INTERNATIONAL STANDARD

ISO/IEC 21000-2

First edition
2003-03-15

Information technology — Multimedia framework (MPEG-21) —

Part 2: Digital Item Declaration

Technologies de l'information — Cadre multimédia (MPEG-21) —

Partie 2: Déclaration d'article numérique

Reference number
ISO/IEC 21000-2:2003(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

List of Tables

Page

List of Figures

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 21000-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 21000 consists of the following parts, under the general title *Information technology — Multimedia framework (MPEG-21)*:

— *Part 1: Vision, Technologies and Strategy*

— *Part 2: Digital Item Declaration*

— *Part 3: Digital Item Identification*

— *Part 4: Intellectual Property Management and Protection (IPMP)*

— *Part 5: Rights Expression Language*

— *Part 6: Rights Data Dictionary*

— *Part 7: Digital Item Adaptation*

— *Part 8: Reference Software*

Executive Summary for MPEG-21

Today, many elements exist to build an infrastructure for the delivery and consumption of multimedia content. There is, however, no “big picture” to describe how these elements, either in existence or under development, relate to each other. The aim for MPEG-21 is to describe how these various elements fit together. Where gaps exist, MPEG-21 will recommend which new standards are required. ISO/IEC JTC 1/SC 29/WG 11 (MPEG) will then develop new standards as appropriate while other relevant standards may be developed by other bodies. These specifications will be integrated into the multimedia framework through collaboration between MPEG and these bodies.

The result is an open framework for multimedia delivery and consumption, with both the content creator and content consumer as focal points. This open framework provides content creators and service providers with equal opportunities in the MPEG-21 enabled open market. This will also be to the benefit of the content consumer providing them access to a large variety of content in an interoperable manner.

The vision for MPEG-21 is to define a multimedia framework *to enable transparent and augmented use of multimedia resources across a wide range of networks and devices* used by different communities.

This second part of MPEG-21 (ISO/IEC 21000-2) specifies the mechanism for declaring the structure and makeup of Digital Items.

Information technology — Multimedia framework (MPEG-21) —

Part 2:
Digital Item Declaration

1 Scope

This document describes the MPEG-21 Digital Item Declaration technology, which is part 2 of the MPEG-21 standard.

1.1 Organization of the document

This technology is described in three normative clauses:

- Model: The Digital Item Declaration Model (clause 6) describes a set of abstract terms and concepts to form a useful model for defining Digital Items. Within this model, a Digital Item is the digital representation of "a work", and as such, it is the thing that is acted upon (managed, described, exchanged, collected, etc.) within the model.
- Representation: Clause 7 contains the normative description of the syntax and semantics of each of the Digital Item Declaration elements, as represented in XML. This clause also contains some non-normative examples for illustrative purposes.
- Schema: Clause 8 contains the normative XML schema comprising the entire grammar of the Digital Item Declaration representation in XML.

In addition, illustrative (non-normative) examples are provided.

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.

Extensible Markup Language 1.0 (Second Edition), W3C Recommendation, 6 October 2000

XML Schema Part 1: Structures and Part 2: Datatypes, W3C Recommendation, 2 May 2001

Canonical XML Version 1.0, W3C Recommendation, 15 March 2001

Uniform Resource Identifiers (URI): Generic Syntax, IETF RFC 2396, 1998

Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. IETF RFC 2045, 1996

3 Terms and definitions

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

3.1 Digital Item

In ISO/IEC 21000-1:2001 (part 1 of MPEG-21: Vision, Technologies and Strategy), Digital Items are defined as structured digital objects, including a standard representation and identification, and meta-data. This entity is the fundamental unit of distribution and transaction within the MPEG-21 framework as a whole; it has, however, no further technical meaning. Within this document (part 2 of MPEG-21: Digital Item Declaration), an *item* is a grouping of sub-*items* and/or *components* that are bound to relevant *descriptors*, as defined within the Digital Item Declaration Model. The term *item* is a technical term, and is, as such, a narrower term than Digital Item. In conclusion, the use of the two different terms Digital Item and *item* within MPEG-21 is consistent and intended.

4 Conventions

4.1 Naming convention

It should be noted that the Digital Item Declaration Model (clause 6) contains the concept names that are used throughout the MPEG-21 standard. As such, this model should be considered to be the "ultimate arbiter" of these MPEG-21 concept names.

4.2 Documentation convention

The semantics of each element in the Digital Item Declaration Model is specified using the constructs provided by EBNF [4], and is shown in this document using a specific font and background:

```
element ::= (part1 | part2)+ part3*
```

The syntax of each element in the Digital Item Declaration Representation is specified using the constructs provided by XML Schema [2].

Element names and attribute names in the representation are in SMALL CAPS. Throughout the document, *italics* are used when referring to elements defined in the Digital Item Declaration Model (see clause 4), hereafter known as the Model.

The syntax of each element in the Digital Item Declaration representation is specified using the following format.

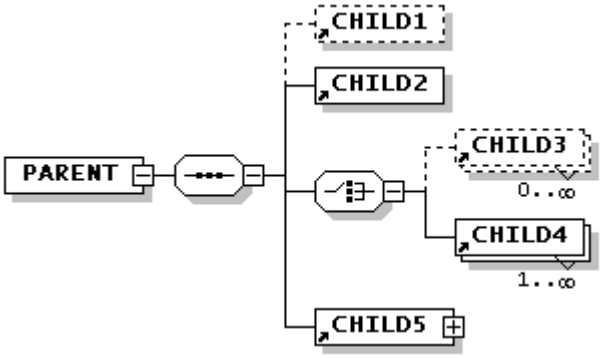
<table>
<tr><td>Diagram</td><td colspan="3"> 
</td></tr>
<tr><td>Children</td><td colspan="3"><CHILD1> <CHILD2> <CHILD3> <CHILD4> <CHILD5></td></tr>
<tr><td>Used by</td><td colspan="3"><GRANDPARENT1> <GRANDPARENT2></td></tr>
<tr><td rowspan="2">Attributes</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>ID</td><td>ID</td><td>A unique ID value, which can be referenced by another element.</td></tr>
<tr><td>Source</td><td colspan="3"><pre>
<xsd:element name="PARENT">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="CHILD1" minOccurs="0"/>
            <xsd:element ref="CHILD2"/>
            <xsd:choice>
                <xsd:element ref="CHILD3" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="CHILD4" minOccurs="1" maxOccurs="unbounded"/>
            </xsd:choice>
            <xsd:element ref="CHILD5"/>
        </xsd:sequence>
        <xsd:attribute name="ID" type="xsd:id"/>
    </xsd:complexType>
</xsd:element>
</pre></td></tr>
</table>

