http://www.w3.org/TR/xlink/

148 W3C Recommendation, *XSL Transformations (XSLT) 1.0*, November 1999, http://www.w3.org/TR/xslt

149 ### 2.1 Other References

150 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards,*
151 http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype

152 ## 3 Terms and Definitions

153 For the purposes of this document, the following terms and definitions apply.

3.1
can
used for statements of possibility and capability, whether material, physical, or causal

3.2
cannot
used for statements of possibility and capability, whether material, physical, or causal

3.3
conditional
indicates requirements to be followed strictly in order to conform to the document when the specified conditions are met

3.4
mandatory
indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.5
may
indicates a course of action permissible within the limits of the document

3.6
need not
indicates a course of action permissible within the limits of the document

3.7
optional
indicates a course of action permissible within the limits of the document

3.8
shall
indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.9
shall not
indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.10
should
indicates that among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required

3.11
should not
indicates that a certain possibility or course of action is deprecated but not prohibited

3.12
unspecified
indicates that this specification does not define any constraints for the referenced CIM element or operation

3.13
CIM element
one of the following components of the CIM metamodel: Namespace, Class, Property, Method, or Qualifier

3.14
XML element
a component of XML that is defined using the ELEMENT construct in the DTD

4 Symbols and Abbreviated Terms

The following symbols and abbreviations are used in this document.

4.1
CIM
Common Information Model

4.2
DTD
document type definition

4.3
PI
processing instruction

4.4
XML
Extensible Markup Language

5 CIM XML Schema Reference

The following subclauses describe the CIM XML Schema entities and elements.

5.1 Entity Descriptions

This section describes each of the parameter entities used in the CIM XML schema vocabulary. The use of parameter entities has been adopted to highlight common features of the DTD.

5.1.1 CIMName

The CIMName entity describes the name of a CIM element (class, instance, method, property, qualifier, or parameter). The value must be a legal CIM element name (DSP0004).

```
<!ENTITY % CIMName "NAME  CDATA  #REQUIRED">
```

5.1.2 CIMType

The CIMType entity describes the allowed type descriptions for a non-reference CIM property, CIM qualifier, or non-reference CIM method parameter.

```
<!ENTITY % CIMType "TYPE
     (boolean | string | char16 | uint8 | sint8 | uint16 | sint16 | uint32 |
     sint32 | uint64 | sint64 | datetime | real32 | real64)">
```

5.1.3 QualifierFlavor

The QualifierFlavor entity describes the flavor settings for a CIM qualifier, modeled as XML attributes.

DEPRECATION NOTE: The TOINSTANCE attribute is deprecated and may be removed from the QualifierFlavor entity in a future version of this document. Use of this qualifier is discouraged.

```
<!ENTITY % QualifierFlavor "OVERRIDABLE (true|false)  'true'
     TOSUBCLASS  (true|false)  'true'
     TOINSTANCE  (true|false)  'false'
     TRANSLATABLE (true|false)  'false'">
```

5.1.4 ClassOrigin

The ClassOrigin entity describes the originating class of a CIM property or method.

The CLASSORIGIN attribute defines the name of the originating class (the class in which the property or method was first defined) of the CIM element represented by the XML element to which the attribute is attached.

```
<!ENTITY % ClassOrigin   "CLASSORIGIN CDATA #IMPLIED">
```

5.1.5 Propagated

The Propagated entity is a convenient shorthand for the PROPAGATED attribute, which may apply to a CIM property, method, or qualifier.

This attribute indicates whether the definition of the CIM property, qualifier, or method is local to the CIM class (respectively, instance) in which it appears, or was propagated without modification from the underlying subclass (respectively, class), as defined by the DSP0004.

```
<!ENTITY % Propagated    "PROPAGATED (true|false) 'false'">
```

Uses of the PROPAGATED attribute include:

- To facilitate the rendering of CIM XML declarations into MOF syntax, which by convention only describes local overrides in a CIM subclass or instance
- To filter XML representations of CIM classes or instances so that they can be returned as responses to CIM operation requests (DSP0200), which require only local elements

5.1.6 ArraySize

The ArraySize entity is a convenient shorthand for the ARRAYSIZE attribute, which may apply to a PROPERTY.ARRAY, PARAMETER.ARRAY, or PARAMETER.REFARRAY element.

```
<!ENTITY % ArraySize    "ARRAYSIZE CDATA #IMPLIED">
```

The ARRAYSIZE attribute defines the size of the array when it is constrained to a fixed number of elements. The value of this attribute (if it is present) must be a positive integer.

5.1.7 SuperClass

The SuperClass entity is a convenient shorthand for the SUPERCLASS attribute.

```
<!ENTITY % SuperClass   "SUPERCLASS CDATA #IMPLIED">
```

This attribute defines the name of the superclass. Where it is omitted, it must be inferred that the owning element has no superclass.

5.1.8 ClassName

The ClassName entity is a convenient shorthand for the CLASSNAME attribute. The value must be a legal CIM class name (DSP0004).

```
<!ENTITY % ClassName    "CLASSNAME CDATA #REQUIRED">
```

5.1.9 ReferenceClass

The ReferenceClass entity is a convenient shorthand for the REFERENCECLASS attribute. If this entity is present, the value must be a legal CIM class name (DSP0004).

```
<!ENTITY % ReferenceClass "REFERENCECLASS CDATA #IMPLIED">
```

The value defines the class name for the reference, and the requirement for the existence of this attribute depends on the entity in which it is used. The expected behavior is that the REFERENCECLASS attribute must exist for classes and should not exist for instances.

5.1.10 ParamType

The ParamType entity describes the allowed type descriptions for parameter values or return values.

```
<!ENTITY % ParamType "PARAMTYPE
     (boolean | string | char16 | uint8 | sint8 | uint16 | sint16 | uint32 |
     sint32 | uint64 | sint64 | datetime | real32 | real64 | reference |
     object|instance)">
```

5.1.11 EmbeddedObject

The EmbeddedObject entity defines an embedded object or an embedded instance. This entity may be applied only to entities that have the Type string.

```
<!ENTITY % EmbeddedObject "(object | instance) #IMPLIED">
```

This attribute is to be used to represent the existence of an EMBEDDEDINSTANCE or EMBEDDEDOBJECT qualifier on the corresponding metadata (method, parameter, or property).

