An Overview of the MPEG-7 Description Definition Language (DDL)

Jane Hunter

***Abstract*—This paper presents an overview of the MPEG-7 Description Definition Language (DDL). The DDL provides the syntactic rules for creating, combining, extending and refining MPEG-7 Descriptors (Ds) and Description Schemes (DSs). In the interests of interoperability, the W3C's XML Schema language, with the addition of certain MPEG-7-specific extensions, has been chosen as the DDL. This paper describes the background to this decision and using examples, provides an overview of the core XML schema features used within MPEG-7 and the extensions made in order to satisfy the MPEG-7 DDL requirements.**

***Index Terms*—DDL, MPEG-7, parser, schema, XML.**

I. Introduction

THE Description Definition Language (DDL) forms a core part of the MPEG-7 standard. It provides the solid descriptive foundation by which users can create their own Description Schemes (DSs) and Descriptors (Ds). The DDL defines the syntactic rules to express, combine, extend and refine DSs and Ds. According to the definition in the MPEG-7 Requirements Document [1] the DDL is

> "…a language that allows the creation of new Description Schemes and, possibly, Descriptors. It also allows the extension and modification of existing Description Schemes."

The DDL is not a modeling language, such as Unified Modeling Language (UML), but a schema language to represent the results of modeling audiovisual data, (i.e., DSs and Ds) as a set of syntactic, structural, and value constraints to which valid MPEG-7 descriptors, description schemes, and descriptions must conform.

According to the MPEG-7 DDL requirements [1], the DDL must be capable of expressing structural, inheritance, spatial, temporal, and conceptual relationships between the elements within a DS and between DSs. It must provide a rich model for links and references between one or more descriptions and the data that it describes. It must be platform and application independent, machine-readable, and preferably human-readable. It must be capable of specifying descriptor datatypes, both primitive (integer, text, date, time) and composite (histograms, enumerated types).

In addition, a DDL Parser is required which is capable of validating the syntax of MPEG-7 DSs (content and structure) and Descriptor datatypes. Given an MPEG-7 description, the parser must also be able to check its conformance to the rules expressed in the corresponding MPEG-7 DSs and Ds.

At the 51st MPEG Meeting in Noordwijkerhout in March 2000, it was decided to adopt the W3C's XML Schema Language as the MPEG-7 DDL. However, because XML Schema language has not been designed specifically for audiovisual content, certain extensions are necessary in order to satisfy all of the MPEG-7 DDL requirements. Hence, the DDL consists of the following components which are described in Sections III, IV, and V:

1) XML Schema structural components;
2) XML Schema datatypes;
3) MPEG-7-specific extensions.

In the remainder of this paper, we will describe the events which have led to the current DDL and its key components and features. Complete specifications can be found in the MPEG-7 DDL Committee Draft [2] and the W3C XML Schema Candidate Recommendations [3]–[5].

II. Historical Background

In response to the MPEG-7 Call for Proposals in October 1998, the MPEG-7 DDL evaluation team compared and evaluated ten DDL proposals at the MPEG-7 AHG Test and Evaluation Meeting in 1998. A summary report [6] was produced which concluded that XML [7] should be used as the syntax for the MPEG-7 DDL. There was also a consensus that the MPEG-7 DDL must support the validation of structural, relational and data typing constraints as well as the expression of semantics through the addition of richer constraints such as inheritance.

Although none of the proposals could satisfy all of the requirements, it was decided to base the DDL on DSTC's proposal, P547 [8], with the integration of ideas and components from other proposals and contributors. In addition, the strategy was to continue monitoring and liaising with related efforts in the W3C community, in particular the XML Schema [9], XLink [10], and XPath [11] Working Groups.

In May 1999, the XML Schema WG produced the first version of a 2-part working draft of the XML Schema language: XML Schema Part 1: Structures [4] and XML Schema Part 2: Datatypes [5]. Preliminary encoding of the Multimedia DSs [12], [13] using XML Schema language demonstrated its suitability as a basis for the DDL. However reservations were raised at the 48th MPEG Meeting in Vancouver in July 1999 concerning MPEG-7's dependency on the output and time schedule of W3C XML Schema WG. As a result, the decision was made to develop a proprietary MPEG-7-specific language in parallel with the XML Schema language developments within W3C. A new grammar based on DSTC's proposal but using MPEG-7 terminology (DeSs and Ds) and with modifications to ensure simple mapping to XML Schema, was developed. Based on this proprietary grammar, a Backus Naur Form (BNF), an XML DTD, nonvalidating parsers, and validation specifications were also developed.

