Document Number: DSP0201

Date: 2014-01-16

Version: 2.4.0

Representation of CIM in XML

Document Type: Specification

Document Status: DMTF Standard

Document Language: en-US

CONTENTS

Tables

Foreword

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

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-XML messages for use by DSP0200 (*CIM Operations over HTTP*).

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

The same XML grammar is also described using XSD as DSP8044.

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-XML 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-XML 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 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

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

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

191 • It requires only one standardized metaschema DTD for CIM rather than an unbounded number
192 of DTDs. This considerably reduces the complexity of management and administration of XML
193 mappings.
194 • An XML DTD does not allow an unordered list of elements. In a static mapping, this restriction
195 would require one of the following actions:

196 – Fixing an arbitrary order for property, method, and qualifier lists (making it harder for a
197 receiving application to process)
198 – Defining a very unwieldy mapping that accounts for all list orderings explicitly (and whose
199 size would grow exponentially with the number of list elements)

200 • In a schema mapping, the names of CIM schema elements (class, property, qualifier, and
201 method names) populate the XML element namespace. To replicate the scoping rules on CIM
202 element names within an XML DTD, it would be necessary to employ XML namespaces to
203 define XML schema to a per-property level of granularity. This would be extremely cumbersome
204 to administer and process. A metaschema mapping introduces only a small, fixed number of
205 terms into the XML element namespace (such as Class, Instance, Property, and so on). As an
206 alternative to the introduction of additional XML namespaces, some renaming of CIM elements
207 could be used (for example, prefixing a qualifier name with the name of its owning property and
208 its owning class), but this would result in XML documents that are verbose and difficult to
209 understand.
210 • Although a schema mapping could allow XML-based validation of instances against classes,
211 this would be possible only if the entire class hierarchy were flattened prior to mapping the CIM
212 class to an XML schema. If this flattening was not performed, inherited properties might be
213 absent from the DTD, which would cause validation to fail against an instance that included the
214 value of an inherited property.

215

Representation of CIM in XML

1 Scope

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

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

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

2 Normative References

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

ANSI/IEEE 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*, August 1985, http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=30711

DMTF DSP0004, *Common Information Model (CIM) Infrastructure 2.7*, http://www.dmtf.org/standards/published_documents/DSP0004_2.7.pdf

DMTF DSP0200, *CIM Operations over HTTP 1.4*, http://www.dmtf.org/standards/published_documents/DSP0200_1.4.pdf

IETF RFC1034 *Domain Names - Concepts and Facilities*, November 1987, http://tools.ietf.org/html/rfc1034

IETF RFC3986, Uniform Resource Identifiers (URI): Generic Syntax, August 1998 344 http://tools.ietf.org/html/rfc2396

ISO/IEC 10646:2003, *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*, http://standards.iso.org/ittf/PubliclyAvailableStandards/c039921_ISO_IEC_10646_2003(E).zip

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

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

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

3 Terms and Definitions

In this document, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause.

The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"), "may", "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described

in ISO/IEC Directives, Part 2, Annex H. The terms in parenthesis are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Annex H specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 5.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The terms defined in DSP0004 and DSP0200 apply to this document. The following additional terms are used in this document. Some additional more detailed terms are defined throughout the subclauses of this document.

3.1

CIM element

one of the following components of the CIM metamodel used to define a schema: Class, instance, property, method, parameter, or qualifier

3.2

CIM object

a namespace, class, instance, or qualifier that is defined in a CIM-XML declaration or accessible in a WBEM server.

3.3

CIM-XML declaration

a declaration of CIM objects (classes, instances and qualifiers), using the DECLARATION element defined in this specification. Note that "CIM declaration" was used for this term before version 2.4 of this specification.

3.4

CIM-XML message

a request or response message in the CIM-XML protocol, using the MESSAGE element defined in this specification. Note that "CIM message" was used for this term before version 2.4 of this specification.

3.5

CIM-XML consumer

a WBEM server, client or listener that receives a CIM-XML message, or a program that consumes a CIM-XML declaration

3.6

CIM-XML producer

a WBEM server, client or listener that sends a CIM-XML message, or a program that produces a CIM-XML declaration

3.7

CIM-XML protocol

the WBEM protocol that uses the CIM operations over HTTP defined in DSP0200 and the representation of CIM in XML defined in this specification

3.8
CIM-XML schema
the XML schema for representing CIM in XML, defined by this specification

3.9
numeric character reference
an escaped UCS character, as defined in clause 4.1 of the W3C XML specification as *character reference*, using the hexadecimal or decimal representation of its UCS code point. Examples are `"&#x0D;"` for the carriage return character or `"&#65;"` for the character `"A"`.

3.10
UCS character
A character from the Universal Multiple-Octet Coded Character Set (UCS) defined in ISO/IEC 10646. Also known as Unicode character. For an overview on UCS characters in CIM, see DSP0004.

3.11
whitespace
one or more consecutive occurrences of any of the characters space (U+0020), carriage return (U+000D), line feed (U+000A) or horizontal tab (U+0009), consistent with the whitespace definition in the W3C XML specification.

3.12
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, defined in DSP0004

4.2
DTD
Document Type Definition, defined in the W3C XML specification

4.3
MOF
Managed Object Format, defined in DSP0004

4.4
XML
Extensible Markup Language, defined in the W3C XML specification

4.5
XSD
XML Schema Definition, defined in the W3C XSD specifications

5 CIM-XML Schema Reference

This clause describes the CIM-XML schema using DTD. DSP0203 defines the same DTD as this specification without any annotations. DSP8044 defines the CIM-XML schema using XSD.

In case of differences between these three documents, this specification (DSP0201) overrules the other two.

5.1 General

5.1.1 Escaping, Whitespace Handling, Character Repertoire

This clause defines the rules for escaping of CIM values in CIM-XML.

This clause uses the term "CIM value" to refer to values at the CIM level (e.g. the value of a property of a CIM instance). Values at the level of XML elements and attributes are referred to as "content of the element" or "attribute value", consistent with the W3C XML specification (for example, the content of a VALUE element may represent the value of a property of a CIM instance).

5.1.1.1 XML Clarifications and Amendments

The rules for escaping and handling of whitespace defined in the W3C XML specification apply to this specification, with the following clarifications and amendments:

- The W3C XML specification uses the terms "XML processor" and "application". CIM-XML does not distinguish software layers within a WBEM client, server or listener.