If the EMBEDDEDOBJECT qualifier is defined for the method, parameter, or property, the `EmbeddedObject` attribute must be attached to the corresponding property in any instance, PARAMVALUE, or RETURNVALUE with the value "object".

If the EMBEDDEDINSTANCE qualifier exists for the method, parameter, or property, the `EmbeddedObject` attribute must be attached to the corresponding property in any instance, PARAMVALUE, or RETURNVALUE with the value "instance".

5.2 Element Descriptions

This section describes each of the elements in the CIM XML schema.

5.2.1 Top-Level Element: CIM

The CIM element is the root element of every XML document that is valid with respect to this schema.

Each document takes one of two forms: it contains a single MESSAGE element that defines a CIM message (to be used in DSP0200), or it contains a DECLARATION element that is used to declare a set of CIM objects.

```
<!ELEMENT CIM (MESSAGE|DECLARATION)>
<!ATTLIST CIM
     CIMVERSION CDATA #REQUIRED
     DTDVERSION CDATA #REQUIRED>
```

The `CIMVERSION` attribute defines the version of the DSP0004 to which the XML document conforms. It must be in the form of "M.N.U", where M is the major version of the specification, N is the minor version of the specification, and U is the update version of the specification, each in their decimal representation without leading zeros. Any draft letter in the version of the specification must not be represented in the attribute (for example, 2.3.0, 2.4.0). Implementations must validate only the major version, as all minor and update versions are backward compatible. Implementations may look at the minor or update version to determine additional capabilities.

The `DTDVERSION` attribute defines the version of the *Specification for the Representation of CIM in XML* (this document) to which the XML document conforms. It must be in the form of "M.N.U", where M is the major version of the specification, N is the minor version of the specification, and U is the update version of the specification, each in their decimal representation without leading zeros. Any draft letter in the version of the specification must not be represented in the attribute (for example, 2.2.0, 2.3.0). Implementations must validate only the major version, as all minor and update versions are backward compatible. Implementations may look at the minor or update version to determine additional capabilities.

5.2.2 Declaration Elements

This section defines those elements of the schema that are concerned with expressing the declaration of CIM objects.

5.2.2.1 DECLARATION

The DECLARATION element defines a set of one or more declarations of CIM objects. These are partitioned into logical declaration subsets.

```
<!ELEMENT DECLARATION (DECLGROUP|DECLGROUP.WITHNAME|DECLGROUP.WITHPATH)+>
```

5.2.2.2 DECLGROUP

The DECLGROUP element defines a logical set of CIM class, instance, and qualifier declarations. It may optionally include a NAMESPACEPATH or LOCALNAMESPACEPATH element, which, if present, defines the common namespace in which all objects within the group are declared.

The objects within the group are CIM classes, instances, and qualifiers. Object declarations must be ordered correctly with respect to the target implementation state. If the DECLGROUP element references a class without defining it first, the server must reject it as invalid if it does not already have a definition of that class.

```
<!ELEMENT DECLGROUP
((LOCALNAMESPACEPATH|NAMESPACEPATH)?,QUALIFIER.DECLARATION*,VALUE.OBJECT*)>
```

5.2.2.3 DECLGROUP.WITHNAME

The DECLGROUP.WITHNAME element defines a logical set of CIM class, instance, and qualifier declarations. It may optionally include a NAMESPACEPATH or LOCALNAMESPACEPATH element, which, if present, defines the common namespace in which all objects within the group are declared.

The objects within the group are CIM classes, instances, and qualifiers. Object declarations must be ordered correctly with respect to the target implementation state. If the DECLGROUP.WITHNAME element references a class without defining it first, the server must reject it as invalid if it does not already have a definition of that class.

The DECLGROUP.WITHNAME element extends the DECLGROUP element in the sense that any instance declaration contains an explicit instance name (that is, a model path in the terms of DSP0004).

```
<!ELEMENT DECLGROUP.WITHNAME
((LOCALNAMESPACEPATH|NAMESPACEPATH)?,QUALIFIER.DECLARATION*,VALUE.NAMEDOBJECT*)>
```

5.2.2.4 DECLGROUP.WITHPATH

The DECLGROUP.WITHPATH element defines a logical set of CIM class and instance declarations. Each object is declared with its own independent naming and location information. Object declarations must be ordered correctly with respect to the target implementation state. If the DECLGROUP.WITHPATH element references a class without defining it first, the server must reject it as invalid if it does not already have a definition of that class.

```
<!ELEMENT DECLGROUP.WITHPATH
(VALUE.OBJECTWITHPATH|VALUE.OBJECTWITHLOCALPATH)*>
```

5.2.2.5 QUALIFIER.DECLARATION

The QUALIFIER.DECLARATION element defines a single CIM qualifier declaration.

A VALUE or a VALUE.ARRAY subelement must be present if the qualifier declaration has a non-NULL default value defined. A VALUE subelement is used if the qualifier has a non-array type. A VALUE.ARRAY subelement is used if the qualifier has an array type. Absence of the VALUE and VALUE.ARRAY subelements must be interpreted as a default value of NULL.

The SCOPE subelement, if present, defines the valid set of scopes for this qualifier. Absence of the SCOPE subelement implies that there is no restriction on the scope at which the qualifier may be applied (so that it has “any” scope in the terminology of DSP0004).

```
<!ELEMENT QUALIFIER.DECLARATION (SCOPE?,(VALUE|VALUE.ARRAY)?)>
<!ATTLIST QUALIFIER.DECLARATION
    %CIMName;
    %CIMType;                  #REQUIRED
    ISARRAY    (true|false)    #IMPLIED
    %ArraySize;
    %QualifierFlavor;>
```

The `CIMName` attribute defines the name of the qualifier, and the `CIMType` and `ISARRAY` attributes together define the CIM type. The `ISARRAY` attribute must be present if the qualifier declares no default value, in order to infer whether the qualifier has an array type. The `ISARRAY` attribute should be absent if the qualifier declares a non-NULL default value; in this case, whether the qualifier has an array type can be deduced from whether a VALUE or VALUE.ARRAY element is used to declare that default. If the `ISARRAY` attribute is present, its value must be consistent with the declared qualifier default value.