Manuscript received September 2000; revised March 25, 2001.

The author is with the DSTC Pty Ltd., The University of Queensland, Qld. 4072, Australia (e-mail: jane@dstc.edu.au).

Publisher Item Identifier S 1051-8215(01)04995-3.

However, in the interests of interoperability, it was decided at the 51st MPEG Meeting in Noordwijkerhout in March 2000 to adopt XML Schema Language, with additional MPEG-7-specific extensions, as the DDL. This decision was made in recognition of the growing stability and expected widespread adoption of XML Schema, the availability of open source parsers and the release of Last Call for Review Working Drafts by the W3C.

A detailed evaluation of XML Schema revealed that although XML Schema satisfies the majority of the MPEG-7 DDL requirements, there are some existing features which are problematic and other features, required by MPEG-7, which are not satisfied. A list of problem issues and feature requests was submitted to the XML Schema WG in response to the Last Call for Review [14]. Certain high-priority features, which are not expected to be implemented within XML Schema, have been implemented as MPEG-7-specific extensions. Validation of these extensions will be through extensions to existing XML Schema parsers implemented during the development of MPEG-7 parsers.

Hence, the DDL consists of the following logical components which are described briefly in the next three sections:

1) XML Schema structural components;
2) XML Schema datatypes;
3) MPEG-7-specific extensions.

The complete detailed specifications of the DDL can be found in the MPEG-7 DDL Committee Draft [2] and the W3C XML Schema Candidate Recommendations [3]–[5].

III. XML Schema Structural Components

The purpose of this section is to describe the language constructs provided by XML Schema which can be used to constrain the content structure and attributes associated with MPEG-7 Description Schemes and Descriptors.

XML Schema consists of three categories of schema components. The primary components are:

1) namespaces and the xchema wrapper around the definitions and declarations;
2) element declarations;
3) attribute declarations;
4) type definitions: simple; complex; derived, anonymous.

The secondary components are:

1) attribute group definitions;
2) model group definitions;
3) identity-constraint definitions;
4) notation declarations.

The third group are the "helper" components which contribute to the other components and cannot stand alone:

1) annotations;
2) model groups;
3) particles;
4) wildcards.

In this paper, we only describe those features of most importance to MPEG-7: the primary components and group definitions. Details of other components not described here can be found in the XML Schema Candidate Recommendations [3]–[5].

A. Namespaces and the Schema Wrapper

XML namespaces [15] provide a simple method for referring to Ds and DSs from multiple different schemas so they can be used in descriptions or re-used to create new schemas. Namespace qualifiers associate elements and attributes with a particular namespace identified by a URI reference. Every schema definition must begin with a preamble in order to identify the current namespace. This also enables the generation of descriptions based on schemas which combine schema components from multiple different namespaces. The mandatory preamble consists of an XML element "schema" which includes the following attributes:

1) *xmlns*: a URI to the XML Schema namespace;
2) *targetNamespace*: the URI by which the current schema is to be identified;
3) *xmlns:mpeg7*: a URI to the MPEG7 DDL to be used for validation;
4) *xmlns*: References to other imported schemas and abbreviations for referring to definitions in these external schemas.

```
<schema
 xmlns="http://www.w3.org/2000/10/XMLSchema"
 xmlns:mpeg7=
  "http://www.mpeg7.org/2001/MPEG-7_Schema"
 targetNamespace=
  "http://www.mpeg7.org/2001/MPEG-7_Schema">
......
......
</schema>
```

B. Element Declarations

Element declarations enable the appearance in document instances of elements with specific names and types. An element declaration specifies a type definition for a schema element either explicitly or by reference, and may provide occurrence (*minOccurs* and *maxOccurs*) and default information (through the *default* attribute). For example, the element declaration below associates the name *Country* with an existing type definition, *countryCode*, specifies that the default value for the Country element is "en" (England) and that the *Country* element can occur zero or more times.

```
<element name="Country" type="countryCode"
   default="en" minOccurs="0"
   maxOccurs="unbounded"/>
```

The default values for minOccurs and maxOccurs are:

minOccurs = 1;

maxOccurs =

- unbounded, if the maxOccurs [attribute] equals unbounded;
- otherwise the number corresponding to the normalized value of the maxOccurs [attribute], if present,
- otherwise 1.