Table 1 — Example element specification

The Language Definition clause contains syntax diagrams for each element. Here is an example syntax diagram with annotations:

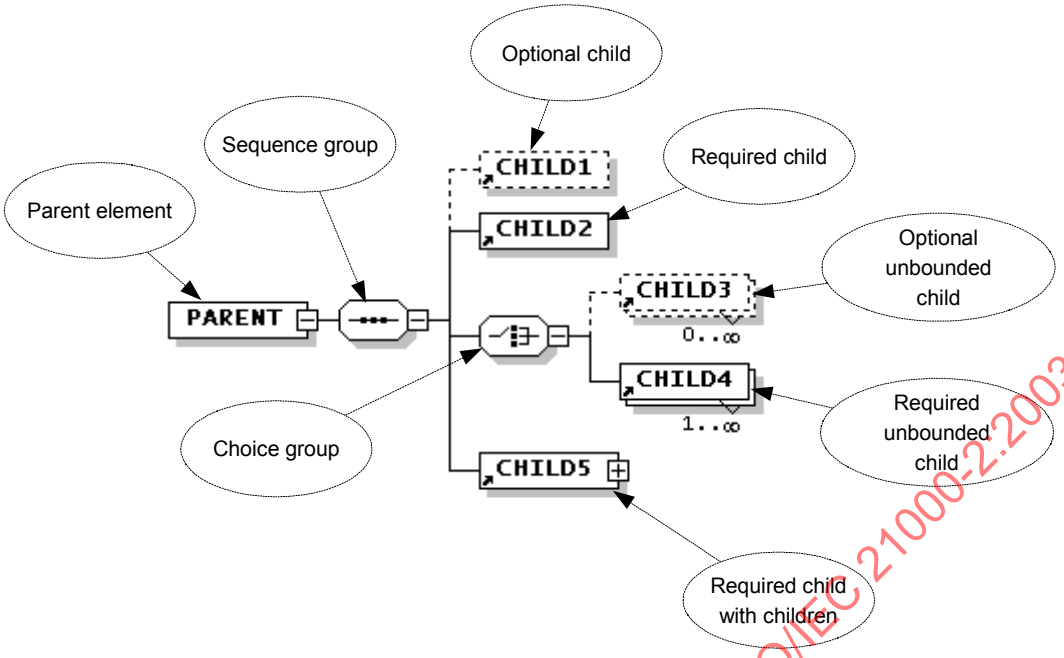


Figure 1 — Example element syntax diagram

Non-normative examples are included in separate clauses, and are shown in this document using a separate font and background:

```
<Example attribute1="example attribute value">
   <Element1>example element content</Element1>
</Example>
```


5 Symbols and abbreviated terms

For the purposes of this document, the following abbreviations apply.

DID: Digital Item Declaration

DIDL: Digital Item Declaration Language

EBNF: Extended Backus-Naur Form

IANA: Internet Assigned Numbers Authority

IPMP: Intellectual Property Management and Protection

JPEG: Joint Photographic Experts Group

MPEG: Moving Picture Experts Group

MPEG-21: ISO/IEC 21000 (all parts)

MP3: MPEG-1/2 layer III (audio coding)

URI: Uniform Resource Identifier (IETF Standard is RFC 2396)

URL: Uniform Resource Locator (IETF Standard is RFC 1738)

URN: Uniform Resource Name (IETF Standard is RFC 2396)

XML: Extensible Markup Language (W3C Recommendation)

6 Digital Item Declaration Model

6.1 Purpose and Overview

The purpose of this clause is to describe a set of abstract terms and concepts to form a useful model for defining Digital Items. Within this model, a Digital Item is the digital representation of "a work", and as such, it is the thing that is acted upon (managed, described, exchanged, collected, etc.) within the model. The goal of this model is to be as flexible and general as possible, while providing for the "hooks" that enable higher level functionality and interoperability. This, in turn, will allow the model to serve as a key foundation in the building of higher level models in other MPEG-21 elements (such as Identification or IPMP). This model specifically does not define a language in and of itself. Instead, the model helps to provide a common set of abstract concepts and terms that can be used to define such a scheme, or to perform mappings between existing schemes capable of Digital Item Declaration, for comparison purposes.

6.2 Abstract Model

Please note that in the descriptions below, the defined elements in *italics* are intended to be unambiguous terms within this model. The prose descriptions define the semantic "meaning" of the terms, and the EBNF representations define the precise intended relationship or structure between terms within the model.

6.2.1 Container

A *container* is a structure that allows *items* and/or *containers* to be grouped. These groupings of *items* and/or *containers* can be used to form logical packages (for transport or exchange) or logical shelves (for organization). *Descriptors* allow for the "labelling" of *containers* with information that is appropriate for the purpose of the grouping (e.g. delivery instructions for a package, or category information for a shelf).

It should be noted that a *container* itself is not an *item*; *containers* are groupings of *items* and/or *containers*.

```
container ::= container* item* descriptor*
```

6.2.2 Item

An *item* is a grouping of sub-*items* and/or *components* that are bound to relevant *descriptors*. *Descriptors* contain information about the *item*, as a representation of a work. *Items* may contain *choices*, which allow them to be customized or configured. *Items* may be conditional (on *predicates* asserted by *selections* defined in the *choices*). An *item* that contains no sub-*items* can be considered an entity -- a logically indivisible work. An *item* that does contain sub-*items* can be considered a compilation -- a work composed of potentially independent sub-parts. *Items* may also contain *annotations* to their sub-parts.