  In this specification, any definitions in the W3C XML specification that use these two terms shall be interpreted as if both of these software layers were within the WBEM client, server or listener.

- The W3C XML specification defines in clause 2.10 that any whitespace not in XML markup shall be passed by the XML processor to the application. Further, it defines in clause 2.10 that all characters shall be preserved. In clause 2.11, it defines that carriage return characters (U+000D) with and without directly following line feed characters (U+000A) shall be converted into line feed characters when processing external parsed entities (that is, XML files).

  This specification defines certain elements and attributes or certain uses thereof as character-preserving or whitespace-tolerant. Clauses 5.1.1.2 and 5.1.1.3 define the escaping and whitespace handling rules for character-preserving and whitespace-tolerant elements and attributes.

- The W3C XML specification defines in clause 2.4 provisions for escaping. It does not explicitly state how those provisions apply in the context of a protocol that uses XML in its payload., and it does not explicitly state whether any character may be escaped using numeric character references (vs. just certain special characters).

  This specification defines rules on escaping in 5.1.1.4.

- The W3C XML specification defines in clause 2.7 that character data may be represented using CDATA sections (`<![CDATA[ ... ]]>`), but it is not clear from the text whether attribute values are considered character data for this purpose. The syntax in its BNF rule [10] clarifies that attribute values cannot be represented using CDATA sections.

  This specification points that out in 5.1.1.4.3.

5.1.1.2 Character-preserving Elements and Attributes

This specification defines certain XML elements and attributes or uses thereof to be character-preserving.

For character-preserving elements, attributes, or uses thereof, the following applies:

- CIM-XML producers shall set the content of the element or the value of the attribute from the CIM value with only the following transformation:
  1) Escaping as defined in 5.1.1.4.
- CIM-XML consumers shall set the CIM value from the content of the element or the value of the attribute with only the following transformation:
  1) Unescaping as defined in 5.1.1.4.

NOTE: For character-preserving elements and attributes, all characters are preserved, including any kind of whitespace characters and specifically including any carriage return characters (U+000D), regardless of their position within a string (that is, even if they are followed by line feed, U+000A).

5.1.1.3 Whitespace-tolerant Elements and Attributes

Any XML elements, attributes, or uses thereof not explicitly defined to be character-preserving (see 5.1.1.2) are considered whitespace-tolerant.

For whitespace-tolerant elements, attributes, or uses thereof, the following applies:

- CIM-XML producers shall set the content of the element or the value of the attribute from the CIM value with only the following transformations, in the stated order:

DEPRECATED

1) Adding zero or more leading or trailing whitespace characters of the following set of characters: U+0009 (horizontal tab), U+000A (line feed), U+0020 (space). Adding such leading or trailing characters is discouraged and is described only for compatibility with implementations of earlier versions of this specification.

DEPRECATED

2) Escaping as defined in 5.1.1.4.

- CIM-XML consumers shall set the CIM value from the content of the element or the value of the attribute with only the following transformations, in the stated order:
  1) Unescaping as defined in 5.1.1.4.

DEPRECATED

2) Removing any leading or trailing characters from the set of whitespace characters defined in the previous list item about CIM-XML producers. For compatibility with implementations of earlier versions of this specification, such removal is recommended for an implementation even if the addition of such leading or trailing characters is not performed by that implementation.

DEPRECATED

5.1.1.4 Escaping of element content and attribute values

The provisions (that is, keywords like "shall", etc.) about escaping defined in clause 2.4 of the W3C XML specification shall be interpreted to apply to CIM-XML producers.

CIM-XML consumers shall support the unescaping of all forms of escaping permissible for CIM-XML producers, as defined in the W3C XML specification and in this specification.

5.1.1.4.1 Use of XML numeric character references

Clarifying clause 2.4 of the W3C XML specification, CIM-XML producers may escape any characters in the content of any elements and in the value of any attributes using XML numeric character references (e.g. `&#x26;` or `&#38;`).

Note that this includes all UCS characters valid for the element or attribute in question, not just those that may be escaped using XML entity references (see 5.1.1.4.2).

5.1.1.4.2 Use of XML entity references

Clarifying clause 2.4 of the W3C XML specification, CIM-XML producers may escape the respective characters in the content of any elements (except when within a CDATA section) and in the value of any attributes using the XML entity references defined in that clause:

```
&amp;
&lt;
&gt;
&apos;
&quot;
```

5.1.1.4.3 Use of CDATA sections

Restricting clause 2.7 of the W3C XML specification, CDATA sections may be used by CIM-XML producers only for representing all or part of the content of any elements defined as #PCDATA. CDATA sections shall not be used in the content of any other elements or in the value of any attributes.

Note that some CIM-XML elements are defined as #PCDATA but do not represent string-typed CIM values (for example, VALUE and KEYVALUE elements for boolean and numeric CIM values, or HOST).

Note that the W3C XML specification permits the use of multiple non-nested CDATA sections in the content of a single element, and the use of CDATA sections that cover only a subset of the element content. Using multiple CDATA sections may occur for example in the presence of nested embedded instances (see 5.1.1.4.5).

5.1.1.4.4 Combining different escaping mechanisms

Note that the W3C XML specification permits the combined use of numeric character references, entity references and CDATA sections in the content of the same element. Note that entity and character references can only be combined with CDATA sections if the CDATA section covers a subset of the character data and the references are used in the portion of the character data that is outside of the CDATA section. Entity and character references that occur in CDATA sections are not invalid, but they are interpreted as a sequence of characters, rather than a character reference.

Example:

The following VALUE element:

```
<VALUE>XML-escaped: &amp; &lt; &#x26; <![CDATA[CDATA section escaped: & <
&#38;]]></VALUE>
```

has the following CIM value after unescaping:

```
XML-escaped: & < & CDATA section escaped: & < &#38;
```

5.1.1.4.5 Nested embedded objects

An embedded object in the value of a property in an embedded instance is termed "nested embedded object". Such nesting may occur with arbitrary depth, whereby the non-leaf levels always are embedded instances and the leaf level may be an embedded instance or an embedded class.

Each level of nested embedded objects shall be escaped separately, treating the result of the previous escaping as input to the next level of escaping. What is escaped at each level, is always the value of the string-typed property defined as the embedded object or embedded instance, that is, the content of the VALUE element representing that property value, as described in 5.3.3.1.1.

Nested escaping using XML numeric character references (see 5.1.1.4.1) or entity references (see 5.1.1.4.2) works automatically by applying the escaping rules on the string-typed property value representing the embedded object.