Sometimes it is preferable to reference an existing element rather than declare a new element.

```
<element ref="Country" minOccurs="1"/>
```

This declaration references an existing element (*Country*) that was declared elsewhere in the schema. In general, the value of the ref attribute must reference a global element, i.e., one that has been declared under *schema* rather than as part of a complex type definition. The consequence of this declaration is that an element called *Country* must appear at least once in an instance document, and its content must be consistent with that element's type, the *countryCode*.

C. Attribute Declarations

Attribute declarations enable the appearance in document instances of attributes with specific names and types by associating an attribute name with a simple datatype. Within an attribute declaration, the use attribute specifies presence $(required|optional|fixed|default|prohibited)$; and the *value* attribute indicates a fixed or default value. The default value of *use* is *optional*. The declaration which follows indicates that the appearance of a *lang* attribute is optional and its default value is "en-uk":

```
<attribute name="lang" type="language"
    use="default" value="en-uk"/ >
<complexType name="AnnotationType">
 <simpleContent>
  <extension base="string">
   <attribute ref="xml:lang"/>
  </extension>
 <simpleContent>
<complexType>
```

Following is a valid instance of an element of the above type:

```
<Annotation lang="en-us">
 Shower scene
<Annotation>
```

D. Type Definitions

In XML Schema there is a fundamental distinction between type definitions (which create new types) and declarations which enable the appearance in document instances of elements and attributes with specific names and types. Type definitions define internal schema components which can be used in other schema components such as element or attribute declarations or other type definitions. XML Schema provides *simple* and *complex* type definitions.

1) Simple Type Definitions: Simple types cannot have element content and cannot carry attributes. Both elements and attributes can be declared to have simple types. XML Schema provides a large number of simple types through a set of built-in primitive types and a set of built-in derived types (derived from the primitive types) [5]. These built-in datatypes are described in Section IV of this paper. In addition to the built-in simple types, new simple types can be derived by the application of restrictions such as the *enumeration* facet on a string or the *minInclusive* and *maxInclusive* facets on an integer, as shown in the following examples. The facets which are applicable to each datatype are listed in [5, Appendix C]:

```
<simpleType name="directionType">
 <restriction base="string">
  <enumeration value="uni-directional"/ >
  <enumeration value="bi-directional"/ >
 <restriction>
<simpleType>
<simpleType name="unsigned6">
 <restriction base="nonNegativeInteger">
  <minInclusive value="0"/ >
  <maxInclusive value="63"/ >
 <restriction>
<simpleType>
```

In addition to atomic (indivisible) simple types, XML Schema also provides two aggregate simple types: *list* types and *union* types. List types are composed of sequences of atomic types. Union types enable element or attribute values to be instances of a type drawn from the union of multiple atomic and list types. These two aggregate types are described in Sections IV-D and IV-E.

2) Complex Type Definitions: Unlike simple types, complex types allow children elements in their content and may carry attributes. Complex type definitions provide:

a) constraints on the appearance and content of attributes;
b) constraints on children elements and the content model of the type;
c) derivation of complex types from other simple or complex types through extension or restriction.

New complex types are defined using the *complexType* element and such definitions typically contain a set of element declarations, element references, and attribute declarations. For example, the complex type '"Organization," which is defined as follows, contains three element declarations and one attribute declaration

```
<complexType name="OrganizationType">
 <sequence>
  <element name="OrgName" type="string"/ >
  <element name="ContactPerson"
     type="IndividualType"
     minOccurs="1"
     maxOccurs="unbounded"/ >
  <element name="Address" type="PlaceType"
     minOccurs="1" maxOccurs="1"/ >
 <sequence>
 <attribute name="id" type="ID"
    use="required"/ >
<complexType>
<element name="ProdComp"
type="OrganizationType"/ >
```

The consequence of this definition is that any *ProdComp* elements appearing in an MPEG-7 description must consist of an *OrgName element*, one or more *ContactPerson* elements, and one *Address* element. The first of these elements will contain a string, the second will contain the complexType *IndividualType*, and the third will contain the complexType *PlaceType*. Any element whose type is declared to be *Organization* must also appear with an attribute called *id*, which must contain an *ID* (unique identifier). Below is an example of a valid instance

```
<ProdComp id="ORG754">
 <OrgName>DSTC Pty Ltd<OrgName>
 <ContactPerson>Liz Armstrong<ContactPerson>
 <Address>University of Qld<Address>
<ProdComp>
```

It is also possible to constrain the content model of a complexType to the following:

1) *empty:* no child elements only attributes;
2) *mixed:* character data appears between elements and their children. The *mixed* attribute must be set to "true;"
3) *complexContent:* the default content type which consists of elements and attributes;
4) *simpleContent:* use when deriving a complexType from a simpleType. This indicates that the new type contains only character data and no elements.