The `ArraySize` attribute must not be present if the value of the `ISARRAY` attribute is `true`. The presence of the `ArraySize` attribute indicates that the values taken by this qualifier must be of the size specified by the value of this attribute.

The flavor attributes declared using the QualifierFlavor entity define the propagation and override semantics for the qualifier.

5.2.2.6 SCOPE

The SCOPE element defines the scope of a QUALIFIER.DECLARATION when there are restrictions on the scope of the qualifier declaration.

```
<!ELEMENT SCOPE EMPTY>
<!ATTLIST SCOPE
     CLASS     (true|false)   "false"
     ASSOCIATION (true|false)   "false"
     REFERENCE  (true|false)   "false"
     PROPERTY   (true|false)   "false"
     METHOD    (true|false)   "false"
     PARAMETER  (true|false)   "false"
     INDICATION  (true|false)   "false">
```

The attributes define which scopes are valid. A SCOPE element must declare at least one attribute with a `true` value. (Otherwise, the qualifier would have no applicable scope.)

5.2.3 Value Elements

This section defines those elements of the schema that are concerned with expressing the value of CIM objects.

5.2.3.1 VALUE

The VALUE element is used to define a single (non-array), non-reference, non-NULL CIM property value, CIM qualifier value, CIM method return value, or CIM method parameter value.

```
<!ELEMENT VALUE (#PCDATA)>
```

Because a value's type cannot be validated using DTD, each value appears in PCDATA format irrespective of the type. The TYPE attribute of the parent element determines the (CIM) type of the value. The format of the PCDATA value depends on the CIM type and is described in the following subclauses.

5.2.3.1.1 String Values

If the CIM type is `string`, the PCDATA value must be a sequence of zero or more UCS-2 characters. An empty PCDATA value represents an empty string (that is, ""). The value must not be surrounded by string delimiter characters (such as double-quote or single-quote characters). The actual representation of characters depends on the `encoding` attribute defined for the `<?xml>` processing instruction.

If this value contains reserved XML characters, it must be escaped using standard XML character escaping mechanisms.

5.2.3.1.2 Character Values

If the CIM type is `char`, the PCDATA value must be a single UCS-2 character. The value must not be surrounded by single-quote characters. If this value is a reserved XML character, it must be escaped using standard XML character escaping mechanisms. The actual representation of the character depends on the `encoding` attribute defined for the `<?xml>` processing instruction.

5.2.3.1.3 Real Values

If the CIM type is `real32` or `real64`, the PCDATA value must conform to the following syntax, where `decimalDigit` is any character from the set {`0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`}:

```
[ "+" | "-" ] *decimalDigit "." 1*decimalDigit [ ( "e" | "E" ) [ "+" | "-" ]
1*decimalDigit ]
```

The basis for the exponent must be 10. The significand must be represented with a precision of at least 9 decimal digits for real32 and at least 17 digits for real64. Trailing zeros in the fractional part and leading zeros in the whole part of the significand may be omitted. The exponent must be represented with a precision of at least 3 decimal digits for real32 and at least 4 digits for real64. Leading zeros in the exponent may be omitted.

NOTE: This definition of a minimum precision guarantees that the value of CIM real types in their binary representation (defined by IEEE 754) does not change when converting it to the decimal representation and back to the binary representation.

5.2.3.1.4 Boolean Values

If the CIM type is `boolean`, the PCDATA value must be either TRUE or FALSE. These values must be treated as case-insensitive.

5.2.3.1.5 Integer Values

If the CIM type belongs to the set {`uint8`, `uint16`, `uint32`, `uint64`}, the PCDATA value must be a valid unsigned decimal or hexadecimal value.

If the CIM type belongs to the set {`sint8`, `sint16`, `sint32`, `sint64`}, the PCDATA value must be a valid signed decimal or hexadecimal value.

Decimal values have the following format, where `decimalDigit` is any character from the set {`0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`} and `positiveDecimalDigit` is any decimal digit other than `0`:

```
[ "+" | "-" ] ( positiveDecimalDigit *decimalDigit | "0" )
```

The leading sign character must not be used when the CIM type is unsigned.

Hexadecimal values have the following format, where `hexDigit` is either a `decimalDigit` or a character from the set {`a`, `A`, `b`, `B`, `c`, `C`, `d`, `D`, `e`, `E`, `f`, `F`}:

```
[ "+" | "-" ] ( "0x" | "0X" ) 1*hexDigit
```

The leading sign character must not be used when the CIM type is unsigned.

5.2.3.1.6 Datetime Values

If the CIM type is `Datetime`, the PCDATA value must be a valid datetime value as defined in detail by DSP0004. (For interval values, the format is `ddddddddhhmmss.mmmmmm:000`; for absolute values, the format is `yyyymmddhhmmss.mmmmmmsutc`.)

The value must not be surrounded by string delimiter characters (such as double-quote or single-quote characters).

5.2.3.2 VALUE.ARRAY

The VALUE.ARRAY element is used to represent the value of a CIM property or qualifier that has an array type.

CIM arrays are classified as "Bag", "Ordered", or "Indexed" (refer to DSP0004) using the ARRAYTYPE qualifier. If the array is Ordered or Indexed, the subelements of VALUE.ARRAY must appear in the order of the array entries.

If the value of an array entry is NULL, the VALUE.NULL subelement must be used to represent the array entry. Otherwise, the VALUE subelement must be used.