The relationship between *items* and Digital Items (as defined in ISO/IEC 21000-1:2001, MPEG-21 Vision, Technologies and Strategy) could be stated as follows: *items* are declarative representations of Digital Items.

```
item ::= (item | component)* choice* descriptor* condition* annotation*
```

6.2.3 Component

A *component* is the binding of a *resource* to a set of *descriptors*. These *descriptors* are information related to all or part of the specific *resource* instance. Such *descriptors* will typically contain control or structural information about the *resource* (such as bit rate, character set, start points or encryption information) but not information describing the "content" within.

It should be noted that a *component* itself is not an *item*; *components* are building blocks of *items*.

```
component ::= resource descriptor* anchor* condition*
```

6.2.4 Anchor

An *anchor* binds *descriptors* to a *fragment,* which corresponds to a specific location or part of a *resource*.

```
anchor ::= fragment descriptor* condition*
```

6.2.5 Descriptor

A *descriptor* associates information with the enclosing element. This information may be a *component* (such as a thumbnail of an image, or a text *component*), or a textual *statement.*

```
descriptor ::= descriptor* (component | statement) condition*
```

6.2.6 Condition

A *condition* describes the enclosing element as being optional, and links it to the *selection*(s) that affect its inclusion. Multiple *predicates* within a *condition* are combined as a conjunction (an AND relationship). Any *predicate* can be negated within a *condition*. Multiple *conditions* associated with a given element are combined as a disjunction (an OR relationship) when determining whether to include the element.

```
condition ::= predicate+
```

6.2.7 Choice

A *choice* describes a set of related *selections* that can affect the configuration of an *item*. The *selections* within a *choice* are either exclusive (choose exactly one) or inclusive (choose any number, including all or none).

```
choice ::= selection+ descriptor* condition*
```

6.2.8 Selection

A *selection* describes a specific decision that will affect one or more *conditions* somewhere within an *item*. If the *selection* is chosen, its predicate becomes true; if it is not chosen, its *predicate* becomes false; if it is left unresolved, its *predicate* is undecided.

```
selection ::= predicate descriptor* condition*
```

6.2.9 Annotation

An *annotation* describes a set of information about another identified element of the model without altering or adding to that element. The information can take the form of *assertions*, *descriptors*, and *anchors*.

```
annotation ::= assertion* descriptor* anchor*
```

6.2.10 Assertion

An *assertion* defines a full or partially configured state of a *choice* by asserting true, false or undecided values for some number of *predicates* associated with the *selections* for that *choice*.

```
assertion ::= predicate*
```

6.2.11 Resource

A *resource* is an individually identifiable asset such as a video or audio clip, an image, or a textual asset. A *resource* may also potentially be a physical object. All *resources* must be locatable via an unambiguous address.

6.2.12 Fragment

A *fragment* unambiguously designates a specific point or range within a *resource*. *Fragment* may be *resource* type specific.

6.2.13 Statement

A *statement* is a literal textual value that contains information, but not an asset. Examples of likely *statements* include descriptive, control, revision tracking or identifying information (such as an identifier as described in any other normative part of ISO/IEC 21000).

6.2.14 Predicate

A *predicate* is an unambiguously identifiable declaration that can be true, false or undecided.

The following diagram is an example showing the most important elements within this model, how they are related, and as such, the hierarchical structure of the Digital Item Declaration Model.