Following is an example of the *empty* content model and a valid instance of this example:

```
<element name="Price">
 <complexType>
  <attribute name="currency"
      type="currencyCode"/ >
  <attribute name="value" type="decimal"/ >
 <complexType>
<element>
<Price currency="EU" value="423.46"/ >.
```

Following is an example of the *mixed* content model and a valid instance of the example:

```
<element name="Introduction">
 <complexType mixed="true">
  <sequence>
   <element name="Name" type="string"/ >
  <sequence>
 <complexType>
<element>
<Introduction>Dear Ms.<Name>Hetty
Wilson<Name>, <Introduction>
```

3) Derived Types: It is possible to derive new complexTypes by extension or restriction of simple or complex base-type definitions. A complex type extends another by having additional content model particles at the end of the other definition's content model, or by having additional attribute declarations, or both. A new complexType can be derived by extending a simpleType through the addition of attributes. To indicate that the content model of the new type contains only character data and no elements, the *simpleContent* element is used. In the following example, the *source* and *target* attributes are added to the simple string base type to define the *RelationType*:

```
<complexType name="RelationType">
 <simpleContent>
  <extension base="string">
   <attribute name="source"
     type="IDREF" use="optional"/ >
   <attribute name="target"
     type="IDREF" use="optional"/ >
  <extension>
 <simpleContent>
<complexType>
<element name="Relation"
type="RelationType"/ >
<Relation source="edge1" target="dge2">
 morph
<Relation>
```

A new complexType may also be derived by extending an existing complexType. In the example below, the *Person* type is extended through the addition of a new *role* element, to create the *Creator* type. The *complexContent* element is required to indicate that we intend to restrict or extend the content model of a complex type and that the new type contains only elements and no character data

```
<complexType name="Creator">
 <complexContent>
  <extension base="PersonType">
   <sequence>
      <element name="role"
     type="ControlledTermType"/ >
   <sequence>
  <extension>
 <complexContent>
<complexType>
```

A type definition whose declarations or facets are in a one-to-one relation with those of another specified type definition, with each in turn restricting the possibilities of the one it corresponds to, is said to be a restriction. The specific restrictions might include narrowed ranges or reduced alternatives. Members of a type, A, whose definition is a restriction of the definition of another type, B, are always members of type B as well.

In the example below the *SimpleName* type is derived through restriction of the *PersonName* type. The restriction is on the occurrence constraints of the *Title* and *Forename* elements which are reduced from unbounded to 1

```
<complexType name="PersonNameType">
 <sequence>
```

```
  <element name="Title" minOccurs="0"/>
  <element name="Forename" minOccurs="0"
         maxOccurs="unbounded"/>
 <sequence>
<complexType>
 <complexType name="SimpleNameType">
 <complexContent>
  <restriction base="PersonNameType">
   <sequence>
    <element name="Title" minOccurs="1"
           maxOccurs="1"/>
    <element name="Forename" minOccurs="1"
            maxOccurs="1"/>
   <sequence>
  <restriction>
 <complexContent>
<complexType>
<PersonName>
 <Title>Prof.<Title>
 <Forename>Simon<Forename>
 <Forename>Daniel<Forename>
 <Forename>Kaplan<ForeName>
<PersonName>
<SimpleName>
 <Title>Prof.<Title>
 <Forename>Kaplan<Forename>
<SimpleName>
```

4) Anonymous Type Definitions: Schemas can be constructed by defining named types and then declaring elements that reference the types using the $element\ name = \cdots type = \cdots$construction. This style of schema construction is straightforward but it can become unwieldy if you define many types that are referenced only once and contain very few constraints. In these cases, a type can be more succinctly defined as an anonymous type which saves the overhead of having to be named and explicitly referenced

```
<element name="Affiliation">
 <complexType>
  <choice>
   <element name="Organization"
      type="OrganizationType"/>
   <element name="PersonGroup"
      type="PersonGroupType"/>
  <choice>
 <complexType>
<element>
```

5) Group Definitions: The *attributeGroup* and *group* elements provide mechanisms for creating and naming groups of attributes and groups of elements respectively. Such groups can then be incorporated by reference into complexType definitions

```
<attributeGroup name="id_href_Group">
  <attribute name="id" type="ID"/>
  <attribute name="href"
type="uriReference"/>
<attributeGroup>
<complexType name="Classification">
 <sequence>
  <element ref="Genre"/>
 <sequence>
 <attributeGroup ref="id_href_Group"/>
<complexType>
```

Three compositors are also provided to construct unnamed groups of elements.