NOTE: For string datatypes, a VALUE element with an empty PCDATA value indicates an empty string (that is, "").

```
<!ELEMENT VALUE.ARRAY (VALUE|VALUE.NULL)*>
```

5.2.3.3 VALUE.REFERENCE

The VALUE.REFERENCE element is used to define a single CIM reference property value.

If a LOCALCLASSPATH or LOCALINSTANCEPATH subelement is used, the target object is assumed to be on the same host. If a CLASSNAME or INSTANCENAME subelement is used, the target object is assumed to be in the same namespace.

```
<!ELEMENT VALUE.REFERENCE
(CLASSPATH|LOCALCLASSPATH|CLASSNAME|INSTANCEPATH|LOCALINSTANCEPATH|INSTANCENAME)>
```

5.2.3.4 VALUE.REFARRAY

The VALUE.REFARRAY element is used to represent the value of an array of CIM references.

CIM arrays are classified as "Bag", "Ordered", or "Indexed" (refer to DSP0004) using the ARRAYTYPE qualifier. If the array is Ordered or Indexed, the subelements must appear in the order of the array entries.

If the value of an array entry is NULL, the VALUE.NULL subelement must be used to represent the array entry. Otherwise, the VALUE.REFERENCE subelement must be used.

```
<!ELEMENT VALUE.REFARRAY (VALUE.REFERENCE|VALUE.NULL)*>
```

5.2.3.5 VALUE.OBJECT

The VALUE.OBJECT element is used to define a value that comprises a single CIM class or instance definition.

```
<!ELEMENT VALUE.OBJECT (CLASS|INSTANCE)>
```

5.2.3.6 VALUE.NAMEDINSTANCE

The VALUE.NAMEDINSTANCE element is used to define a value that comprises a single named CIM instance definition.

```
<!ELEMENT VALUE.NAMEDINSTANCE (INSTANCENAME,INSTANCE)>
```

5.2.3.7 VALUE.NAMEDOBJECT

The VALUE.NAMEDOBJECT element is used to define a value that comprises a single named CIM class or instance definition.

```
<!ELEMENT VALUE.NAMEDOBJECT (CLASS|(INSTANCENAME,INSTANCE))>
```

5.2.3.8 VALUE.OBJECTWITHPATH

The VALUE.OBJECTWITHPATH element is used to define a value that comprises a single CIM object (class or instance) definition with additional information that defines the absolute path to that object.

```
<!ELEMENT VALUE.OBJECTWITHPATH ((CLASSPATH,CLASS)|(INSTANCEPATH,INSTANCE))>
```

5.2.3.9 VALUE.OBJECTWITHLOCALPATH

The VALUE.OBJECTWITHLOCALPATH element is used to define a value that comprises a single CIM object (class or instance) definition with additional information that defines the local path to that object.

```
<!ELEMENT VALUE.OBJECTWITHLOCALPATH
((LOCALCLASSPATH,CLASS)|(LOCALINSTANCEPATH,INSTANCE))>
```

5.2.3.10 VALUE.NULL

The VALUE.NULL element is used to represent a NULL value.

NOTE: In some cases, omission of a subelement indicates the NULL value, instead of using VALUE.NULL.

```
<!ELEMENT VALUE.NULL EMPTY>
```

5.2.3.11 VALUE.INSTANCEWITHPATH

The VALUE.INSTANCEWITHPATH element is used to define a value that comprises a single CIM instance definition with additional information that defines the absolute path to that object.

```
<!ELEMENT VALUE.INSTANCEWITHPATH (INSTANCEPATH,INSTANCE)>
```

5.2.4 Naming and Location Elements

This clause defines those elements of the schema that are concerned with expressing the name and location of CIM objects.

5.2.4.1 NAMESPACEPATH

The NAMESPACEPATH element is used to define a namespace path. It consists of a HOST element and a LOCALNAMESPACEPATH element.

The NAMESPACE elements must appear in hierarchy order, with the root namespace appearing first.

```
<!ELEMENT NAMESPACEPATH (HOST,LOCALNAMESPACEPATH)>
```

5.2.4.2 LOCALNAMESPACEPATH

The LOCALNAMESPACEPATH element is used to define a local namespace path (one without a host component). It consists of one or more NAMESPACE elements (one for each namespace in the path).

```
<!ELEMENT LOCALNAMESPACEPATH (NAMESPACE+)>
```

5.2.4.3 HOST

The HOST element is used to define a single host. The element content must specify a legal value for a hostname in accordance with DSP0004.

```
<!ELEMENT HOST (#PCDATA)>
```

5.2.4.4 NAMESPACE

The NAMESPACE element is used to define a single namespace component of a namespace path.

```
<!ELEMENT NAMESPACE EMPTY>
<!ATTLIST NAMESPACE
    %CIMName;>
```

The `CIMName` attribute defines the name of the namespace.

5.2.4.5 CLASSPATH

The CLASSPATH element defines the absolute path to a CIM class. It is formed from a namespace path and class name.

```
<!ELEMENT CLASSPATH (NAMESPACEPATH,CLASSNAME)>
```

5.2.4.6 LOCALCLASSPATH

The LOCALCLASSPATH element defines the local path to a CIM class. It is formed from a local namespace path and class name.

```
<!ELEMENT LOCALCLASSPATH (LOCALNAMESPACEPATH, CLASSNAME)>
```

5.2.4.7 CLASSNAME

The CLASSNAME element defines the qualifying name of a CIM class.

```
<!ELEMENT CLASSNAME EMPTY>
<!ATTLIST CLASSNAME
    %CIMName;>
```

The `CIMName` attribute defines the name of the class.

5.2.4.8 INSTANCEPATH

The INSTANCEPATH element defines the absolute path to a CIM instance. It comprises a namespace path and an instance name (model path).

```
<!ELEMENT INSTANCEPATH (NAMESPACEPATH,INSTANCENAME)>
```

5.2.4.9 LOCALINSTANCEPATH

The LOCALINSTANCEPATH element defines the local path to a CIM instance. It comprises a local namespace path and an instance name (model path).

```
<!ELEMENT LOCALINSTANCEPATH (LOCALNAMESPACEPATH,INSTANCENAME)>
```

5.2.4.10 INSTANCENAME

The INSTANCENAME element defines the location of a CIM instance within a namespace (it is referred to in DSP0004 as a model path). It comprises a class name and key-binding information.

