SMPTE STANDARD

D-Cinema — XML Data Types

Table of Contents

Approved
March 3, 2008

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in Part XIII of its Administrative Practices.

SMPTE Standard 433 was prepared by Technology Committee DC28.

Introduction

In the development of XML Schema for a number of D-Cinema artifacts, it was observed that a number of common data types were defined and redefined multiple times in application schemas. This document combines the definition of the most commonly duplicated data types in a single namespace, along with additional types that are most commonly defined in large XML projects, and which have general applicability to the field of Digital Cinema Systems. The intention is that this namespace can be included in other schema to easily use these types as needed. These data types are defined using W3C XML Schema Definition Language (XSDL) as defined in [XML-SCH-1].

1 Scope

The purpose of this document is to define and describe common data types used in D-Cinema metadata structures for applications using XML as a data description language. In addition, this document includes normative references for types and identifiers, defined elsewhere, which are necessary to use XML in standards documents. The structure and composition of the types defined in this document are simple yet general in order to promote reusability in other standards.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate 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; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

3 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

[XML-SCH-1] XML Schema Part 1: Structures http://www.w3.org/TR/xmlschema-1-20041028/

[XML-SCH-2] XML Schema Part 2: Datatypes http://www.w3.org/TR/xmlschema-2-20041028/

[ISO-639-2] International Standards Organization (ISO) ISO-639-2 Codes for the representation of names of languages-- Part 2: alpha-3 code.
Normative Text: http://www.loc.gov/standards/iso639-2/normtext.html
Registration Authority: US Library of Congress: http://www.loc.gov/standards/iso639-2/

[ISO-3166-1] International Standards Organization (ISO) ISO 3166-1:1997 Codes for the representation of names of countries and their subdivisions - Part 1: Country codes
Official List: http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html
Decoding Table: http://www.iso.ch/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/iso_3166-1_decoding_table.html

[RFC-4122] Internet Engineering Task Force (IETF) RFC 4122 "A Universally Unique Identifier (UUID) URN Namespace" http://www.ietf.org/rfc/rfc4122.txt

[RFC-3066] Internet Engineering Task Force (IETF) RFC 3066 "Tags for Identification of Languages" (January 2001) http://www.ietf.org/rfc/rfc3066.txt

[RFC-3629] Internet Engineering Task Force (IETF) RFC 3629: "UTF-8, a transformation format of ISO 10646" http://www.ietf.org/rfc/rfc3629.txt

[D-Cert] SMPTE 430-2-2006, D-Cinema Operations – Digital Certificate

4 DCML Namespace

This document uses many of the basic datatypes specified in [XML-SCH-2] to define a number of derived types for D-Cinema applications. This collection of types shall be referred to as Digital Cinema Mark-up Language (DCML).

The types defined in this document shall be included in an XML namespace, whose Namespace Name (URI) shall be:

```
http://www.smpte-ra.org/schemas/433/2008/dcmlTypes/
```

When published, the version number of the informative XML Schema representation of the namespace shall be set to "1.0", where "1" is the major part of the version number, and "0" is the minor part. If the namespace is subsequently revised to add new types not defined in this standard, but not to modify any of the existing types in a way that breaks reverse compatibility, the minor part of the version number shall be incremented by one. If the namespace is subsequently modified with result of changing any of the types defined in this document in a way that breaks reverse compatibility, the major part of the version number shall be incremented by one. The version number shall be specified by the version attribute the xs:schema element of the XML schema that this namespace represents.

5 Basic Data Types

This section describes a set of fundamental data types that are intended for use in the construction of more complex types in application schemas.