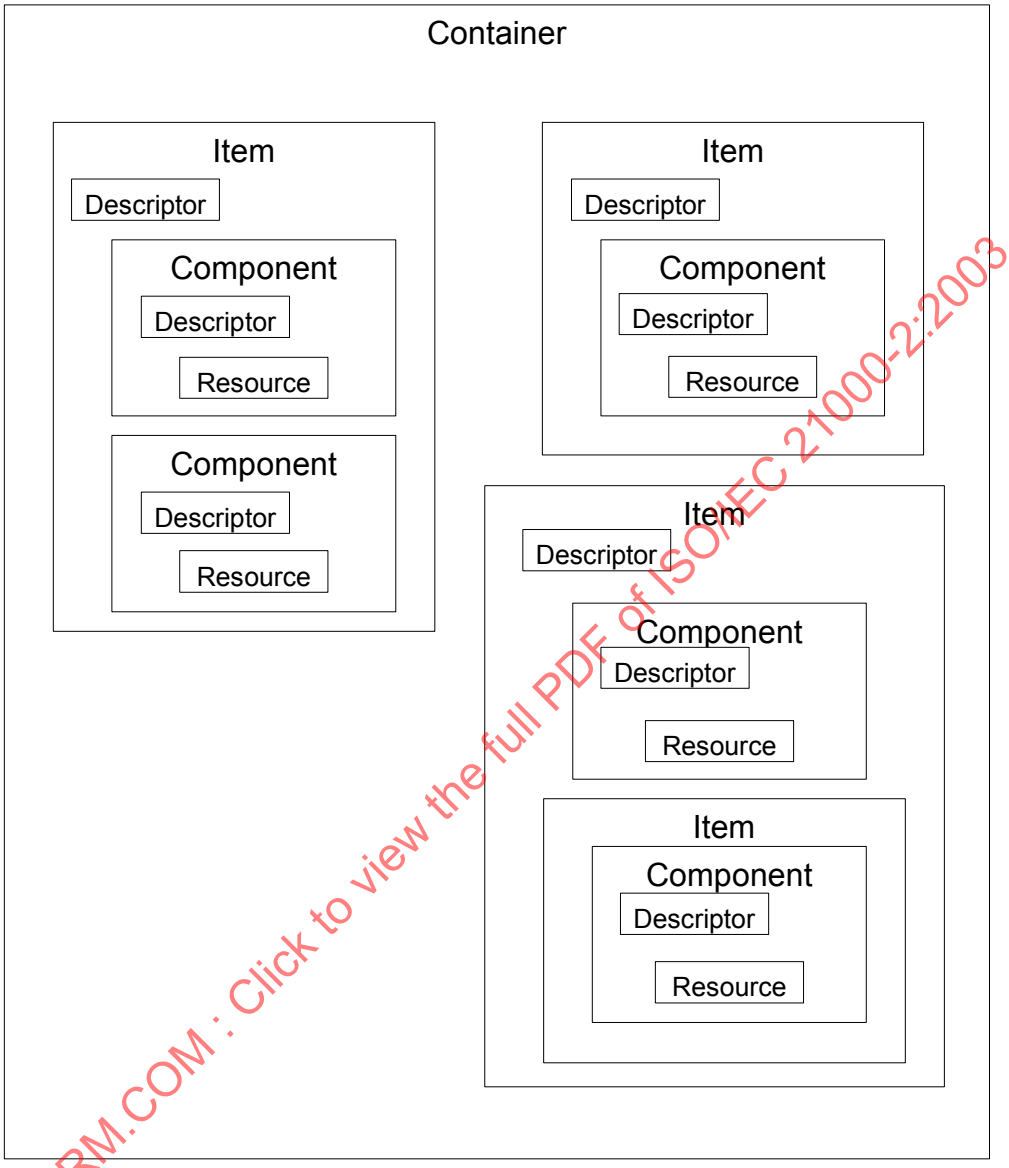


Figure 2 — Example Digital Item Declaration model

7 Digital Item Declaration Representation

7.1 Introduction

The purpose of this clause is to describe the XML schema for declaring Digital Items. The goal of this schema is to be as flexible and general as possible, while providing the "hooks" for higher level functionality that will allow it to serve as a key foundation in the building of higher level schema in other MPEG-21 domains (such as Identification or Rights Management).

7.1.1 DIDL Overview

DIDL documents are XML 1.0 [1] documents. The reader is assumed to be familiar with the terms and concepts of XML 1.0.

In addition, DIDL syntax is based on an abstract structure defined in the Digital Item Declaration Model (see clause 6 above). The following abstract elements defined in the Model are each represented in DIDL by a like-named DIDL element:

- Container
- Item
- Component
- Anchor
- Descriptor
- Choice
- Selection
- Condition
- Annotation
- Assertion
- Resource
- Statement

For example, the abstract *descriptor* element in the Model is represented in DIDL by the DESCRIPTOR element. Therefore, the reader is likewise assumed to be familiar with the terms and concepts defined in the Model.

A DIDL document consists of a DIDL root element with a single ITEM child element or CONTAINER child element. Thus, a DIDL document can represent either an *item* or a *container*.

In addition, DIDL defines the following special element types that do not correspond to any of the Model elements: REFERENCE, and DECLARATIONS. These special elements are used for specific purposes within DIDL.

The REFERENCE element is used to link the contents of an element inside another element. The linkage is by reference, rather than by value. References can be made to elements within a document, or to elements in a different document. The former type of reference is known as an internal reference; the latter is known as an external reference. An internal reference allows a single source to be maintained for an element that occurs in

more than one place in a DIDL document. An external reference allows a DIDL document to be split up into multiple linked discrete documents.

The DECLARATIONS element is used to define a set of DIDL elements in a document without actually instantiating them. A declared element (i.e. a child element of a DECLARATIONS element) is not considered to be instantiated unless it is referenced (by a REFERENCE element).

DIDL makes broad use of XML's ID attribute type. Generally, attributes of this type are used to make an internal association between one DIDL element and another. For example, many DIDL elements have an ID attribute, which makes them available as targets of internal references by REFERENCE elements, and, in limited cases, available for annotation by ANNOTATION elements. In addition, other attributes of type ID that are not named 'id' are used to make specific kinds of associations between specific elements. For example, the SELECT_ID attribute of the SELECTION element allows CONDITION elements to be associated with specific SELECTIONS. It is strongly recommended that attributes of type ID be assigned globally unique values, in order to avoid collisions when DIDL documents are merged. This is especially important when there are external dependencies on the ID values, such as a signature on the DIDL document, or an external reference from some other document or resource. Finally, it is noted that the use of the ID attribute should not be confused with normative identification mechanisms defined in any other part of ISO/IEC 21000.

7.2 DIDL Definition

7.2.1 Validation

Validating a document against the DIDL schema is necessary, but not sufficient, to determine its validity with respect to DIDL. After a document is validated against the DIDL schema, it must also be subjected to additional validation rules. These additional rules are given below in the descriptions of the elements to which they pertain.

7.2.2 Canonicalization

Like any XML document, a single logical DIDL document may be manifested in a wide variety of syntactic representations. Although the various syntactic representations each contain a different sequence of characters, they are all logically equivalent. In certain applications, such as generating a digest value for a digital signature on all or part of a DIDL document, it is necessary to define a method for generating a single predictable (deterministic) syntactic representation. Achieving this result for DIDL documents containing internal references requires special consideration. The following canonicalization method takes this consideration into account:

Canonical XML 1.0 [3] with the following additional constraint:

> All internal references are syntactically resolved; that is, the logical replacement is reflected in the Canonical syntactic representation. The REFERENCE tags corresponding to internal references are removed in this process.

This canonicalization method is identified by the following URN: "urn:mpeg:mpeg21:2002:01:did-canonicalization". This URN identifier is for use by any part of ISO/IEC 21000 that requires canonicalization, or by any other application with similar requirements.

7.2.3 Element Descriptions

The following basic principles apply to all element types:

— Any element with an ID attribute may have a REFERENCE child.

— Wherever DESCRIPTOR children are allowed, they are always the first children (except where preceded by CONDITION).

— Elements containing a REFERENCE inherit attribute values from the reference target for any attributes that the referring element does not specify. Because of this, elements that allow a REFERENCE child do not have any required attributes.

7.2.4 <DIDL>

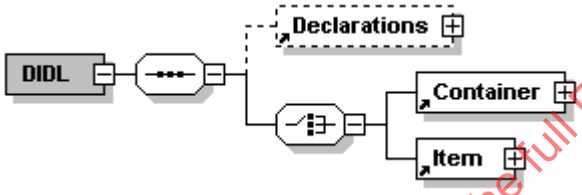
The DIDL element is the root element of a DIDL instance document. The DIDL root element may contain an optional DECLARATIONS element, followed by exactly one CONTAINER or ITEM.

The DIDL element must include a namespace declaration that declares the DIDL namespace for the root DIDL element and its contents. This is required so that applications will recognize the document as a DIDL document, that is covered by this specification. The DIDL namespace URI is "urn:mpeg:mpeg21:2002:01-DIDL-NS". The "01" represents a serial number that is expected to change as the DIDL schema evolves along with this part of ISO/IEC 21000.