If the class has a single key property, a single KEYVALUE or VALUE.REFERENCE subelement may be used to describe the (necessarily) unique key value without a key name. Alternatively, a single KEYBINDING subelement may be used instead.

If the class has more than one key property, a KEYBINDING subelement must appear for each key.

If no key-bindings are specified, the instance is assumed to be a singleton instance of a keyless class.

```
<!ELEMENT INSTANCENAME (KEYBINDING*|KEYVALUE?|VALUE.REFERENCE?)>
<!ATTLIST INSTANCENAME
    %ClassName;>
```

The `ClassName` attribute defines the name of the class for this path.

5.2.4.11 OBJECTPATH

The OBJECTPATH element is used to define a full path to a single CIM object (class or instance).

```
<!ELEMENT OBJECTPATH (INSTANCEPATH|CLASSPATH)>
```

5.2.4.12 KEYBINDING

The KEYBINDING element defines a single key property value binding.

```
<!ELEMENT KEYBINDING (KEYVALUE|VALUE.REFERENCE)>
<!ATTLIST KEYBINDING
```

```
        %CIMName;>
```

The CIMName attribute indicates the name of the key property.

5.2.4.13 KEYVALUE

The KEYVALUE element defines a single property key value when the key property is a non-reference type.

```
<!ELEMENT KEYVALUE (#PCDATA)>
<!ATTLIST KEYVALUE
     VALUETYPE  (string|boolean|numeric) "string"
     %CIMType;   #IMPLIED>
```

Because a value's type cannot be validated using DTD, each value appears in PCDATA format irrespective of the type. The data type of the underlying key property determines the format of the PCDATA value. The rules for how the content of this element is formatted depending on that data type are exactly the same as for the VALUE element.

The VALUETYPE attribute provides information regarding the data type to allow the transformation of the key value to and from its textual equivalent (as part of a text-based CIM object path, as defined in DSP0004). The value of this attribute must conform to the following rules:

- If the CIM type is string, datetime, or char16, the value is `string`.
- If the CIM type is boolean, the value is `boolean`.
- Otherwise, the value is `numeric`.

The CIMType attribute is optional and, when provided, can be used to improve performance. If specified, the CIMType attribute must be the data type of the underlying key property.

5.2.5 Object Definition Elements

This section defines those elements of the schema that are concerned with expressing the definition of CIM objects (classes, instances, properties, methods, and qualifiers).

5.2.5.1 CLASS

The CLASS element defines a single CIM class.

```
<!ELEMENT CLASS
(QUALIFIER*,(PROPERTY|PROPERTY.ARRAY|PROPERTY.REFERENCE)*,METHOD*)>
<!ATTLIST CLASS
     %CIMName;
     %SuperClass;>
```

The CIMName attribute defines the name of the class.

The SuperClass attribute, if present, defines the name of the superclass of this class. If this attribute is absent, it should be inferred that the class in question has no superclass.

5.2.5.2 INSTANCE

The INSTANCE element defines a single CIM instance of a CIM class.

The instance must contain only properties defined in or inherited by the CIM class. Not all these properties are required to be present in an instance. (This is in accordance with the requirement that CIM instances have all properties defined in or inherited by the CIM class, because an <INSTANCE> is only a copied representation of the CIM instance, in a particular context). Specifications using the mapping defined in this document must define the rules for any properties that are not present.

```
<!ELEMENT INSTANCE (QUALIFIER*,(PROPERTY|PROPERTY.ARRAY|PROPERTY.REFERENCE)*)>
<!ATTLIST INSTANCE
    %ClassName;
    xml:lang  NMTOKEN #IMPLIED>
```

The `ClassName` attribute defines the name of the CIM class of which this is an instance.

5.2.5.3 QUALIFIER

The QUALIFIER element defines a single CIM qualifier. If the qualifier has a non-array type, it contains a single VALUE element that represents the value of the qualifier. If the qualifier has an array type, it contains a single VALUE.ARRAY element to represent the value.

If the qualifier has no assigned value (that is, it was specified without a value), the VALUE and VALUE.ARRAY subelements must be absent. DSP0004 defines how to interpret this case, dependent on the CIM datatype.

```
<!ELEMENT QUALIFIER ((VALUE|VALUE.ARRAY)?)>
<!ATTLIST QUALIFIER
    %CIMName;
    %CIMType;             #REQUIRED
    %Propagated;
    %QualifierFlavor;
    xml:lang NMTOKEN      #IMPLIED>
```

The `CIMName` attribute defines the name of the qualifier, and the `CIMType` attribute defines the CIM type.

5.2.5.4 PROPERTY

The PROPERTY element defines the value in a CIM instance or the definition in a CIM class of a single (non-array) CIM property that is not a reference.

CIM reference properties are described using the PROPERTY.REFERENCE element.

```
<!ELEMENT PROPERTY (QUALIFIER*,VALUE?)>
<!ATTLIST PROPERTY
    %CIMName;
    %CIMType;             #REQUIRED
    %ClassOrigin;
    %Propagated;
    %EmbeddedObject;
    xml:lang  NMTOKEN     #IMPLIED>
```

A VALUE subelement must be present if the property value or the default value of the property definition is non-NULL. Absence of the VALUE subelement must be interpreted as a value of NULL.

The `CIMName` attribute defines the name of the property, and the `CIMType` attribute defines the CIM type.

If the class definition for the property includes the EMBEDDEDOBJECT or EMBEDDEDINSTANCE qualifier, the corresponding `EmbeddedObject` attribute and `EmbeddedClassName` attribute must be included for properties in instances of that class. These attributes must not be attached to class elements.