5.1 UUID Type

UUIDs are used as object identifiers in many places in schema for D-Cinema. The structure of a UUID is normatively defined in [RFC-4122]. In schemas, UUIDs shall be represented by the following data type:

```
<xs:simpleType name="UUIDType">
        <xs:restriction base="xs:anyURI">
                <xs:pattern value="urn:uuid:[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-
F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}"/>
        </xs:restriction>
</xs:simpleType>
```

5.2 User Text Type

In various places in schemas, user text data is specified. In order to provide for language specification and a default language, the following User Text type is extended from a simple string type. Schemas shall use this type to represent stored text intended for viewing by users. Note that the language identifier attribute is optional, but defaults to English in the absence of the attribute. The language identifiers are defined in [RFC-3066]. In XSDL, the User Text Type shall be defined as follows:

```
<xs:complexType name="UserTextType">
    <xs:simpleContent>
        <xs:extension base="xs:string">
               <xs:attribute name=”language” type=”xs:language” use="optional"
default="en"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

5.3 Rational Type

Many relationships in media are described in terms of ratios of integers. Image aspect ratios and edit rates are common examples of this. In order to provide a consistent representation of these ratios, the Rational data type is defined as an ordered list of two long integers. The first long integer in the list is the numerator, and the second is the denominator. In XSDL, the Rational Type shall be defined as follows:

```
<xs:simpleType name="RationalType">
        <xs:restriction>
                <xs:simpleType>
                        <xs:list itemType="xs:long"/>
                </xs:simpleType>
                <xs:length value="2"/>
                <xs:pattern value="[-+]?[0-9]+ 0*[1-9][0-9]*"/>
        </xs:restriction>
</xs:simpleType>
```

5.4 Units of Measure

The basic datatypes defined in [XML-SCH-2] do not address units of measure. In Digital Cinema systems, many specifications are expressed in units of measurement, and many quantities must be measured. While the units of measurement may be implicit in some cases, in other cases explicit units are required to clarify the meaning of the measurement. This section defines XML data types for common units of measure that are useful for digital cinema applications, such as system set-up and monitoring.

5.4.1 Units of Temperature

Temperatures are normally expressed on one of three scales. This type defines a set of tokens, which provide for the expression of temperature scale units. The set of temperature unit tokens shall be defined as follows:

```
<xs:simpleType name="temperatureUnitsToken">
   <xs:restriction base="xs:token">
      <xs:enumeration value="celsius"/>
      <xs:enumeration value="fahrenheit"/>
      <xs:enumeration value="kelvin"/>
   </xs:restriction>
</xs:simpleType>
```

5.4.2 Units of Voltage

Units of voltage are well understood, but are usually scaled in chunks of three orders of magnitude when expressed as measurements. This type defines tokens for the common scaling orders used in common electronic systems. The set of voltage unit tokens shall be defined as follows:

```
<xs:simpleType name="voltageUnitsToken">
   <xs:restriction base="xs:token">
      <xs:enumeration value="volts"/>
      <xs:enumeration value="millivolts"/>
      <xs:enumeration value="microvolts"/>
   </xs:restriction>
</xs:simpleType>
```

5.4.3 Units of Current

Units of current are expressed in amperes, and are usually scaled in chunks of three orders of magnitude when expressed as measurements. This type defines tokens for the common scaling orders used in common electronic systems. The set of current unit tokens shall be defined as follows:

```
<xs:simpleType name="currentUnitsToken">
   <xs:restriction base="xs:token">
      <xs:enumeration value="amps"/>
      <xs:enumeration value="milliamps"/>
   </xs:restriction>
</xs:simpleType>
```

Current is commonly understood to be measured in two modes: either alternating current (AC) or direct current (DC). The current mode shall be expressed with a token defined in the following type:

```
<xs:simpleType name="currentModeToken">
   <xs:restriction base="xs:token">
      <xs:enumeration value="AC"/>
      <xs:enumeration value="DC"/>
   </xs:restriction>
</xs:simpleType>
```

5.4.4 Units of Duration (Time)

Duration is defined in units of seconds, where sixty seconds is one minute, sixty minutes are one hour, twenty-four hours are one day, seven days are one week, and 365 days are one year. Standard engineering practice provides the three orders of magnitude decimal offsets for subdivisions of seconds. Durations of time shall be expressed using the following tokens to convey the unit period of time expressed.