Case 1) *Sequence*: constrains the elements in the group to appear in the same order in which they are declared;

Case 2) *Choice*: only one of the elements in the group may appear in an instance;

Case 3) *All*: all of the elements in the group may appear once or not at all and in any order.

In the example below, the ContactGroup is defined as a choice between two elements, *Organization* and *Person*. The *PublisherType* is then defined as a sequence of *ContactGroup* and *Address* and with an id attribute

```
<group name="ContactGroup">
 <choice>
   <element ref="Organization"/>
   <element ref="Person"/>
 <choice>
<group>
<complexType name="PublisherType">
 <sequence>
  <group ref name="ContactGroup"/>
  <element ref="Address"/>
 <sequence>
 <attribute name="id" type="ID"
      use="optional"/>
<complexType>
```

Example of a valid instance:

```
<Publisher id="ORG123">
 <Organization>DSTC Pty Ltd<Organization/>
 <Address>Uni. Of Qld<Address>
<Publisher>
```

IV. XML Schema Datatypes

This section describes the built-in primitive datatypes, the built-in derived datatypes and mechanisms for defining customized derived datatypes such as facets, lists and union datatypes. These facilities can be used to constrain the possible values of MPEG-7 descriptors within instantiated descriptions.

A. Built-In Primitive Datatypes

The following built-in primitive datatypes are provided within XML Schema:Datatypes:

1) string;

2) boolean;
3) float;
4) double;
5) decimal;
6) timeDuration [ISO 8601];
7) recurringDuration;
8) binary;
9) uriReference;
10) ID;
11) IDREF;
12) ENTITY;
13) QName.

B. Built-In Derived Datatypes

The following built-in datatypes, which have been derived from the primitive types listed above, are also provided:

1) CDATA;
2) token;
3) language [RFC 1766];
4) IDREFS;
5) ENTITIES;
6) NMTOKEN, NMTOKENS;
7) Name, NCName;
8) NOTATION;
9) integer, nonPositiveInteger, negativeInteger, nonNegativeInteger, positiveInteger;
10) long, unsignedLong;
11) int, unsignedInt;
12) short, unsignedShort;
13) byte, unsignedByte;
14) timeInstant, time, timePeriod;
15) date, month, year, century;
16) recurringDate, recurringDay.

C. Facets

A derived datatype is defined by applying constraining facets to a primitive datatype or another derived datatype. Table I lists the facets which are provided to generate customized datatypes.

The following example illustrates the application of the *minInclusive* and *maxInclusive* facets to a *float* datatype to restrict elements of type *height* to between 0.0 and 120.0:

```
<simpleType name="height" base="float">
 <minInclusive value="0.0"/>
 <maxInclusive value="120.0">
<simpleType>
```

The example below restricts elements of type *PhoneNum* to strings of three digits followed by a dash followed by four digits:

```
<simpleType name="PhoneNum" base="string">
 <pattern value= "\ d{3}-\ d{4}"/>
<simpleType>
```

TABLE I
FACETS PROVIDED TO GENERATE CUSTOMIZED DATA TYPES

Bounds facets	minInclusive, minExclusive maxInclusive, maxExclusive.
Numeric facets	precision, scale
Date/Time facets	duration, period
Pattern facet	pattern
Enumeration facet	enumeration
Length facets	length, minlength, maxlength
Encoding facet	encoding (hex or base64)
Whitespace facet	whitespace

D. The List Datatype

List types are composed of sequences of atomic types, separated by whitespace. XML Schema has three built-in list types, NMTOKENS, IDREFS, and ENTITIES. In addition, you can create new list types by derivation from existing atomic types. You cannot create list types from existing list types or from complex types but you can apply facets (*length*, *minLength*, *maxLength* and *enumeration*) to derive new list types.