- A property that is defined in MOF as an EmbeddedObject with the inclusion of the EmbeddedObject qualifier on the property must be represented using the attribute EmbeddedObject with the value "object". The value must be a valid INSTANCE element, defining a single CIM instance of a CIM class or a valid CLASS element.
- A property that is defined in MOF as an EmbeddedInstance with the inclusion of the EmbeddedInstance qualifier on a property must be represented using the attribute

EmbeddedObject with the value "instance". The value must be a valid INSTANCE element, defining a single CIM instance.

5.2.5.5 PROPERTY.ARRAY

The PROPERTY.ARRAY element defines the value in a CIM instance or the definition in a CIM class of a single CIM property with an array type.

There is no element to model a property that contains an array of references because this is not a valid property type according to DSP0004.

```
<!ELEMENT PROPERTY.ARRAY (QUALIFIER*,VALUE.ARRAY?)>
<!ATTLIST PROPERTY.ARRAY
    %CIMName;
    %CIMType;              #REQUIRED
    %ArraySize;
    %ClassOrigin;
    %Propagated;
    %EmbeddedObject;
    xml:lang  NMTOKEN      #IMPLIED>
```

A VALUE.ARRAY subelement must be present if the property value (that is, the array itself) or the default value of the property definition (that is, the array itself) is non-NULL. Absence of the VALUE.ARRAY subelement must be interpreted as a value of NULL.

The `CIMName` attribute defines the name of the property, and the `CIMType` attribute defines the CIM type.

If the `ArraySize` attribute is not present on a PROPERTY.ARRAY element within a containing CLASS element, the array is of variable size.

The presence or absence of the `ArraySize` attribute on a PROPERTY.ARRAY element within a containing INSTANCE element must not be interpreted as meaning that the property type is or is not a fixed-size array (that is, the CLASS definition is always authoritative in this respect).

If the class definition for the property includes the EMBEDDEDOBJECT or EMBEDDEDINSTANCE qualifier, the corresponding `EmbeddedObject` attribute must be included.

1) A property that is defined in MOF as an EmbeddedObject with the inclusion of the EmbeddedObject qualifier on the property must be defined using the type "object". The value must be a valid INSTANCE element, defining a single CIM instance of a CIM class or a valid CLASS element.

2) A property that is defined in MOF as an EmbeddedInstance with the inclusion of the EmbeddedInstance qualifier on a property must be defined as the type "instance". The value must be a valid INSTANCE element, defining a single CIM instance.

5.2.5.6 PROPERTY.REFERENCE

The PROPERTY.REFERENCE element defines the value in a CIM instance or the definition in a CIM class of a single CIM property with reference semantics. In the future, the features of XML Linking may be used to identify linking elements within the XML document.

```
<!ELEMENT PROPERTY.REFERENCE (QUALIFIER*,VALUE.REFERENCE?)>
<!ATTLIST PROPERTY.REFERENCE
    %CIMName;
    %ReferenceClass;
    %ClassOrigin;
    %Propagated;>
```

The VALUE.REFERENCE subelement must be present if the property value or the default value of the property definition is non-NULL. Absence of the VALUE.REFERENCE subelement must be interpreted as a value of NULL.

The `CIMName` attribute defines the name of the property.

The `ReferenceClass` attribute, if present, defines the strong type of the reference. The absence of this attribute indicates that this reference is not strongly typed. The expected behavior is that the `ReferenceClass` attribute must exist for PROPERTY.REFERENCE usage in class entities and should not exist for instance entities because the reference class name should be defined in the property value.

The `ClassOrigin` and `Propagated` entities are used in the same manner as for other CIM properties.

5.2.5.7 METHOD

The METHOD element defines a single CIM method. It may have qualifiers, and zero or more parameters.

The order of the PARAMETER, PARAMETER.REFERENCE, PARAMETER.ARRAY and PARAMETER.REFARRAY subelements is not significant.

```
<!ELEMENT METHOD
(QUALIFIER*,(PARAMETER|PARAMETER.REFERENCE|PARAMETER.ARRAY|PARAMETER.REFARRAY)*)>
<!ATTLIST METHOD
    %CIMName;
    %CIMType;     #IMPLIED
    %ClassOrigin;
    %Propagated;>
```

The `CIMName` attribute defines the name of the method.

The `CIMType` attribute defines the method return type, if the method returns a value. If this attribute is absent, the method must return no value (that is, it has the special return type `void`).

5.2.5.8 PARAMETER

The PARAMETER element defines a single (non-array, non-reference) parameter to a CIM method. The parameter may have zero or more qualifiers.

```
<!ELEMENT PARAMETER (QUALIFIER*)>
<!ATTLIST PARAMETER
    %CIMName;
    %CIMType;   #REQUIRED>
```

The `CIMName` attribute defines the name of the parameter. The `CIMType` attribute defines the CIM type of the parameter.

5.2.5.9 PARAMETER.REFERENCE

The PARAMETER.REFERENCE element defines a single reference parameter to a CIM method. The parameter may have zero or more qualifiers.

```
<!ELEMENT PARAMETER.REFERENCE (QUALIFIER*)>
<!ATTLIST PARAMETER.REFERENCE
    %CIMName;
    %ReferenceClass;>
```

The `CIMName` attribute defines the name of the parameter.

The `ReferenceClass` attribute, if present, defines the strong type of the reference. If this attribute is absent, the parameter is assumed to be a reference that is not strongly typed.

The expected behavior is that the `ReferenceClass` attribute must exist for PARAMETER.REFERENCE entities.

5.2.5.10 PARAMETER.ARRAY

The PARAMETER.ARRAY element defines a single parameter to a CIM method that has an array type. The parameter may have zero or more qualifiers.

```
<!ELEMENT PARAMETER.ARRAY (QUALIFIER*)>
<!ATTLIST PARAMETER.ARRAY
    %CIMName;
    %CIMType;              #REQUIRED
    %ArraySize;>
```

The `CIMName` attribute defines the name of the parameter. The `CIMType` attribute defines the CIM type of the parameter.

The `ArraySize` attribute is present if the array is constrained to a fixed number of elements. If the attribute has empty content, the array is of variable size.