The namespace declaration can take the form of a default namespace declaration, or a prefix-specific namespace declaration, as shown respectively in the following two examples:

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
...
</DIDL>
```

```
<didl:DIDL xmlns:didl="urn:mpeg:mpeg21:2002:01-DIDL-NS">
...
</didl:DIDL>
```

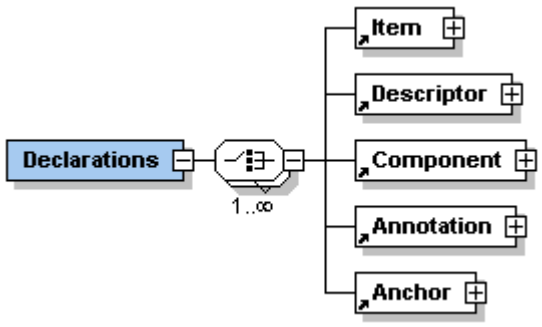

Diagram	
Children	<Declarations> <Container> <Item>

Source	`<xsd:element name="DIDL">`
	`  <xsd:complexType>`
	`  <xsd:sequence>`
	`  <xsd:element ref="Declarations" minOccurs="0"/>`
	`  <xsd:choice>`
	`  <xsd:element ref="Container"/>`
	`  <xsd:element ref="Item"/>`
	`  </xsd:choice>`
	`  </xsd:sequence>`
	`  </xsd:complexType>`
	`</xsd:element>`

Table 2 — DIDL element syntax

7.2.5 <Declarations>

The DECLARATIONS element is used to define a set of DIDL elements - without instantiating them - for later use in a document via an internal reference (see REFERENCE element).

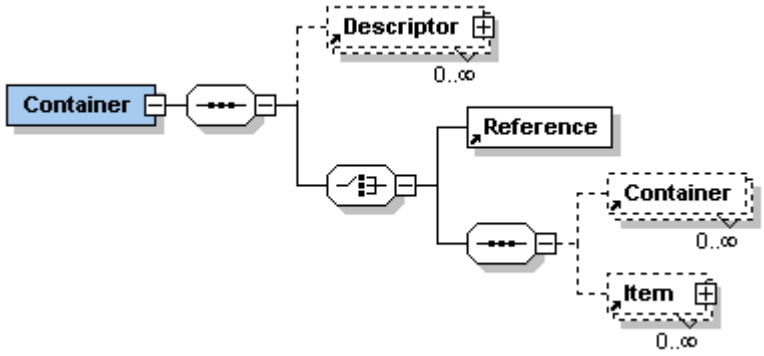
Diagram	
Children	<Item> <Descriptor> <Component> <Annotation> <Anchor>
Used by	<DIDL>

Source	`<xsd:element name="Declarations">`
	`  <xsd:complexType>`
	`  <xsd:choice maxOccurs="unbounded">`
	`  <xsd:element ref="Item"/>`
	`  <xsd:element ref="Descriptor"/>`
	`  <xsd:element ref="Component"/>`
	`  <xsd:element ref="Annotation"/>`
	`  <xsd:element ref="Anchor"/>`
	`  </xsd:choice>`
	`  </xsd:complexType>`
	`</xsd:element>`

Table 3 — DECLARATIONS element syntax

7.2.6 <Container>

The CONTAINER element represents a *Container*. As such, it is a grouping of ITEMS and/or possibly other CONTAINERS, bound with a set of DESCRIPTORS that contain descriptive information about the *container*.

<table>
<tr><td>Diagram</td><td> 
</td></tr>
<tr><td>Children</td><td><Descriptor> <Reference> <Container> <Item></td></tr>
<tr><td>Used by</td><td><Container> <DIDL></td></tr>
<tr><td>Attributes</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>id</td><td>ID</td><td>A unique ID value.</td></tr>
</table>
</td></tr>
<tr><td>Source</td><td>

```
<xsd:element name="Container">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Descriptor" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:choice>
                <xsd:element ref="Reference"/>
                <xsd:sequence>
                    <xsd:element ref="Container" minOccurs="0"
                        maxOccurs="unbounded"/>
                    <xsd:element ref="Item" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:choice>
        </xsd:sequence>
        <xsd:attributeGroup ref="ID_ATTRS"/>
    </xsd:complexType>
</xsd:element>
```

</td></tr>
</table>

Table 4 — CONTAINER element syntax

7.2.7 <Item>

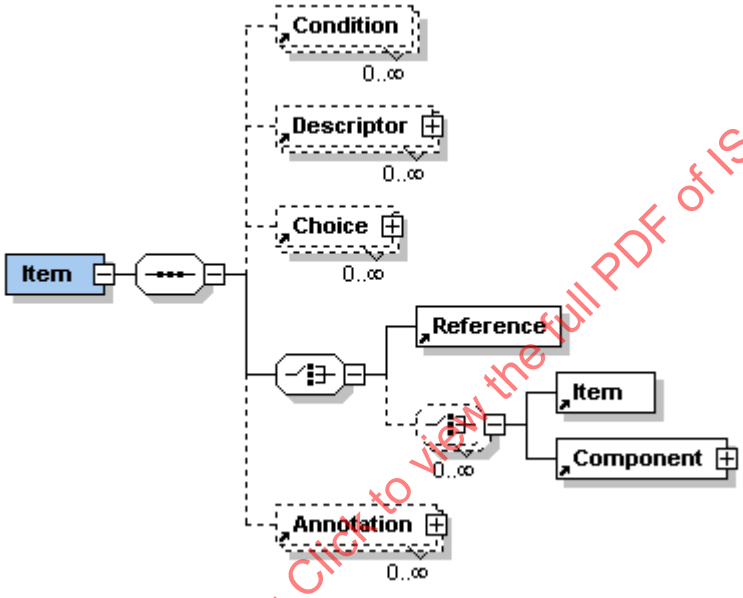
An ITEM element represents an *Item*. As such, it is a grouping of possible sub-ITEMS and/or COMPONENTS, bound to a set of relevant DESCRIPTORS containing descriptive information about the *item*. In addition, an ITEM can be made conditional via a set of CONDITION child elements, made configurable via a set of CHOICE elements, and annotated via a set of ANNOTATION elements.

Items are intended to be the lowest level of granularity transacted by Users within the MPEG-21 framework.

Validation Rule:

— An ITEM element cannot be conditional on any of its descendant SELECTION elements. In other words, an ITEM cannot contain a CONDITION element specifying a SELECT_ID value that identifies any descendant SELECTION element within the ITEM.

Diagram



Children <Condition> <Descriptor> <Choice> <Reference> <Item> <Component> <Annotation>

Used by <Declarations> <Item> <Container> <DIDL>

Attributes

Name	Type	Description
id	ID	A unique ID value.

Source