Escaping using CDATA sections (see 5.1.1.4.3) requires a specific approach in order for it to work with nested embedded objects, because CDATA sections cannot simply be nested within each other. The character data that needs to be escaped again in context of an outer embedded instance already contains the CDATA section of the inner embedded object. One approach that works in this situation is to use two adjacent CDATA sections for the outer escaping that split each end marker (`]]>`) of the inner CDATA section such that its first portion (for example `]]`) goes into the first (outer) CDATA section and its remainder (in this case `>`) goes into the second (outer) CDATA section. This prevents the end parker of an inner CDATA section to end an outer CDATA section. The unchanged start marker of an inner CDATA section does not hurt, because once in the outer CDATA section, the inner start marker is encountered but is treated as normal character data.

Example with a nesting level of two:

Class definitions in MOF:

```
class A {
  [EmbeddedInstance("B")]
   string InstanceOfB;
};

class B {
  [EmbeddedInstance("C")]
   string InstanceOfC;
};

class C {
   string PropC;
};
```

Representation of an instance of class A using escaping with XML entity references. The color scheme indicates the levels of nesting. One can see that the number of nested applications of XML-escaping matches the nesting level of the instances:

```
<INSTANCE CLASSNAME="A">
  ...
  <PROPERTY NAME="InstanceOfB" TYPE="string" EmbeddedObject="instance">
    <VALUE>
      &lt;INSTANCE CLASSNAME="B"&gt;
        ...
        &lt;PROPERTY NAME="InstanceOfC" TYPE="string"
                    EmbeddedObject="instance"&gt;
          &lt;VALUE&gt;
            &amp;lt;INSTANCE CLASSNAME="B"&amp;gt;
              ...
              &amp;lt;PROPERTY NAME="PropC" TYPE="string"&amp;gt;
                &amp;lt;VALUE&amp;gt;a string&amp;lt;/VALUE&amp;gt;
              &amp;lt;/PROPERTY&amp;gt;
              ...
            &amp;lt;/INSTANCE&amp;gt;
          &lt;/VALUE&gt;
        &lt;/PROPERTY&gt;
        ...
      &lt;/INSTANCE&gt;
    </VALUE>
  </PROPERTY>
  ...
</INSTANCE>
```

Representation of an instance of class A using escaping with CDATA sections. One can see that the usage of the CDATA section for escaping the embedded instance of B in property A.InstanceOfB uses one simple application of CDATA escaping. However, an end marker was found in the data to be escaped. Therefore, two adjacent CDATA sections are used when escaping the value of property A.InstanceofB, whose boundary cuts the inner end marker into two parts.

```
<INSTANCE CLASSNAME="A">
  ...
  <PROPERTY NAME="InstanceOfB" TYPE="string" EmbeddedObject="instance">
    <VALUE>
      <![CDATA[                                    <- start marker of first outer
                                                      CDATA section
        <INSTANCE CLASSNAME="B">
          ...
          <PROPERTY NAME="InstanceOfC" TYPE="string"
                    EmbeddedObject="instance">
            <VALUE>
              <![CDATA[                            <- start marker of inner CDATA
                                                      section
                <INSTANCE CLASSNAME="B">
                  ...
                  <PROPERTY NAME="PropC" TYPE="string">
                    <VALUE>a string</VALUE>
                  </PROPERTY>
                  ...
                </INSTANCE>
                                                      end marker of inner CDATA
                ↓↓           ↓                    <- section, cut into two parts
                ]]]]><![CDATA[>
                  ↑↑↑↑↑↑↑↑↑↑↑↑                     <- end marker of first and
                                                      start marker of second
                                                      outer CDATA section
            </VALUE>
          </PROPERTY>
          ...
        </INSTANCE>
      ]]>                                          <- end marker of second outer
                                                      CDATA section
    </VALUE>
  </PROPERTY>
  ...
</INSTANCE>
```

5.1.1.4.6 Requirements for escaping

For better interoperability, the following rules on escaping apply to CIM-XML producers, in addition to the rules defined in the W3C XML specification:

- UCS characters in the range of U+0000 to U+001F should be escaped, using any of the escaping mechanisms described in 5.1.1.

5.1.1.5 Character Repertoire

The following rules about the repertoire of UCS characters apply for representing the CIM-XML payload:

- If the XML declaration of the CIM-XML payload specifies version="1.0":
  - The valid UCS characters in the range of U+0020 to U+10FFFF may be used.
  - The UCS characters U+0009, U+000A, U+000D may be used.
  - The other UCS characters in the range of U+0000 to U+001F shall not be used.
- If the XML declaration of the CIM-XML payload specifies version="1.1":
  - The valid UCS characters in the range of U+0001 to U+10FFFF may be used.
  - The UCS character U+0000 shall not be used.

Note that these rules are consistent with the W3C XML specification; XML 1.0 only supports representation of the UCS characters U+0009, U+000A, U+000D in the range of U+0000 to U+001F. XML 1.1 extends the support for that range to all characters except U+0000.

Note that DSP0004 permits string and char16 typed values to use all UCS characters in the range of U+0000 to U+001F. Thus, CIM-XML supports only a subset of that range, depending on the XML version used.

5.2 Entity Descriptions

This subclause 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.

5.2.1 CIMName

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

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

5.2.2 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)">
```

5.2.3 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'">
```

5.2.4 ClassOrigin

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

The originating class of a CIM property or method for the purpose of the ClassOrigin entity is the leaf-most class that defines or overrides the property or method.

The CLASSORIGIN attribute defines the name of the originating class of the CIM element represented by the XML element to which the attribute is attached.

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

5.2.5 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

5.2.6 ArraySize

The ArraySize entity is a convenient shorthand for the ARRAYSIZE attribute.

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

The ARRAYSIZE attribute defines the size of the array when it is a fixed-length array (see DSP0004). The value of this attribute (if it is present) shall be a positive integer.

5.2.7 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 shall be inferred that the owning element has no superclass.

5.2.8 ClassName

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

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

5.2.9 ReferenceClass

The ReferenceClass entity is a convenient shorthand for the REFERENCECLASS attribute. If this entity is present, the value shall 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 element in which it is used. The expected behavior is that the REFERENCECLASS attribute shall exist for classes and should not exist for instances.