5.2.5.11 PARAMETER.REFARRAY

The PARAMETER.REFARRAY element defines a single parameter to a CIM method that has an array of references type. The parameter may have zero or more qualifiers.

```
<!ELEMENT PARAMETER.REFARRAY (QUALIFIER*)>
<!ATTLIST PARAMETER.REFARRAY
    %CIMName;
    %ReferenceClass;
    %ArraySize;>
```

The `CIMName` attribute defines the name of the parameter.

The `ReferenceClass` attribute defines the strong type of a reference. If this attribute is absent, the parameter is not a strongly typed reference. The expected behavior is that the `ReferenceClass` attribute must exist for PARAMETER.REFARRAY entities.

The `ArraySize` attribute is present if the array is constrained to a fixed number of elements. If this attribute is absent, the array is of variable size.

5.2.6 Message Elements

This section defines those elements of the schema that are concerned with expressing the definition of CIM messages for DSP0200.

5.2.6.1 MESSAGE

The MESSAGE element models a single CIM message. This element is used as the basis for CIM Operation Messages and CIM Export Messages.

```
<!ELEMENT MESSAGE (SIMPLEREQ|MULTIREQ|SIMPLERSP|MULTIRSP|
SIMPLEEXPREQ|MULTIEXPREQ|SIMPLEEXPRSP|MULTIEXPRSP)>
<!ATTLIST MESSAGE
    ID       CDATA            #REQUIRED
    PROTOCOLVERSION CDATA     #REQUIRED>
```

The ID attribute defines an identifier for the MESSAGE element. The content of the value is not constrained by this specification, but the intention is that ID attribute be used as a correlation mechanism between two CIM entities.

The PROTOCOLVERSION attribute defines the version of DSP0200 to which this message conforms. It must be in the form of "M.N", where M is the major version of the specification in numeric form, and N is the minor version of the specification in numeric form (for example, 1.0, 1.1). Implementations must validate only the major version because all minor versions are backward compatible. Implementations may look at the minor version to determine additional capabilities.

DSP0200 provides more details on the values that these attributes may take.

5.2.6.2 MULTIREQ

The MULTIREQ element defines a multiple CIM operation request. It contains two or more subelements that define the SIMPLEREQ elements that make up this multiple request.

```
<!ELEMENT MULTIREQ (SIMPLEREQ,SIMPLEREQ+)>
```

5.2.6.3 SIMPLEREQ

The SIMPLEREQ element defines a simple CIM operation request. It contains either a METHODCALL (extrinsic method) element or an IMETHODCALL (intrinsic method) element.

```
<!ELEMENT SIMPLEREQ (METHODCALL|IMETHODCALL)>
```

5.2.6.4 METHODCALL

The METHODCALL element defines a single method invocation on a class or instance. It specifies the local path of the target class or instance, followed by zero or more PARAMVALUE subelements as the parameter values to be passed to the method.

```
<!ELEMENT METHODCALL ((LOCALCLASSPATH|LOCALINSTANCEPATH),PARAMVALUE*)>
<!ATTLIST METHODCALL
    %CIMName;>
```

The CIMName attribute defines the name of the method to be invoked.

5.2.6.5 PARAMVALUE

The PARAMVALUE element defines a single method named parameter value. The absence of a subelement indicates that the parameter has the NULL value.

```
<!ELEMENT PARAMVALUE (
VALUE|VALUE.REFERENCE|VALUE.ARRAY|VALUE.REFARRAY|CLASSNAME|INSTANCENAME|
CLASS|INSTANCE|VALUE.NAMEDINSTANCE)?>
<!ATTLIST PARAMVALUE
    %CIMName;
    %ParamType;
    %EmbeddedObject;>
```

The CIMName attribute defines the name of the parameter. The ParamType attribute defines the type of the parameter.

The EmbeddedObject attribute defines that this PARAMVALUE represents a CIM embedded object. This attribute may be applied only to string types and represents a parameter that has the EMBEDDEDOBJECT or EMBEDDEDINSTANCE qualifier attached.

5.2.6.6 IMETHODCALL

The IMETHODCALL element defines a single intrinsic method invocation. It specifies the target local namespace, followed by zero or more IPARAMVALUE subelements as the parameter values to be passed to the method.

```
<!ELEMENT IMETHODCALL (LOCALNAMESPACEPATH,IPARAMVALUE*)>
<!ATTLIST IMETHODCALL
    %CIMName;>
```

The CIMName attribute defines the name of the method to be invoked.

5.2.6.7 IPARAMVALUE

The IPARAMVALUE element defines a single intrinsic method named parameter value. The absence of a subelement indicates that the parameter has the NULL value.

```
<!ELEMENT IPARAMVALUE
(VALUE|VALUE.ARRAY|VALUE.REFERENCE|CLASSNAME|INSTANCENAME|QUALIFIER.DECLARATION|
CLASS|INSTANCE|VALUE.NAMEDINSTANCE)?>
<!ATTLIST IPARAMVALUE
    %CIMName;>
```

The CIMName attribute defines the name of the parameter.

5.2.6.8 MULTIRSP

The MULTIRSP element defines a multiple CIM operation response. It contains two or more subelements that define the SIMPLERSP elements that make up this multiple response.

```
<!ELEMENT MULTIRSP (SIMPLERSP,SIMPLERSP+)>
```

5.2.6.9 SIMPLERSP

The SIMPLERSP element defines a simple CIM operation response. It contains either a METHODRESPONSE (for extrinsic methods) element or an IMETHODRESPONSE (for intrinsic methods) element.

```
<!ELEMENT SIMPLERSP (METHODRESPONSE|IMETHODRESPONSE)>
```

5.2.6.10 METHODRESPONSE

The METHODRESPONSE element defines the response to a single CIM extrinsic method invocation. It contains either an ERROR subelement (to report a fundamental error that prevented the method from executing) or a combination of an optional return value and zero or more out parameter values.

```
<!ELEMENT METHODRESPONSE (ERROR|(RETURNVALUE?,PARAMVALUE*))>
<!ATTLIST METHODRESPONSE
    %CIMName;>
```

The CIMName attribute defines the name of the method that was invoked.