```
<xsd:element name="Item">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Condition" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element ref="Descriptor" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element ref="Choice" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:choice>
                <xsd:element ref="Reference"/>
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element ref="Item"/>
                    <xsd:element ref="Component"/>
```

	</xsd:choice> </xsd:choice> <xsd:element ref="Annotation" minOccurs="0" maxOccurs="unbounded"/> </xsd:sequence> <xsd:attributeGroup ref="ID_ATTRS"/> </xsd:complexType> </xsd:element>

Table 5 — ITEM element syntax

Example (informative):

The following example illustrates how CONTAINERS and ITEMS can be used to represent a *container* of some kind that contains a single composite *item* – a "compilation." The CONTAINER represents the shelf or package, the outermost ITEM represents the composite *item* as a whole, and the inner ITEMS represent the individual *items* that make up the compilation.

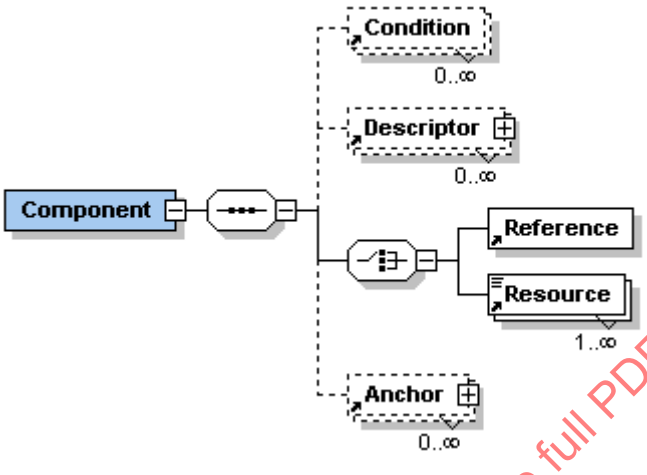
```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS ">
  <Container>
    <Item>
      <Item>
        .
        .
        .
      </Item>
      <Item>
        .
        .
        .
      </Item>
    </Item>
  </Container>
</DIDL>
```

7.2.8 <Component>

A COMPONENT element represents a *Component*. As such, it groups a RESOURCE element with a set of DESCRIPTORS containing descriptive information about the *resource*, plus a set of ANCHORS specifying points or regions of interest in the *resource*. The COMPONENT, being a logical union of a *resource* with relevant descriptive data and *anchors*, is intended to be the basic building block of digital content within a DIDL document.

If multiple RESOURCE children are present, they are considered equivalent and any one of them may be used. An agent may discriminate between them using specific information it has about retrieval from these sources, or using such information present in a DESCRIPTOR.

Diagram



Children <Condition> <Descriptor> <Reference> <Resource> <Anchor>

Used by <Declarations> <Descriptor> <Item>

Attributes

Name	Type	Description
id	ID	A unique ID value.

Source

```
<xsd:element name="Component">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Condition" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element ref="Descriptor" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:choice>
                <xsd:element ref="Reference"/>
                <xsd:element ref="Resource" minOccurs="1"
                    maxOccurs="unbounded"/>
            </xsd:choice>
            <xsd:element ref="Anchor" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="ID_ATTRS"/>
    </xsd:complexType>
</xsd:element>
```

Table 6 — COMPONENT element syntax

7.2.9 <Resource>

A RESOURCE element represents a *Resource*. As such, it defines an individually identifiable asset such as a video or audio clip, an image, an electronic ticket, or a textual work.

Normally, a resource is defined in a RESOURCE element by reference, by specifying the resource's URI in the REF attribute. The URI identifies the resource for the purpose of allowing an application to retrieve the resource's contents. This URI can be a traditional URL, which gives the explicit physical location from which to retrieve the contents, or a more abstract identifier, such as a URN, which identifies the resource contents independent from their location.

The data type of the resource is identified by the MIMETYPE attribute, which is a MIME media type identifier, as defined in RFC 2045 (e.g. 'video/mpeg').

The LOCALPATH attribute specifies the required relative location for a local copy of the *resource*. This is useful when there are location dependencies between *resources*, such as an HTML file that references a set of JPEG images in some directory hierarchy.

It is also possible to define a *resource* by value, by including the *resource*'s data as character data within the RESOURCE element. In this case, if the RESOURCE contains a binary data format (anything other than a text-based format), the data must be encoded as base64, as defined in [2].

Validation Rule:

— If the REF attribute is specified, then the RESOURCE cannot contain character data, and vice versa.

<table>
<tr><td>Diagram</td><td colspan="3">  </td></tr>
<tr><td>Used by</td><td colspan="3"><Component></td></tr>
<tr><td rowspan="4">Attributes</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>mimeType</td><td>string</td><td>A MIME media type value indicating the type of the resource (as defined in RFC 2045).</td></tr>
<tr><td>ref</td><td>anyURI</td><td>The URI value that identifies the resource. If the ref attribute is omitted, then the element must contain the resource inline as character data.</td></tr>
<tr><td>localPath</td><td>anyURI</td><td>Specifies the required relative location for a cached version of the resource.</td></tr>
<tr><td>Source</td><td colspan="3"><pre>
<xsd:element name="Resource">
    <xsd:complexType mixed="true">
        <xsd:attribute name = "mimeType" use = "required"
            type = "xsd:string"/>
        <xsd:attribute name="ref" type="xsd:anyURI"/>
        <xsd:attribute name="localPath" type="xsd:anyURI"/>
    </xsd:complexType>
    <!-- "mixed" content model allows for embedded resources -->
</xsd:element>
</pre></td></tr>
</table>

Table 7 — RESOURCE element syntax

Example (informative):

COMPONENTS and RESOURCES are added to the example document from 7.2.7 to illustrate how they might be used. Each individual *item*, represented by each inner ITEM, contains a *component* that comprises a single photographic *resource*. Thus, this document could be used to represent a shelf (the CONTAINER) containing a photo album (the outermost ITEM) made up of two individual photographs (the two inner ITEMS).

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
  <Container>
    <Item>
      <Item>
        <Component>
          <Resource ref="myFirstPicture.jpg" mimeType="image/jpg" />
        </Component>
      </Item>
      <Item>
        <Component>
          <Resource ref="mySecondPic.bmp" mimeType="image/bmp" />
        </Component>
      </Item>
    </Item>
  </Container>
</DIDL>
```