5.2.10 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)">
```

DEPRECATED: The values "object" and "instance" have been deprecated in version 2.4.0 of this specification because they are not used.

5.2.11 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 shall 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 shall be attached to the corresponding PROPERTY in any instance, PARAMVALUE, or RETURNVALUE with the value "instance".

5.3 Element Descriptions

This subclause describes each of the elements in the CIM-XML schema.

5.3.1 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-XML 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 shall 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 shall not be represented in the

attribute (for example, 2.3.0, 2.4.0). Implementations need to 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 DSP0201 (this document) to which the XML document conforms. It shall 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 shall not be represented in the attribute (for example, 2.2.0, 2.3.0). Implementations need to 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.

5.3.2 Declaration Elements

This subclause defines those XML elements that are concerned with expressing the declaration of CIM objects.

5.3.2.1 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)+>
```

5.3.2.2 DECLGROUP

The DECLGROUP element defines a 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.

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

5.3.2.3 DECLGROUP.WITHNAME

The DECLGROUP.WITHNAME element defines a 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.

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*)>
```

5.3.2.4 DECLGROUP.WITHPATH

The DECLGROUP.WITHPATH element defines a set of CIM class and instance declarations. Each object is declared with its own independent naming and location information.

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

5.3.2.5 QUALIFIER.DECLARATION

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

A VALUE or a VALUE.ARRAY subelement shall 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 shall 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 NAME attribute (defined by the CIMName entity) defines the name of the qualifier, and the TYPE attribute (defined by the CIMType entity) and ISARRAY attributes together define the CIM type.

The ISARRAY attribute shall 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 shall be consistent with the declared qualifier default value.

The ARRAYSIZE attribute (defined by the ArraySize entity) shall not be present. Its use on this element has been deprecated in version 2.4 of this document. Note that DSP0004 defines that qualifier types that are arrays need to be variable-length arrays.

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

5.3.2.6 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 shall declare at least one attribute with a `true` value. (Otherwise, the qualifier would have no applicable scope.)

5.3.3 Value Elements

This subclause defines those XML elements that are concerned with expressing CIM-typed values.

5.3.3.1 VALUE

The VALUE element is used to define a single (non-array), non-reference, non-NULL CIM value.

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

Because the same element is used for values of all CIM types, the CIM type determines the format of the content of the VALUE element, as defined in the following subclauses of this clause. These subclauses also define when the VALUE element is character-preserving (see 5.1.1.2). In most cases, the CIM type is provided using the `TYPE` or `PARAMTYPE` attributes of the direct or an indirect parent element of the VALUE element.

5.3.3.1.1 String Values

If the CIM type is `string`, the content of the VALUE element shall be a sequence of zero or more UCS characters that represent the CIM value. An empty content of the VALUE element value represents an empty string (that is, "" in MOF). The character repertoire defined for the CIM string type shall be supported as defined in DSP0004 and 5.1.1.5. The content of the VALUE element shall not have additional surrounding string delimiter characters (such as double-quote or single-quote characters) compared to the CIM value. The actual representation of UCS characters depends on the `encoding` attribute defined in the XML declaration (`<?xml ... ?>`).

This use of the VALUE element is character-preserving (see 5.1.1.2).

5.3.3.1.2 Character Values

If the CIM type is `char16`, the content of the VALUE element shall be a single UCS character that represents the CIM value. The character repertoire defined for the CIM char16 type shall be supported as defined in DSP0004 and 5.1.1.5. The content of the VALUE element shall not have additional surrounding string delimiter characters (such as double-quote or single-quote characters) compared to the CIM value. The actual representation of the UCS character depends on the `encoding` attribute defined in the XML declaration (`<?xml ... ?>`).

This use of the VALUE element is character-preserving (see 5.1.1.2).

5.3.3.1.3 Real Values

If the CIM type is `real32` or `real64`, the content of the VALUE element shall conform to the format defined by the following ABNF rules and shall represent the CIM value, 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 ] / specialState

specialState = "INF" / "-INF" / "NaN"
```

The basis for the exponent shall be 10. The significand shall 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. 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.

The special states for floating point numbers defined by IEEE 754, +Infinity, -Infinity, and the NaN states, shall be represented by the literals "INF", "-INF", and "NaN", respectively (consistent with the XML datatypes xs:float and xs:double defined in XML Schema, Part 2). The NaN states shall all be represented by the same string, "NaN". These literals shall be produced with the lexical case as stated; they shall be consumed using case insensitive parsing, for backward compatibility with existing implementations.

This use of the VALUE element is whitespace-tolerant (see 5.1.1.3).

5.3.3.1.4 Boolean Values

If the CIM type is boolean, the content of the VALUE element shall be either TRUE or FALSE and shall represent the CIM value. These values shall be treated as case-insensitive by CIM-XML consumers. CIM-XML producers should use upper case.

This use of the VALUE element is whitespace-tolerant (see 5.1.1.3).

5.3.3.1.5 Integer Values

If the CIM type belongs to the set {uint8, uint16, uint32, uint64}, the content of the VALUE element shall be a valid unsigned decimal or hexadecimal value that represents the CIM value.

If the CIM type belongs to the set {sint8, sint16, sint32, sint64}, the content of the VALUE element shall be a valid signed decimal or hexadecimal value that represents the CIM value.

Decimal values have the format defined by the following ABNF rule, 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 shall not be used when the CIM type is unsigned.

Hexadecimal values have the format defined by the following ABNF rule, 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 shall not be used when the CIM type is unsigned.

This use of the VALUE element is whitespace-tolerant (see 5.1.1.3).

5.3.3.1.6 Datetime Values

If the CIM type is datetime, the content of the VALUE element shall be a valid datetime value that represents the CIM 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 shall not be surrounded by string delimiter characters (such as double-quote or single-quote characters).

This use of the VALUE element is character-preserving (see 5.1.1.2).

NOTE: This use of the VALUE element needs to be character-preserving in order to properly handle the case where a key property has datetime type and its TYPE attribute is not provided (see KEYVALUE).

5.3.3.2 VALUE.ARRAY

The VALUE.ARRAY element is used to represent a CIM value of 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 shall appear in the order of the array entries.

If the value of an array entry is NULL, the VALUE.NULL subelement shall be used to represent the array entry. Otherwise, the VALUE subelement shall 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)*>
```

5.3.3.3 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)>
```

5.3.3.4 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 shall appear in the order of the array entries.