5.2.6.11 IMETHODRESPONSE

The IMETHODRESPONSE element defines the response to a single intrinsic CIM method invocation. It contains either an ERROR subelement (to report a fundamental error that prevented the method from executing) or an optional return value and zero or more out parameter values.

```
<!ELEMENT IMETHODRESPONSE (ERROR|IRETURNVALUE?, PARAMVALUE*)>
<!ATTLIST IMETHODRESPONSE
    %CIMName;>
```

867 The CIMName attribute defines the name of the method that was invoked.

868 #### 5.2.6.12 ERROR

869 The ERROR element is used to define a fundamental error that prevented a method from executing
870 normally. It consists of a status code, an optional description, and zero or more instances that contain
871 detailed information about the error.

```
872     <!ELEMENT ERROR (INSTANCE*)
873     <!ATTLIST ERROR
874         CODE          CDATA    #REQUIRED
875         DESCRIPTION   CDATA    #IMPLIED>
```

876 The CODE attribute contains a numerical status code that indicates the nature of the error. The valid
877 status codes are defined in DSP0004. The DESCRIPTION attribute, if present, provides a human-
878 readable description of the error.

879 #### 5.2.6.13 RETURNVALUE

880 The RETURNVALUE element specifies the value returned from an extrinsic method call. The absence of
881 a subelement indicates that the return value has the NULL value.

```
882     <!ELEMENT RETURNVALUE (VALUE|VALUE.REFERENCE)?>
883     <!ATTLIST RETURNVALUE
884         %EmbeddedObject;
885         %ParamType;       #IMPLIED>
```

886 The ParamType attribute defines the type of the return value.

887 The EmbeddedObject attribute defines that this RETURNVALUE element represents a CIM embedded
888 object. This attribute may be applied only to string types and represents a parameter that has the
889 EMBEDDEDOBJECT or EMBEDDEDINSTANCE qualifier attached.

890 #### 5.2.6.14 IRETURNVALUE

891 The IRETURNVALUE element specifies the value returned from an intrinsic method call. The absence of
892 a subelement indicates that the return value has the NULL value.

```
893     <!ELEMENT IRETURNVALUE
894     (CLASSNAME*|INSTANCENAME*|VALUE*|VALUE.OBJECTWITHPATH*|VALUE.OBJECTWITHLOCALPATH*
895     VALUE.OBJECT*|OBJECTPATH*|QUALIFIER.DECLARATION*|VALUE.ARRAY?|VALUE.REFERENCE?|
896     CLASS*|INSTANCE*|VALUE.NAMEDINSTANCE*)>
```

897 #### 5.2.6.15 MULTIEXPREQ

898 The MULTIEXPREQ element defines a multiple CIM export request. It contains two or more subelements
899 that define the SIMPLEEXPREQ elements that make up this multiple request.

```
900     <!ELEMENT MULTIEXPREQ (SIMPLEEXPREQ,SIMPLEEXPREQ+)>
```

901 #### 5.2.6.16 SIMPLEEXPREQ

902 The SIMPLEEXPREQ element defines a simple CIM export request. It contains an EXPMETHODCALL
903 (export method) subelement.

```
904     <!ELEMENT SIMPLEEXPREQ (EXPMETHODCALL)>
```

905 #### 5.2.6.17 EXPMETHODCALL

906 The EXPMETHODCALL element defines a single export method invocation. It specifies zero or more
907 EXPPARAMVALUE subelements as the parameter values to be passed to the method.

```
<!ELEMENT EXPMETHODCALL (EXPPARAMVALUE*)>
<!ATTLIST EXPMETHODCALL
    %CIMName;>
```

The CIMName attribute defines the name of the export method to be invoked.

5.2.6.18 MULTIEXPRSP

The MULTIEXPRSP element defines a multiple CIM export response. It contains two or more subelements that define the SIMPLEEXPRSP elements that make up this multiple response.

```
<!ELEMENT MULTIEXPRSP (SIMPLEEXPRSP,SIMPLEEXPRSP+)>
```

5.2.6.19 SIMPLEEXPRSP

The SIMPLEEXPRSP element defines a simple CIM export response. It contains an EXPMETHODRESPONSE (for export methods) subelement.

```
<!ELEMENT SIMPLEEXPRSP (EXPMETHODRESPONSE)>
```

5.2.6.20 EXPMETHODRESPONSE

The EXPMETHODRESPONSE element defines the response to a single export method invocation. It contains either an ERROR subelement (to report a fundamental error that prevented the method from executing) or an optional return value.

```
<!ELEMENT EXPMETHODRESPONSE (ERROR|IRETURNVALUE?)>
<!ATTLIST EXPMETHODRESPONSE
    %CIMName;>
```

The CIMName attribute defines the name of the export method that was invoked.

5.2.6.21 EXPPARAMVALUE

The EXPPARAMVALUE element defines a single export method named parameter value. The absence of a subelement indicates that the parameter has the NULL value.

```
<!ELEMENT EXPPARAMVALUE (INSTANCE?)>
<!ATTLIST EXPPARAMVALUE
    %CIMName;>
```

The CIMName attribute defines the name of the parameter.

5.2.6.22 ENUMERATIONCONTEXT

The ENUMERATIONCONTEXT element is used to define the context of an enumeration operation to be passed between the client and the server during the life of a Pull enumeration.

```
<!ELEMENT ENUMERATIONCONTEXT (#PCDATA)>
```

The data in the ENUMERATIONCONTEXT element is to be considered opaque data by the client. If this value contains reserved XML characters, it must be escaped using standard XML character escaping mechanisms.

ANNEX A
(informative)

Change History

Version	Date	Description
Version 2.0.0	June 2, 1999	Final
Version 2.2.0	January 11, 2007	Final
Version 2.3.0	November 11, 2008	DMTF Standard
Version 2.3.1	July 29, 2009	DMTF Standard