7.2.10 <Descriptor>

A DESCRIPTOR represents a *Descriptor*. As such, it associates information with its parent element. This information may be contained in a COMPONENT element, or a STATEMENT element.

Typically, a DESCRIPTOR is used to associate descriptive data with a parent element. Descriptive data can take the form of a *component* or a *statement*. An example of a *component* containing descriptive data is that of a thumbnail version of a photographic image. An example of a *statement* containing descriptive data is that of a simple textual description, or meta-data, such as the title and author of a work.

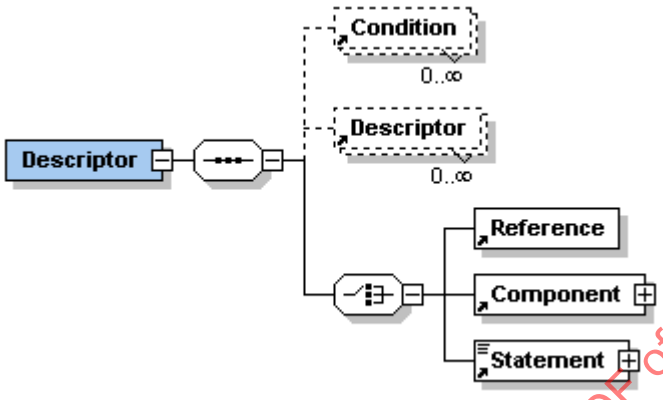
<table>
<tr><td>Diagram</td><td> 
</td></tr>
<tr><td>Children</td><td><Condition> <Descriptor> <Reference> <Component> <Statement></td></tr>
<tr><td>Used by</td><td><Anchor> <Annotation> <Choice> <Component> <Container> <Declarations> <Descriptor> <Item> <Selection></td></tr>
<tr><td>Attributes</td><td><table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>id</td><td>ID</td><td>A unique ID value.</td></tr>
</table></td></tr>
<tr><td>Source</td><td><pre>
<xsd:element name="Descriptor">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Condition" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element ref="Descriptor" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:choice>
                <xsd:element ref="Reference"/>
                <xsd:element ref="Component"/>
                <xsd:element ref="Statement"/>
            </xsd:choice>
        </xsd:sequence>
        <xsd:attributeGroup ref="ID_ATTRS"/>
    </xsd:complexType>
</xsd:element>
</pre></td></tr>
</table>

Table 8 — DESCRIPTOR element syntax

7.2.11 <Statement>

A STATEMENT represents a *Statement*. As such, it defines a piece of information pertaining to the parent element. Examples of likely STATEMENTS include descriptive, control, revision tracking or identifying information.

A STATEMENT can contain any data format, including plain text and various machine-readable formats such as well-formed XML. The data type of the STATEMENT is identified by the MIMETYPE attribute, which is a MIME media type identifier, as defined in RFC 2045 (e.g. 'text/xml').

A STATEMENT containing plain text can be used to associate a human-readable description with the parent element. A STATEMENT containing data in some machine-readable format (such as well-formed XML) can be used to express application-specific metadata in an application-specific form. The machine-readable type of STATEMENT is intended to preserve associations of application-specific meta-data with various elements, without requiring that all applications handling a document be able to process the meta-data. For example, a STATEMENT that binds a proprietary type of meta-data with an Item may be understandable only by a particular product of a particular company, but still any product that handles DIDL documents will preserve the association.

To ensure interoperability among different applications, further specification beyond the scope of this part of ISO/IEC 21000 may be required, in particular for the interoperable use of plain text STATEMENTS.

Note (informative): Although many of the examples given in this document use plain text for STATEMENT contents, it should be understood that the STATEMENT is primarily intended to carry machine-interpretable formats (such as XML-based formats).

For STATEMENTS containing well-formed XML, the grammar (i.e. the schema) of the XML fragment contained within the STATEMENT is identified by the namespace of the fragment. For any STATEMENT containing a binary data format (anything other than a text-based format) the data must be encoded as base64, as defined in [2].

<table>
<tr><td>Diagram</td><td colspan="3"> 
</td></tr>
<tr><td>Used by</td><td colspan="3"><Descriptor></td></tr>
<tr><td rowspan="2">Attributes</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>mimeType</td><td>string</td><td>A MIME media type value indicating the type of the resource (as defined in RFC 2045).</td></tr>
<tr><td>Source</td><td colspan="3"><pre>
<xsd:element name="Statement">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:any namespace="##any" processContents="lax" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="mimeType" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
</pre></td></tr>
</table>

Table 9 — STATEMENT element syntax

Example (informative):