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

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

5.3.3.5 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)>
```

5.3.3.6 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)>
```

5.3.3.7 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))>
```

5.3.3.8 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))>
```

5.3.3.9 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))>
```

5.3.3.10 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>
```

5.3.3.11 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)>
```

5.3.4 Naming and Location Elements

This clause defines those XML elements that are concerned with expressing the name and location of CIM objects (namespaces, classes, instances and qualifiers).

5.3.4.1 NAMESPACEPATH

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

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

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

5.3.4.2 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+)>
```

5.3.4.3 HOST

The HOST element is used to define a single host, optionally including a port number.

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

The format of the content of the HOST element shall conform to the following ABNF rule:

```
hostport = host [ ":" port ]
```

Where `host` and `port` are ABNF rules defined in RFC3986.

If port is not specified, the CIM-XML consumer shall assume the port numbers registered with IANA for the CIM-XML protocol as defaults:

- port 5988, for use with CIM-XML over HTTP
- port 5989, for use with CIM-XML over HTTPS

This specification defines the following additional rules for using the ABNF rules from RFC3986 for the content of the HOST element:

- `host` (and `port`) shall not use URI percent-encoding
- `reg-name` (used to specify hostnames) is restricted to conform to the syntax for DNS domain names as defined in section 3.1 of RFC1034 (that is, segments are separated by a dot, each segment is limited to 63 characters, and the total length is limited to 255 characters)

Note that specifying zone identifiers (also known as zone indices) for IPv6 addresses does not make sense in IPv6 addresses that are transmitted in a protocol to another host, because their meaning is strictly local to the originating host. For this reason, the syntax defined by the `IPv6address` ABNF rule from RFC3986 (which does not permit the use of zone identifiers) is sufficient.

Implementations shall support the specification of port, and the syntax defined by `IPv4address` and `reg-name`. Implementations should in addition support the syntax defined by `IPv6address`. Implementations do not need to support the syntax defined by the `IPFuture` ABNF rule from RFC3986 at this point.

5.3.4.4 NAMESPACE

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

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

The NAME attribute (defined by the CIMName entity) defines the name of the namespace.

5.3.4.5 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)>
```

5.3.4.6 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)>
```

5.3.4.7 CLASSNAME

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

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

The NAME attribute (defined by the CIMName entity) defines the name of the class.

5.3.4.8 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)>
```

5.3.4.9 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)>
```

5.3.4.10 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 shall 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 (defined by the ClassName entity) defines the name of the class for this path.

5.3.4.11 OBJECTPATH

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

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

5.3.4.12 KEYBINDING

The KEYBINDING element defines a single key property value binding.

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

The NAME attribute (defined by the CIMName entity) indicates the name of the key property.

5.3.4.13 KEYVALUE

The KEYVALUE element defines the value of a (scalar) key property that has a non-reference type.

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

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, for example as defined in DSP0207). The value of this attribute shall 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 VALUETYPE attribute has been deprecated in version 1.4 of this document. Use the TYPE attribute instead.

The TYPE attribute (defined by the CIMType entity) identifies the CIM type of the key property. The TYPE attribute is required to be provided by CIM-XML producers as of version 1.4 of this document. For implementations of earlier versions of this document, it is strongly recommended that CIM-XML producers always provide the TYPE attribute, because it supports strongly typed representations of values in CIM-XML consumers and can be used to improve performance.

The content of the KEYVALUE element represents the property value. Note that key properties cannot be NULL. Because the KEYVALUE element is used for key property values of all CIM types, the CIM type of the key property determines the format of the content of the KEYVALUE element, as follows:

- CIM-XML producers shall set the content of the KEYVALUE element based on the CIM type of the key property as defined in 5.3.3.1.
- If the TYPE attribute is provided, CIM-XML consumers shall interpret the content of the KEYVALUE element based on the TYPE attribute as defined in 5.3.3.1.
- If the TYPE attribute is not provided (for example, when earlier versions of this specification are implemented), CIM-XML consumers that have no knowledge about the CIM type of the key property shall interpret the content of the KEYVALUE element based on the VALUETYPE attribute as follows:
  - If the value of the VALUETYPE attribute is `string`, the content of the KEYVALUE element shall be interpreted as defined in 5.3.3.1.1.
  - If the value of the VALUETYPE attribute is `boolean`, the content of the KEYVALUE element shall be interpreted as defined in 5.3.3.1.4.
  - If the value of the VALUETYPE attribute is `numeric`, the content of the KEYVALUE element shall be interpreted as defined in 5.3.3.1.3 or 5.3.3.1.5, depending on which syntax matches.

5.3.5 Object Definition Elements

This subclause defines those XML elements that are concerned with expressing the declaration of CIM objects (classes, instances, and qualifiers) and their components (properties, methods, and parameters).

5.3.5.1 CLASS

The CLASS element defines a single CIM class.

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

The NAME attribute (defined by the CIMName entity) defines the name of the class.

The SUPERCLASS attribute (defined by the SuperClass entity), 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.

5.3.5.2 INSTANCE

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

The instance shall 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 shall 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 (defined by the ClassName entity) defines the name of the CIM class of which this is an instance.

5.3.5.3 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 shall 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 NAME attribute (defined by the CIMName entity) defines the name of the qualifier, and the TYPE attribute (defined by the CIMType entity) defines the CIM type.

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

The NAME attribute (defined by the CIMName entity) defines the name of the property, and the TYPE attribute (defined by the CIMType entity) defines the CIM type.

If the class definition for the property has the EMBEDDEDOBJECT or EMBEDDEDINSTANCE qualifier attached, the EmbeddedObject attribute (defined by the EmbeddedObject entity) shall be provided on PROPERTY elements representing properties in instances of that class, as follows. The EmbeddedObject attribute shall not be provided on PROPERTY elements representing properties in class definitions.

- A property that is defined in the class as an embedded object by attaching the EMBEDDEDOBJECT qualifier on the property shall be represented using the EmbeddedObject attribute with a value of `"object"`. The (string-typed) property value shall be a valid INSTANCE element, defining a single CIM instance of a CIM class or a valid CLASS element; where these elements shall be escaped as defined in 5.1.1.4.5.
- A property that is defined in the class as an embedded instance by attaching the EMBEDDEDINSTANCE qualifier on the property shall be represented using the EmbeddedObject attribute with a value of `"instance"`. The (string-typed) property value shall be a valid INSTANCE element, defining a single CIM instance of the CIM class specified in the EMBEDDEDINSTANCE qualifier; where this element shall be escaped as defined in 5.1.1.4.5.