```
<xs:simpleType name="timeUnitToken">
   <xs:restriction base="xs:token">
      <xs:enumeration value="week"/>
      <xs:enumeration value="day"/>
      <xs:enumeration value="hour"/>
      <xs:enumeration value="minute"/>
      <xs:enumeration value="second"/>
      <xs:enumeration value="millisecond"/>
      <xs:enumeration value="microsecond"/>
   </xs:restriction>
</xs:simpleType>
```

Note: Durations expressed in years do not account for leap year corrections – that work is left to the application.

5.5 Temperature Type

Quantities of temperature shall be expressed with the following data type:

```
<xs:complexType name="temperatureType">
   <xs:simpleContent>
      <xs:extension base="xs:decimal">
         <xs:attribute name="units" type="dcml:temperatureUnitsToken"
use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

5.6 Voltage Type

Quantities of voltage shall be expressed with the following data type:

```
<xs:complexType name="voltageType">
   <xs:simpleContent>
      <xs:extension base="xs:decimal">
         <xs:attribute name="units" type="dcml:voltageUnitsToken"
use="required"/>
         <xs:attribute name="mode" type="dcml:currentModeToken" use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

5.7 Current Type

Quantities of current shall be expressed with the following data type:

```
<xs:complexType name="currentType">
   <xs:simpleContent>
      <xs:extension base="xs:decimal">
         <xs:attribute name="units" type="dcml:currentUnitsToken"
use="required"/>
         <xs:attribute name="mode" type="dcml:currentModeToken" use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

5.8 Duration Type

Durations (quantities of time) shall be expressed with the following data type:

```
<xs:complexType name="durationType">
    <xs:simpleContent>
        <xs:extension base="xs:decimal">
            <xs:attribute name="units" type="dcml:timeUnitToken"
use="required"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

5.9 Rate Type

Many interesting measurements are expressed as **rates**, or the number of instances of an event that occur in a given period of time. Definition of a Rate Type requires specification of the type of event being counted and the period of time of counting. The previously defined rational type is extended to define a Rate Type, which shall be defined as follows:

```
<xs:complexType name="rateType">
   <xs:simpleContent>
      <xs:extension base="dcml:RationalType">
         <xs:attribute name="eventscope" type="xs:anyURI" use="optional"
default="http://www.smpte-ra.org/schemas/433/2008/dcmlTypes#rate-scope-tokens"/>
         <xs:attribute name="event" type="xs:token" use="optional"/>
         <xs:attribute name="period" type="dcml:timeUnitToken" use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

The optional eventscope attribute in the definition above shall have a default value of

```
   "http://www.smpte-ra.org/schemas/433/2008/dcmlTypes#rate-scope-tokens"
```

which refers to the list of possible values for the event token. This default value for the eventscope attribute URI shall refer to the values listed in the following table. If an online resource directory is implemented for this schema, a list of allowable tokens shall be provided at the resource page that the attribute value resolves to when treated as a URL.

Token	Description
Cycle	a cycle of repeating function
Revolution	a complete revolution of a rotating device
Frame	a frame of content, as defined for that content
Sample	a sample of essence, as defined for that essence
Byte	eight bits
Unit	a generic unit

5.9.1 Rate Type Examples (Informative)

```
<FanSpeed event="revolution" period="second">1200 1</FanSpeed>
<MainsRate event="cycle" period="second">50 1</MainsRate>
<Retries event="cycle" period="hour">3 1</Retries>
<samplerate event="sample" period="second">48000 1</samplerate>
<framerate event="frame" period="second">24000 1001</framerate>
```

5.10 Device Identifier Types

A fundamental requirement for building systems of multiple devices is to identify the devices in the system. This section defines the data types that shall be used to identify devices in a digital cinema system.

Depending on the role of a particular physical or logical device, a device may be identified either by a UUID, or by a security certificate thumbprint, or both. By definition, a device, which functions as a security device, must contain a security certificate. Because a single device may have multiple roles, a device may have both kinds of identifiers associated with it.

5.10.1 Basic Identifiers

5.10.1.1 Device UUID

If a device is to be identified by a UUID, that UUID shall be represented using the dcml:UUIDType.