Examples of list-type definitions and a valid instance

```
<simpleType name="integerVector">
 <list itemType="integer"/>
<simpleType>
<simpleType name="integerVector4">
 <restriction base"integerVector">
  <length value="4"/>
 <restriction>
<simpleType>
<integerVector4>5 8 11 2<integerVector4>
```

E. The Union Datatype

Union types enable element or attribute values to be one or more instances of one type drawn from the union of multiple atomic and list types. In the example below, the element Unsigned6OrDirection can have a value which is of either type Unsigned6 or Direction, because it is defined as the union of these two types

```
<element name="Unsigned6OrDirection">
 <simpleType>
  <union memberTypes="unsigned6
         directionType"/>
 <simpleType>
<element>
```

The examples below are both valid instances of this element:

```
<Unsigned6OrDirection>
 45
<Unsigned6OrDirection>
<Unsigned6OrDirection>
 unidirectional
<Unsigned6OrDirection>
```

V. MPEG-7-Specific Extensions

It has been necessary to add the following features to XML Schema in order to satisfy the MPEG-7 DDL requirements:

1) array and matrix datatypes;
2) typed references;
3) built-in derived datatypes such as enumerated datatypes for MimeType, CountryCode, RegionCode, CharacterSetCode, and CurrencyCode and a set of additional time datatypes.

A. Array and Matrix Datatypes

MPEG-7 requires a DDL mechanism to restrict the size of arrays and matrices to either a pre-defined facet value in a schema definition or to an attribute at time of instantiation. Using the *list* datatype, two methods are provided for specifying sizes of (1-D) arrays and multi-dimensional matrices. A new *mpeg7:dimension* facet, which is a list of positive integers, is provided to enable the specification of the dimensions of a fixed-size array or matrix. To ensure XML Schema, parsers ignore MPEG-7-specific extensions and MPEG-7 parsers validate them, then the <*annotation*> <*appinfo*> wrapper is required.

The following example illustrates the definition and instantiation of an integer matrix with three rows and four columns:

```
<simpleType name="IntMatrix2D">
 <list itemType="integer">
  <annotation><appinfo>
   <mpeg7:dimension value="unbounded
          unbounded"/>
  <appinfo><annotation>
 <list>
<simpleType>
<simpleType name="IntMatrix3x4">
 <restriction base="IntMatrix2D">
  <annotation><appinfo>
   <mpeg7:dimension value="3 4"/>
  <appinfo><annotation>
 <restriction>
<simpleType>
<element name="IntMat3x4"
type="IntMatrix3x4"/>
<IntMat3x4>
5 8 9 4
7 6 1 2
1 3 5 8
<IntMat3x4>
```

The special *mpeg7:dim* attribute is also provided to support parameterized array and matrix sizes. It specifies the dimensions to be applied to a list type at the time of instantiation and is defined in the mpeg7 namespace as a list of positive integers

```
<simpleType name="dim">
  <list itemType="integer"/>
<simpleType>
<complexType name="NDimIntegerArray">
 <simpleContent>
  <extension base="listOfInteger">
   <attribute ref="mpeg7:dim"/>
  <extension>
 <simpleContent>
<complexType>
<element name"IntegerMatrix"
   type="NDimIntegerArray"/>
```

In the following example, a matrix with 2 rows and 4 columns is specified at the time of instantiation using *mpeg7:dim*:

```
<IntegerMatrix mpeg7:dim="2 4">
  1 2 3 4
  5 6 7 8
<IntegerMatrix>
```

B. Typed References

The *mpeg7:refType* facet provides a way to check the type of a referenced element. The type of the referenced element must be of the type specified by the value of *refType*, or a type derived from that type. In the example that follows, the value of element SegmentRef must be an IDREF to a SegmentType element:

```
<simpleType name="IdRefSegment">
 <restriction base="IDREF">
  <annotation><appinfo>
   <mpeg7:refTypvalue="mpeg7:SegmentType"/>
  <appinfo><annotation>
 <restriction>
<simpleType>
<element name="SegmentRef"
   type="mpeg7:IdRefSegment"/>
```

C. Built-In Derived Datatypes

In addition to the built-in derived types provided by XML Schema:Datatypes, the following built-in datatypes are also provided to explicitly satisfy the requirements of MPEG-7 implementors:

1) mimeType—IANA list of Mime Types [16], [17];
2) country Code—ISO3166-1:1997 [18];
3) region Code—ISO3166-2:1998;
4) currency Code—ISO4217:1995;
5) characterSetCode—IANA List of Character Sets [19].