As a result, if an embedded instance has properties that are again embedded objects, each such level of embedding will be escaped separately, and thus, recursively.

5.3.5.5 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 shall 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 shall be interpreted as a value of NULL.

The NAME attribute (defined by the CIMName entity) defines the name of the property, and the TYPE attribute (defined by the CIMType entity) defines the CIM type.

On a PROPERTY.ARRAY element within a containing CLASS element, the ARRAYSIZE attribute (defined by the ArraySize entity) shall be present if the array is a fixed-length array, and shall be absent if the array is a variable-length array.

On a PROPERTY.ARRAY element within a containing INSTANCE element, the ARRAYSIZE attribute should be absent, and the presence or absence of the ARRAYSIZE attribute shall not be interpreted as meaning that the property type is a fixed-length or variable-length array (that is, the CLASS definition is always authoritative in this respect).

If the class definition for the property has the EMBEDDEDOBJECT or EMBEDDEDINSTANCE qualifier attached, the EmbeddedObject attribute (defined by the EmbeddedObject entity) shall be provided on PROPERTY.ARRAY elements representing properties in instances of that class, as follows. The EmbeddedObject attribute shall not be provided on PROPERTY.ARRAY elements representing properties in class definitions.

- A property that is defined in the class as an embedded object by attaching the EMBEDDEDOBJECT qualifier on the property shall be represented using the EmbeddedObject attribute with a value of `"object"`. The (string-typed) property value shall be a valid INSTANCE element, defining a single CIM instance of a CIM class or a valid CLASS element; where these elements shall be escaped as defined in 5.1.1.4.5.
- A property that is defined in the class as an embedded instance by attaching the EMBEDDEDINSTANCE qualifier on the property shall be represented using the EmbeddedObject attribute with a value of `"instance"`. The (string-typed) property value shall be a valid INSTANCE element, defining a single CIM instance of the CIM class specified in the EMBEDDEDINSTANCE qualifier; where this element shall be escaped as defined in 5.1.1.4.5.

As a result, if an embedded instance has properties that are again embedded objects, each such level of embedding will be escaped separately, and thus, recursively.

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

The NAME attribute (defined by the CIMName entity) defines the name of the property.

The REFERENCECLASS attribute (defined by the ReferenceClass entity), 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 shall 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.

5.3.5.7 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 NAME attribute (defined by the CIMName entity) defines the name of the method.

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

5.3.5.8 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 NAME attribute (defined by the CIMName entity) defines the name of the parameter. The TYPE attribute (defined by the CIMType entity) defines the CIM type of the parameter.

5.3.5.9 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 NAME attribute (defined by the CIMName entity) defines the name of the parameter.

The REFERENCECLASS attribute (defined by the ReferenceClass entity), 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 shall exist for PARAMETER.REFERENCE entities.

5.3.5.10 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 NAME attribute (defined by the CIMName entity) defines the name of the parameter. The TYPE attribute (defined by the CIMType entity) defines the CIM type of the parameter.

The ARRAYSIZE attribute (defined by the ArraySize entity) shall be present if the array is a fixed-length array, and shall be absent if the array is a variable-length array.

5.3.5.11 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 NAME attribute (defined by the CIMName entity) defines the name of the parameter.

The REFERENCECLASS attribute (defined by the ReferenceClass entity) 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 shall exist for PARAMETER.REFARRAY entities.

The ARRAYSIZE attribute (defined by the ArraySize entity) shall be present if the array is a fixed-length array, and shall be absent if the array is a variable-length array.

5.3.6 Message Elements

This subclause defines those XML elements that are concerned with expressing CIM-XML messages for DSP0200.

5.3.6.1 MESSAGE

The MESSAGE element models a single CIM-XML 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 shall 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 shall 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.

5.3.6.2 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+)>
```

5.3.6.3 SIMPLEREQ

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

In addition, it contains zero or more CORRELATOR elements, each representing a client-defined operation correlator. For details on operation correlators, see DSP0200.