5.10.1.2 Certificate Thumbprint

If the subject device is to be identified by a security certificate, the device identifier shall be its Public Key Thumbprint,as defined in [D-Cert] Section 5.4 “Certificate and Public Key Thumbprint”. The Public Key Thumbprint shall be stored in XML using the ds:DigestValueType.

5.10.2 Device Identifier List Type

This data type shall be used to identify a specific device within a log record, an approved device list, a site device inventory report, a trusted device list (TDL), or anyplace else that a device in a theater system must be specifically identified. The DeviceIdentifierList shall contain at least one of the following two elements, and may contain both. Either type (Device UUID or Certificate Thumbprint) may be used in either the Primary ID or the Secondary ID element. Either of the two elements may be used to refer to the identified device, as both identifiers are unique to the device. This type may also be used to map from one identifier to the other when necessary.

5.10.2.1 Primary ID

The PrimaryID element shall contain the primary identifier of the device. This may be either the Device UUID or the device Certificate Thumbprint.

5.10.2.2 Secondary ID (optional)

The SecondaryID element, if present, shall contain the secondary identifier of the device. If the device has two identifiers, the secondary identifier may be provided in this field. This may be either the Device UUID or the device Certificate Thumbprint.

5.10.3 Schema Representation

Because the DeviceIdentifierList items are expressed using polymorphous types, the types are defined in some detail in order to provide for automated type checking as well as human readability and interpretation.

5.10.3.1 Device Identifier Type

The following XSDL shall define a union of the two data types allowed for device identification. This permits the two types to be used interchangeably in a well-defined manner.

```
<xs:simpleType name="deviceIdentifierUnion">
   <xs:union memberTypes="dcml:UUIDType ds:DigestValueType"/>
</xs:simpleType>
```

The Device Identifier Type shall be a polymorphic type with an attribute to indicate which sub-type of identifier is used in the expression. The Device Identifier Type shall be defined as follows:

```
<xs:complexType name="deviceIdentifierPolyType">
    <xs:simpleContent>
        <xs:extension base="dcml:deviceIdentifierUnion">
            <xs:attribute name="idtype" use="required">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
```

```
                        <xs:enumeration value="DeviceUID"/>
                        <xs:enumeration value="CertThumbprint"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

5.10.3.2 Device Identifier List Type

In order to support device identification using both types of identifiers in a well-defined manner, this type defines a standard XML wrapper for Device Identifiers. Lists of Device Identifiers shall be represented as follows:

```
<xs:complexType name="deviceIdentifierListType">
   <xs:sequence>
      <xs:element name="PrimaryID" type="dcml:deviceIdentifierPolyType"/>
      <xs:element name="SecondaryID" type="dcml:deviceIdentifierPolyType"
minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

5.10.4 Device Identifier Instance Example (Informative)

Assuming that an application schema used this type by defining an element as follows:

```
<xs:element name="DeviceIdentifierList" type="dcml:deviceIdentifierListType"/>
```

The following example illustrates an instance of a Device Identifier with both a Primary and (optional) Secondary identifier specified.

```
<DeviceIdentifierList>
   <PrimaryID idtype="DeviceUID">urn:uuid:00000000-0000-0000-0000-
000000000000</PrimaryID>
   <SecondaryID idtype="CertThumbprint">857ABCD023abcd84</SecondaryID>
</DeviceIdentifierList>
```

Note: It is recommended that deviceIdentifierPolyType not be used by itself in Digital Cinema applications. The deviceIdentifierListType type should used wherever a device identifier is required in order to provide a common device identifier data type that is suitable for representing all types of devices.

5.11 Device Type Identifier Type

The Device Type Identifier Type shall be used to describe the functional role of a device in a digital cinema system.