The following MPEG-7-specific time datatypes are also currently under consideration for incorporation within the DDL:

1) basicTimePoint;
2) basicDuration.

VI. Current Status

The DDL is currently at the final Committee Draft stage and is expected to move to final Draft International Standard in July 2001. Issues which still require resolution include such problems as completion of an open source MPEG-7 parser, clarification, and location of MPEG-7 datatypes, data inheritance mechanisms, and the identification and handling of unused XML Schema features.

Another perceived problem is the instability of XML Schema. It is currently at the Candidate Recommendation stage, but is expected to move to Proposed Recommendation in the first half of 2001 and to a Recommendation later in 2001. The MPEG-7 DDL will reflect any changes to XML Schema as it progresses from CR to PR and finally to R but these are expected to be minimal. In addition, the MPEG-7 DDL group will continue to work with and provide feedback to the XML Schema Working Group to ensure that the MPEG communities' needs are considered and met as far as possible.

Detailed descriptions of the other parts of the MPEG-7 standard can be found in [20].

REFERENCES

[1] MPEG Requirements Group, "55th MPEG Meeting," MPEG-7 requirements Document V.13, Pisa, Italy, Doc. ISO/MPEG N3933, Jan. 2001.

[2] Text of ISO/IEC CD 15938-2 information technology—Multimedia content description interface: Description definition language. (2000, Oct.) 54th MPEG Meeting, La Baule, France. [Online]. Available: http://www.cselt.it/mpeg/public/mpeg-7_ddl_cd. zip

[3] XML Schema Part 0: Primer, W3C Candidate Recommendation (2000, Oct.). [Online]. Available: http://www.w3.org/TR/xmlschema-0/

[4] XML Schema Part 1: Structures, W3C Candidate Recommendation (2000, Oct.). [Online]. Available: http://www.w3.org/TR/xmlschema-1/

[5] XML Schema Part 2: Datatypes, W3C Candidate Recommendation (2000, Oct.). [Online]. Available: http://www.w3.org/TR/xmlschema-2/

[6] MPEG7 Requirements Group, "47th MPEG Meeting," ISO/IEC JTC1/SC29/WG11 N2730, Seoul, Korea, Mar. 1999.

[7] Extensible Markup Language (XML) 1.0, W3C Recommendation (1998, Feb.). [Online]. Available: http://www.w3.org/TR/REC-xml

[8] J. Hunter, "A proposal for an MPEG-7 DDL," presented at the P547, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, U.K., Feb. 1999.

[9] XML Schema Working Group (1998). [Online]. Available: http://www.w3.org/XML/Group/Schemas.html

[10] XML Linking Language (XLink) W3C Proposed Recommendation (2000, Dec.). [Online]. Available: http://www.w3.org/TR/xlink/

[11] XML Path Language (XPath) Version 1.0 W3C Recommendation (1999, Nov.). [Online]. Available: http://www.w3.org/TR/xpath/

[12] "MPEG-7 multimedia description schemes XM (v6.0)," presented at the 55th MPEG Meeting, ISO/IEC JTC1/SC29/WG11 w3815, Pisa, Italy, Jan. 2001.

[13] MPEG-7 Multimedia Description Scheme Group, "Study of MPEG-7 multimedia description schemes CD (v1.0)," 55th MPEG Meeting, Pisa, Doc. ISO/IEC JTC1/SC29/WG11 w3816, Jan. 2001.

[14] MPEG-7 DDL Response to XML Schema Last Call for Review (2000, May). [Online]. Available: http://archive.dstc.edu.au/mpeg7-ddl/issues.html

[15] W3C Recommendation (1999). [Online]. Available: http://www.w3.org/TR/REC-xml-names/

[16] IANA List of Mime Types (2001). [Online]. Available: ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types

[17] Internet Draft (2000, May). [Online]. Available: http://www.ietf.org/internet-drafts/draft-murata-xml-04.txt

[18] ISO Country Codes ISO31661:1997 (1997). [Online]. Available: http://www.din.de/gremien/nas/nabd/iso3166ma/codlstp1/en_listp1.html

[19] IANA List of Character Sets (2001). [Online]. Available: http://www.iana.org/assignments/character-sets

[20] *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 685–772, June 2001.