```
<!ELEMENT SIMPLEREQ (CORRELATOR*, (METHODCALL | IMETHODCALL))>
```

5.3.6.4 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 NAME attribute (defined by the CIMName entity) defines the name of the method to be invoked.

5.3.6.5 PARAMVALUE

The PARAMVALUE element defines the value of an input or output parameter of an extrinsic method call, or the value of an output parameter of an intrinsic method call. Note that input parameters of intrinsic method calls are represented by the IPARAMVALUE element; this inconsistency for intrinsic methods has historical reasons.

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

The child element of the PARAMVALUE element represents the parameter value. The absence of a child element indicates that the parameter value is NULL.

The NAME attribute (defined by the CIMName entity) defines the name of the parameter.

When PARAMVALUE is used in METHODCALL or METHODRESPONSE (that is, for extrinsic methods), the following applies:

- The PARAMTYPE attribute (defined by the ParamType entity) if provided identifies the CIM type of the parameter.
- If the direct child element of the PARAMVALUE element is a VALUE or VALUE.ARRAY element, the CIM type of the parameter determines the format of the content of the VALUE child element and CIM-XML producers shall provide the PARAMTYPE attribute with one of the values defined in Table 1. The requirement to provide the PARAMTYPE attribute was added in version 2.4.0 of this specification because it supports strongly typed representations of values in CIM-XML consumers and can be used to improve performance. CIM-XML producers that support older versions of this specification may not provide the PARAMTYPE attribute for these direct child elements. In that case, CIM-XML consumers need to have knowledge about the CIM type in order to interpret the value of (direct or indirect) VALUE child elements correctly (for example, the uint8 value 3 and the string value "3" are both represented as `<VALUE>3</VALUE>`). If CIM-XML consumers do not have knowledge about the CIM type, they should assume as a default that the value is string-typed.
- If the direct child element of the PARAMVALUE element is a VALUE.REFERENCE or VALUE.REFARRAY element, CIM-XML producers should provide the PARAMTYPE attribute. If provided, it shall have the value defined in Table 1. Because there is only one possible PARAMTYPE value for each of these child elements, CIM-XML consumers can infer the CIM type of the parameter from the name of the direct child element, if the PARAMTYPE attribute is not provided.
- Other direct child elements of the PARAMVALUE element are not permitted if used in METHODCALL or METHODRESPONSE.

If the class definition for the extrinsic method parameter has the EMBEDDEDOBJECT or EMBEDDEDINSTANCE qualifier attached, the EmbeddedObject attribute (defined by the EmbeddedObject entity) shall be provided, as follows.

- A method parameter that is defined in the class as an embedded object by attaching the EMBEDDEDOBJECT qualifier on the parameter shall be represented using the EmbeddedObject attribute with a value of `"object"`. The (string-typed) parameter value shall be a valid INSTANCE element, defining a single CIM instance of a CIM class or a valid CLASS element; where these elements shall be escaped as defined in 5.1.1.4.5.
- A method parameter that is defined in the class as an embedded instance by attaching the EMBEDDEDINSTANCE qualifier on the parameter shall be represented using the EmbeddedObject attribute with a value of `"instance"`. The (string-typed) parameter value shall be a valid INSTANCE element, defining a single CIM instance of the CIM class specified in the EMBEDDEDINSTANCE qualifier; where this element shall be escaped as defined in 5.1.1.4.5.

As a result, if an embedded instance has properties that are again embedded objects, each such level of embedding will be escaped separately, and thus, recursively.

Table 1 - Requirements for PARAMVALUE when used in METHODCALL or METHODRESPONSE

Direct child element of PARAMVALUE	Requirement to provide PARAMTYPE	Allowed PARAMTYPE values
VALUE	shall provide	boolean, string (including for embedded objects and octet strings), char16, uint8, sint8, uint16, sint16, uint32, sint32, uint64, sint64, datetime, real32, real64

Direct child element of PARAMVALUE	Requirement to provide PARAMTYPE	Allowed PARAMTYPE values
VALUE.ARRAY	shall provide	boolean, string (including for embedded objects and octet strings), char16, uint8 (including for octet strings), sint8, uint16, sint16, uint32, sint32, uint64, sint64, datetime, real32, real64
VALUE.REFERENCE	should provide	reference
VALUE.REFARRAY	should provide	reference

When PARAMVALUE is used in IMETHODRESPONSE (that is, for output parameters of intrinsic methods), the following applies:

- The PARAMTYPE attribute shall not be provided by CIM-XML producers, because the child element to be used and the datatype of the parameter value is known from the definition of the intrinsic method. Note that IPARAMVALUE (used for input parameters of intrinsic methods) does not provide for the specification of PARAMTYPE.
- The CIM types of intrinsic method output parameters and the child elements of PARAMVALUE that are to be used for representing parameter values are defined in DSP0200.

5.3.6.6 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 NAME attribute (defined by the CIMName entity) defines the name of the method to be invoked.

5.3.6.7 IPARAMVALUE

The IPARAMVALUE element defines the value of a parameter of an intrinsic method call.

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

The child element of the IPARAMVALUE element represents the parameter value. The absence of a child element indicates that the parameter value is NULL.

The NAME attribute (defined by the CIMName entity) defines the name of the parameter.

The IPARAMVALUE element does not provide information about the CIM type of the parameter. CIM-XML consumers are expected to have knowledge about the CIM types of intrinsic method parameters. The CIM types of intrinsic method parameters and the child elements of IPARAMVALUE that are to be used for representing parameter values are defined in DSP0200.

5.3.6.8 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+)>
```

5.3.6.9 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)>
```

5.3.6.10 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 NAME attribute (defined by the CIMName entity) defines the name of the method that was invoked.

5.3.6.11 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;>
```

The NAME attribute (defined by the CIMName entity) defines the name of the method that was invoked.

5.3.6.12 ERROR

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

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

The CODE attribute contains a numerical status code that indicates the nature of the error. The valid status codes are defined in DSP0200. The value of the CODE attribute is whitespace-tolerant (see 5.1.1.3).

The DESCRIPTION attribute, if present, provides a human-readable description of the error. The format of the value of the DESCRIPTION attribute, if provided, shall be a sequence of zero or more UCS characters and is character-preserving (see 5.1.1.2). The actual representation of UCS characters depends on the `encoding` attribute defined in the XML declaration (`<?xml ... ?>`).

5.3.6.13 RETURNVALUE

The RETURNVALUE element specifies the (scalar) value returned from an extrinsic method call.

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

The child element of the RETURNVALUE element represents the returned value. The absence of a child element indicates that the returned value is NULL.

The PARAMTYPE attribute (defined by the ParamType entity) if provided identifies the CIM type of the returned value.

If the direct child element of the RETURNVALUE element is a VALUE element, CIM-XML producers shall provide the PARAMTYPE attribute with one of the values defined in Table 2. The requirement to provide the PARAMTYPE attribute was added in version 2.4.0 of this specification because it supports strongly typed representations of values in CIM-XML consumers and can be used to improve performance. CIM-XML producers that support older versions of this specification may not provide the PARAMTYPE attribute for these direct child elements. In that case, CIM-XML consumers need to have knowledge about the CIM type in order to interpret the value of VALUE child elements correctly (for example, the uint8 value 3 and the string value "3" are both represented as `<VALUE>3</VALUE>`). If CIM-XML consumers do not have knowledge about the CIM type, they should assume as a default that the value is string-typed.

If the direct child element of the RETURNVALUE element is a VALUE.REFERENCE element, CIM-XML producers should provide the PARAMTYPE attribute. If provided, it shall have the value defined in Table 2. Because there is only one possible PARAMTYPE value for this child element, CIM-XML consumers can infer the CIM type of the return value from the name of the direct child element, if the PARAMTYPE attribute is not provided.

If the class definition for the extrinsic method has the EMBEDDEDOBJECT or EMBEDDEDINSTANCE qualifier attached, the EmbeddedObject attribute (defined by the EmbeddedObject entity) shall be provided, as follows.

- A return value that is defined in the class as an embedded object by attaching the EMBEDDEDOBJECT qualifier on the method shall be represented using the EmbeddedObject attribute with a value of `"object"`. The (string-typed) return value shall be a valid INSTANCE element, defining a single CIM instance of a CIM class or a valid CLASS element; where these elements shall be escaped as defined in 5.1.1.4.5.
- A return value that is defined in the class as an embedded instance by attaching the EMBEDDEDINSTANCE qualifier on the method shall be represented using the EmbeddedObject attribute with a value of `"instance"`. The (string-typed) return value shall be a valid INSTANCE element, defining a single CIM instance of the CIM class specified in the EMBEDDEDINSTANCE qualifier; where this element shall be escaped as defined in 5.1.1.4.5.

As a result, if an embedded instance has properties that are again embedded objects, each such level of embedding will be escaped separately, and thus, recursively.