In XSDL, the DeviceType type shall be defined by extending a token type with an optional attribute and a default value for that attribute as follows:

```
<xs:complexType name="deviceTypeType">
   <xs:simpleContent>
      <xs:extension base="xs:token">
```

```
         <xs:attribute name="scope" type="xs:anyURI" use="optional"
default="http://www.smpte-ra.org/schemas/433/2008/dcmlTypes/#device-type-
tokens"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

The optional **scope** attribute with the default string value of

```
 "http://www.smpte-ra.org/schemas/433/2008/dcmlTypes/#device-type-tokens"
```

shall refer to the permissible values of the token element. If the attribute is absent or set to the default string value, the permissible values shall be one of the tokens listed in the following table. If an online resource directory is implemented for this schema, a list of allowable tokens shall be provided at the resource page that the attribute value resolves to when treated as a URL.

Table of Device Tokens	
Token	**Description**
DEC	Media Decoder
FMI	Forensic Mark Inserter (Image/Picture)
FMA	Forensic Mark Inserter (Audio/Sound)
LD	Link Decryptor (Image/Picture)
LE	Link Encryptor (Image/Picture)
MDA	Media Decryptor (Audio/Sound)
MDI	Media Decryptor (Image/Picture)
MDS	Media Decryptor (Subtitle)
NET	Networking Device
PLY	Playback Device
PR	Projector
PWR	Power Control System (UPS, etc.)
SM	Security Manager
SMS	Screen Management System
SPB	Secure Processing Block (Security Enclosure)
TAS	Theater Automation System (Lights, Curtain, etc.)
TMS	Theater Management System

5.12 Named Parameters and Parameter Lists

On occasion, it is convenient to encode values in XML whose type and meaning is not defined by available standardized schemas. In order to provide a structured method of doing so while providing for as much instance validation as possible, this section defines a standard method for representing Name-Value pairs and lists thereof, where the data types and names are not formally defined in a specific schema. Use of this technique should only be considered when a standard schema approach is not practical, perhaps because the types of all possible values cannot be defined in advance of their application and use.

5.12.1 Scoped Token Type

The Scoped Token Type shall be used to represent the name of a name-value pair. The optional attribute may refer to a table of tokens defined in a separate document, and if present, shall restrict the possible values of the token to those defined in that table. If the optional attribute is not present, any legal XML token may be used as a value of this type. The scope attribute should point to a table of allowable token values wherever possible.

```
<xs:complexType name="scopedTokenType">
   <xs:simpleContent>
      <xs:extension base="xs:token">
         <xs:attribute name="scope" type="xs:anyURI" use="optional"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

5.12.2 Named Parameter Type

The Named Parameter Type wraps a Scoped Token Type with one wildcard element. This construct shall be a name-value pair, where the value is of any valid type, defined in any namespace. The name element shall be named ‘Name” and the value (wildcard) element shall be named “Value”.

```
<xs:complexType name="namedParmType">
   <xs:sequence>
      <xs:element name="Name" type="dcml:scopedTokenType"/>
      <xs:element name="Value" type="xs:anyType"/>
   </xs:sequence>
</xs:complexType>
```

5.12.3 Parameter List Type

The Parameter List Type shall be used to wrap a sequence of one or more name-value pairs into a list. This type is provided for convenience in specifying a parameter list whose extent is not known in advance.

```
<xs:complexType name="parameterListType">
   <xs:sequence>
      <xs:element name="Parameter" type="dcml:namedParmType" minOccurs="1"
maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
```

5.12.4 Instance Example (Informative)

Following is an example of a parameter list instance fragment. Assuming that a schema used the Parameter List Type to define an element named NVParameterList, an example list instance of length two with token names from two different predefined tables could be written as:

```
<NVParameterList>
    <Parameter>
        <Name scope="http://www.smpte-ra.org/schemas/x/y/#example-
tokens">Authorization</Name>
        <Value>Yes</Value>
    </Parameter>
    <Parameter>
```

```
        <Name scope="http://www.smpte-ra.org/schemas/a/b/#constant-
tokens">Pi</Name>
        <Value>3.1415945</Value>
    </Parameter>
</NVParameterList>
```

5.13 Version Information Types

In a complex electronic system, hardware, firmware, and software may all have version information associated with individual devices. The version information for a device shall be represented as a name version tuple consisting of the device name and a string expressing the version information, represented in XSDL as follows:

```
<xs:group name="strstrNameValueGroup">
   <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Value" type="xs:string"/>
   </xs:sequence>
</xs:group>
```