Table 2 - Requirements for RETURNVALUE

Direct child element of RETURNVALUE	Requirement to provide PARAMTYPE	Allowed PARAMTYPE values
VALUE	shall provide	boolean, string (including for embedded objects and octet strings), char16, uint8, sint8, uint16, sint16, uint32, sint32, uint64, sint64, datetime, real32, real64

Direct child element of RETURNVALUE	Requirement to provide PARAMTYPE	Allowed PARAMTYPE values
VALUE.REFERENCE	should provide	reference

5.3.6.14 IRETURNVALUE

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

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

The child elements of the IRETURNVALUE element represent the returned value. The absence of a child element indicates that the returned value is NULL.

The IRETURNVALUE element does not provide information about the CIM type of the returned value. CIM-XML consumers are expected to have knowledge about the CIM types of intrinsic method return values. The CIM types of intrinsic method return values and the child elements of IRETURNVALUE that are to be used for representing return values are defined in DSP0200.

5.3.6.15 MULTIEXPREQ

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

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

5.3.6.16 SIMPLEEXPREQ

The SIMPLEEXPREQ element defines a simple CIM export request. It contains an EXPMETHODCALL (export method) subelement. In addition, it contains zero or more CORRELATOR elements, each representing a server-defined operation correlator. For details on operation correlators, see DSP0200.

```
<!ELEMENT SIMPLEEXPREQ (CORRELATOR*, EXPMETHODCALL)>
```

5.3.6.17 EXPMETHODCALL

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

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

The NAME attribute (defined by the CIMName entity) defines the name of the export method to be invoked.

5.3.6.18 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+)>
```

5.3.6.19 SIMPLEEXPRSP

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

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

5.3.6.20 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 NAME attribute (defined by the CIMName entity) defines the name of the export method that was invoked.

5.3.6.21 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 NAME attribute (defined by the CIMName entity) defines the name of the parameter.

5.3.6.22 ENUMERATIONCONTEXT (removed)

In version 2.3, this specification defined an ENUMERATIONCONTEXT element for representing the enumeration context value for pulled enumeration operations. However, that element was not used anywhere and has therefore been removed in version 2.4.0 of this specification. Enumeration context values are now represented like strings, as defined in DSP0200.

5.3.6.23 CORRELATOR

The CORRELATOR element defines a single operation correlator. For a description of the concept of operation correlators, see DSP0200.

```
<!ELEMENT CORRELATOR (VALUE)>
<!ATTLIST CORRELATOR
    %CIMName;
    %CIMType;          #REQUIRED>
```

The NAME attribute (defined using the CIMName entity) defines the name of the correlator. The correlator name shall conform to the format defined by the following ABNF rule:

```
correlator-name = org-id ":" local-id
```

`org-id` shall identify the business entity owning the definition of the semantics of the correlator. `org-id` shall include a copyrighted, trademarked, or otherwise unique name that is owned by that business entity or that is a registered ID assigned to that business entity by a recognized global authority. In addition, to ensure uniqueness, `org-id` shall not contain a colon (`:`).

`local-id` shall uniquely identify the correlator within `org-id`.

The TYPE attribute (defined using the CIMType entity) defines the CIM datatype of the correlator value.

The VALUE child element defines the value of the correlator.

ANNEX A
(informative)

Change History

Version	Date	Description
2.0.0	1999-06-02	Released as DMTF Final Standard
2.2.0	2007-01-11	Released as DMTF Final Standard
2.3.0	2008-11-11	Released as DMTF Standard
2.3.1	2009-07-29	Released as DMTF Standard
2.4.0	2014-01-16	Released as DMTF Standard, with the following changes: Changes: • Removed ENUMERATIONCONTEXT element because representation of enumeration context value was changed to string in DSP0200 (see 5.3.6.22) • Added requirement to provide the PARAMTYPE attribute of PARAMVALUE element for certain CIM types (see 5.3.6.5) • Added requirement to provide the TYPE attribute of the KEYVALUE element (see 5.3.4.13) • Updated several normative references (see clause 2) Deprecations:: • Deprecated the use of the values "object" and "instance" for the PARAMTYPE attribute as they were not used (see 5.2.10) • Deprecated the VALUETYPE attribute of the KEYVALUE element; use TYPE instead (see 5.3.4.13) Additional functions and requirements: • Added support for operation correlators (see 5.3.6.23) • Added support for, respectively clarified, the representation of special values (NaN, Infinites) for real numbers (see 5.3.3.1.3) Clarifications: • Clarified that PARAMVALUE (and not IPARAMVALUE) is used for output parameters of intrinsic methods in IMETHODRESPONSE (see 5.3.6.5) • Clarified the allowed child elements of PARAMVALUE (see 5.3.6.5) • Removed ordering requirements for DECLGROUP* elements (see 5.3.2.2ff) • Clarified escaping, white space handling, and character repertoire (see 5.1.1) • Clarified XML encoding of embedded instances and objects (see 5.1.1.4.5) • Clarified that CLASSORIGIN indicates the leaf-most class (see 5.2.4) • Clarified precision requirements for real numbers (see 5.3.3.1.3) • Clarified syntax and requirements for HOST element (see 5.3.4.3) • Clarified use of the ARRAYSIZE attribute in any elements, and deprecated its use on the QUALIFIER.DECLARATION element Editorial changes: • Cleaned up terminology • Fixed incorrect normative and bibliographic references • Fixed syntax errors in DTD • Added the missing INSTANCEPATH and VALUE.INSTANCEWITHPATH child element to IRETURNVALUE, in support of the PullInstancePaths, PullInstancesWithPath and Open<XXX>Instances operations.

Bibliography

DMTF DSP0203, *DTD for Representation of CIM in XML 2.4*,
http://www.dmtf.org/standards/published_documents/DSP0203_2.4.dtd

DMTF DSP0207, *WBEM URI Mapping Specification 1.0*,
http://www.dmtf.org/standards/published_documents/DSP0207_1.0.pdf

DMTF DSP8044 *XSD for Representation of CIM in XML 2.4*,
http://schemas.dmtf.org/wbem/cim-xml/2/dsp8044_2.4.xsd

W3C XML Schema, Part 0: Primer (Second Edition), W3C Recommendation, 28 October 2004,
http://www.w3.org/TR/2004/REC-xmlschema-0-20041028

W3C XML Schema, Part 2: Datatypes (Second Edition), W3C Recommendation, 28 October 2004,
http://www.w3.org/TR/2004/REC-xmlschema-2-20041028