Many devices may consist of a collection of subsystems, each of which may contain versionable components. In many cases, it may be desirable to express the versioned configuration of a device as a collection of the version information of the containing system along with the version information for the contained subsystems. In this case, the system version information shall be expressed as a list of name/version tuples with at least one first entry, which shall be the major (overall) version of the device hardware, firmware or software. Additional tuples may optionally be provided if the manufacturer wishes to provide subsystem names with version information for each subsystem. This construction shall be expressed in XSDL as follows:

```
<xs:complexType name="versionInfoListType">
   <xs:sequence>
      <xs:group ref="dcml:strstrNameValueGroup" minOccurs="1"
maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
```

5.13.1 Example Instance fragment (Informative):

```
<VersionInfo>
   <Name>Mark XX Projector</Name> <Value>1.2.5.2</Value>
   <Name>Lamp Housing Controller</Name> <Value>2.3.1</Value>
   <Name>Automated Lens Turret</Name> <Value>0.3.2</Value>
   <Name>Color Processor</Name> <Value>3.2</Value>
</VersionInfo>
```

6 Composite Types

Basic data types can be assembled into composite types, which can serve as building blocks for yet more complex types. The use of types as building blocks simplifies the construction of more complex types by abstracting out a layer of detail. In addition, the use of well-chosen base types in more complex types provides for homogenous representation of information across multiple related constructs in a suite.

6.1 Device Description Type

This composite data type is intended to describe a device in some detail, given a DeviceIdentifier from a DeviceDescriptionList. The contained information is separated from the DeviceIdentifierList type to minimize redundancy in device lists, log records, or other entities in which device are identified. For example, a device, when generating a report containing a list or sequence of Log Messages on behalf of itself or other devices, could include one complete DeviceDescription record the first time a particular device is mentioned in the list, but omit it in subsequent references to that device. Alternatively, DeviceDescription detail records could be supplied on request to the reporting device or retrieved from a device database, depending on the circumstances. The DeviceDescription Type shall be defined as follows:

6.1.1 Field Descriptions

6.1.1.1 Device Identifier

An identifier, either a UUID or a Certificate Thumbprint, that identifies the device in a permanent way. This identifier is not expected to change over the device lifetime, much like a serial number but globally unique. The value of this field shall be identical to one of the two possible fields in the DeviceIdentifierList for the same device, and shall be represented using the deviceIdentifierPolyType.

6.1.1.2 Device Type ID

A Device Type Identifier, which shall identify the nature or functional role of the device which is described in this instance of a DeviceDescription. The DeviceTypeID shall be represented using the Device Type Identifier Type (deviceTypeType). Its value shall be a token.

6.1.1.3 Device Subsystem Type ID (optional)

A Device Type Identifier, which may be used to indicate that the device specified in this field is a subsystem of the device specified in the preceding field. The DeviceSubsystemTypeID, if present, shall be represented using the Device Type Identifier Type (deviceTypeType), and its value shall be a token.

6.1.1.4 Additional ID (optional)

The AdditionalID element, if present, shall contain additional identification information for the device. The AdditionalID element may be any Type, defined in any namespace.

6.1.1.5 Device Serial

The manufacturer's serial number of the device shall be identified in this element.

6.1.1.6 Manufacturer ID (optional)

The Manufacturer ID, if present, shall be a URN that identifies the manufacturer of the device.

6.1.1.7 Manufacturer Cert ID (optional)

The ManufacturerCertID shall be the thumbprint of the public key of the certificate that the manufacturer used to sign the Device certificate. If the subject device has a certificate, this field may contain the certificate thumbprint of the manufacturer's certificate signing key, computed as the SHA-1 hash of the Public Key from the device certificate, computed per [RFC 3280] section 4.2.1.2 option 1.

6.1.1.8 Device Cert ID (optional)

The DeviceCertID, if present, shall be the thumbprint of the public key of the Device certificate. If the subject device has a certificate, this field may contain the certificate thumbprint, which is the SHA-1 hash of the Public Key from the device certificate, computed per [RFC 3280] section 4.2.1.2 option 1.

6.1.1.9 Manufacturer Name (optional)

The ManufacturerName element, if present, shall contain the device manufacturer's name (e.g. Acme Widgets, Inc.).

6.1.1.10 Device Name (optional)

The DeviceName element, if present, shall contain the model name of the device (e.g. Mark XX Projector)

6.1.1.11 Model Number (optional)

The ModelNumber element, if present, shall contain the model number of the device.

6.1.1.12 Version Info

In XSDL, the VersionInfo element shall be an instance of the versionListInfoType. This type shall be implemented as a sequence of strstrNameValueGroup. This group contains name/value pairs of type **string**, which shall be expressed as a versionInfoListType.

6.1.1.13 Device Comment (optional)

The DeviceComment element may be included as a free text field, that provides a human readable description of the device.

6.1.2 XML Schema Representation

The DeviceDescription type shall be defined as the deviceDescriptionType in XSDL as follows:

```
<xs:complexType name="deviceDescriptionType">
  <xs:sequence>
    <xs:element name="DeviceIdentifier" type="dcml:deviceIdentifierPolyType"/>
    <xs:element name="DeviceTypeID" type="dcml:deviceTypeType"/>
    <xs:element name="AdditionalID" type="xs:anyType" minOccurs="0"/>
    <xs:element name="DeviceSubsystemTypeID" type="dcml:deviceTypeType"
minOccurs="0"/>
    <xs:element name="DeviceSerial" type="xs:string"/>
    <xs:element name="ManufacturerID" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="ManufacturerCertID" type="ds:DigestValueType"
minOccurs="0"/>
```

```
    <xs:element name="DeviceCertID" type="ds:DigestValueType" minOccurs="0"/>
    <xs:element name="ManufacturerName" type="xs:string" minOccurs="0"/>
    <xs:element name="DeviceName" type="xs:string" minOccurs="0"/>
    <xs:element name="ModelNumber" type="xs:string" minOccurs="0"/>
    <xs:element name="VersionInfo" type="dcml:versionInfoListType"/>
    <xs:element name="DeviceComment" type="dcml:UserTextType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

6.1.3 Example Instance (Informative)

Following is an example instance of a Device Description element of the deviceDescriptionType.

Note: This example has been removed from this document to facilitate maintenance and processing with tools suitable to the purpose. If you are reviewing this document, you should have been provided with an up to date version of the informative schema in electronic form. The name of the file should be "DeviceDescExample.xml".

7 XML Schema (Informative)

Note: The Informative Schema has been removed from this document to facilitate maintenance and processing with tools suitable to the purpose. If you are reviewing this document, you should have been provided with an up to date version of the informative schema in electronic form. The name of the file should be "dcmlTypes.xsd".

Annex A (Informative)
Camel Case Usage

XML syntax requires that type names and element names be expressed as single words with no embedded spaces. This leads to the use of "CamelCase", in which multiple words are joined without spaces and are capitalized within the compound word to create descriptive multi-word identifiers. There are two forms of CamelCase: "lowerCamelCase", and "UpperCamelCase". A related form is known as "Title Case", in which the words are separated by spaces and certain small words such as "and" and "the" may or may not be capitalized.

When writing or referring XML type names and element names, the following rules are used to construct descriptive names from sequences of individual words:

- In section titles or body text, when indirectly referring to the name of an item but not naming the item itself, Title Case should be used.
- In body text, when directly referring to an item, the appropriate CamelCase name should be used.
- When naming an item that is included in a document from a normative reference, the naming convention from the normative document should be used.
- When naming an item which is an XML type name, lowerCamelCase should be used.
- When naming an item which is an XML element name, UpperCamelCase should be used.

Annex B (Informative)
Bibliography

B.1 Language Identifiers

[ISO 15924] ISO/DIS 15924 - Codes for the representation of names of scripts (under development by ISO TC46/SC2)

B.2 Character Coding

Unicode Transformation Format (UTF) FAQ: http://www.unicode.org/faq/utf_bom.html