DESCRIPTORS with STATEMENTS are now added to the example from 7.2.9 that express metadata. Here, metadata is expressed in plain (human readable) text, with a custom XML schema, and in an XML-based standardized machine-interpretable format (MPEG-7 in this case). This shows how the STATEMENT can express metadata in any format, and how differently-formatted metadata can coexist. The first pair of DESCRIPTORS gives a description for the CONTAINER. Note that the first DESCRIPTOR has a child DESCRIPTOR that explicitly indicates its relationship with the parent CONTAINER. The second DESCRIPTOR pair gives a title for the photo album. The third and fourth DESCRIPTOR pairs give captions for each of the individual photos in the album using three different mechanisms.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
      xmlns:foo="http://www.bar.org/title-schema"
      xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
  <Container id="C1">
    <Descriptor id="D1">
        <Descriptor>
            <Statement mimeType="text/xml">
                <Relation xsi:type="mpeg7:RelationType" name="titleOf">
                    <mpeg7:Argument idref="D1"/>
                    <mpeg7:Argument idref="C1"/>
                </Relation>
            </Statement>
        </Descriptor>
      <Statement mimeType="text/plain">My Photo Albums</Statement>
    </Descriptor>
      <Descriptor>
        <Statement mimeType="text/xml">
          <mpeg7:Mpeg7>
            <mpeg7:DescriptionUnit xsi:type="CreationInformationType">
              <mpeg7:Creation>
                <mpeg7:Title>
                  My Photo Albums
                </mpeg7:Title>
              </mpeg7:Creation>
            </mpeg7:DescriptionUnit>
          </mpeg7:Mpeg7>
        </Statement>
      </Descriptor>
    <Item>
      <Descriptor>
        <Statement mimeType="text/xml">
          <foo:TITLE>Photo Album #1</foo:TITLE>
        </Statement>
      </Descriptor>
      <Descriptor>
        <Statement mimeType="text/xml">
          <mpeg7:Mpeg7>
            <mpeg7:DescriptionUnit xsi:type="CreationInformationType">
              <mpeg7:Creation>
                <mpeg7:Title>
                  Photo Album #1
                </mpeg7:Title>
              </mpeg7:Creation>
            </mpeg7:DescriptionUnit>
          </mpeg7:Mpeg7>
        </Statement>
      </Descriptor>
      <Item>
        <Descriptor>
          <Statement mimeType="text/plain">
```

```
            Johnny's first day at school
          </Statement>
        </Descriptor>
        <Descriptor>
          <Statement mimeType="text/xml">
            <foo:CAPTION>Johnny's first day at school</foo:CAPTION>
          </Statement>
        </Descriptor>
        <Descriptor>
          <Statement mimeType="text/xml">
            <mpeg7:Mpeg7>
              <mpeg7:DescriptionUnit xsi:type="CreationInformationType">
                <mpeg7:Creation>
                  <mpeg7:Abstract>
                    <mpeg7:FreeTextAnnotation>
                      Johnny's first day at school
                    </mpeg7:FreeTextAnnotation>
                  </mpeg7:Abstract>
                </mpeg7:Creation>
              </mpeg7:DescriptionUnit>
            </mpeg7:Mpeg7>
          </Statement>
        </Descriptor>
        <Component>
          <Resource ref="myFirstPicture.jpg" mimeType="image/jpg"/>
        </Component>
      </Item>
      <Item>
        <Descriptor>
          <Statement mimeType="text/plain">
            Jane's first day at school
          </Statement>
        </Descriptor>
        <Descriptor>
          <Statement mimeType="text/xml">
            <foo:CAPTION>Jane's first day at school</foo:CAPTION>
          </Statement>
        </Descriptor>
        <Descriptor>
          <Statement mimeType="text/xml">
            <mpeg7:Mpeg7>
              <mpeg7:DescriptionUnit xsi:type="CreationInformationType">
                <mpeg7:Creation>
                  <mpeg7:Abstract>
                    <mpeg7:FreeTextAnnotation>
                      Janes's first day at school
                    </mpeg7:FreeTextAnnotation>
                  </mpeg7:Abstract>
                </mpeg7:Creation>
              </mpeg7:DescriptionUnit>
            </mpeg7:Mpeg7>
          </Statement>
        </Descriptor>
        <Component>
          <Resource ref="mySecondPic.bmp" mimeType="image/bmp"/>
        </Component>
      </Item>
    </Item>
  </Container>
</DIDL>
```

7.2.12 <Anchor>

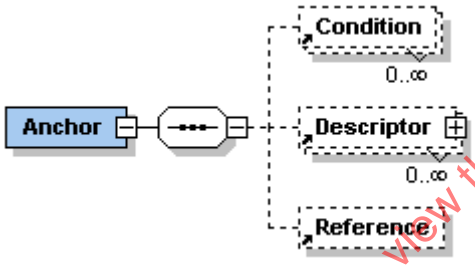
An ANCHOR element represents an *Anchor*. As such, it binds a set of DESCRIPTORS to a specific location or range within the *resource* identified by the RESOURCE element within the parent COMPONENT element. The *fragment* part of the *anchor* element in the Model is represented in an ANCHOR element by the FRAGMENT attribute. The FRAGMENT attribute is a string which, when appended to the *resource* URI plus a '#' character, specifies the desired location or part of interest within the associated *resource*.

The ID attribute, in addition to its normal usage as the target of a DIDL REFERENCE, can be used as a reference target by an external *resource*, such as the *resource* with which the ANCHOR is associated. This allows for a two-way linkage between an *anchor* and a *resource*. In practice, as with all attributes of type ID, it is recommended that the value of this ID attribute be made globally unique, so that it never needs to be changed. Since DIDL agents will not necessarily be able to edit any *resource*, this is the only way to guarantee that the two-way linkage is preserved.

Multiple ANCHORS within an ITEM may have the same PRECEDENCE attribute values. In such cases, the answer to the question which ANCHOR has the higher precedence (in other words, which ANCHOR is default for the ITEM) will be determined at the application-level.

Validation Rules:

— If the *resource* URI is a URL, a URL-compliant fragment identifier is required.

<table>
<tr><td>Diagram</td><td colspan="3"> 
</td></tr>
<tr><td>Children</td><td colspan="3"><Condition> <Descriptor> <Reference></td></tr>
<tr><td>Used by</td><td colspan="3"><Annotation> <Component> <Declarations></td></tr>
<tr><td rowspan="4">Attributes</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>precedence</td><td>unsignedInt</td><td>A unsigned integer value indicating the relative ranking of this Anchor among the other anchors in an Item. The Anchor with the highest precedence value is the default anchor for the Item. If this attribute is omitted, the precedence of the anchor is zero.</td></tr>
<tr><td>fragment</td><td>string</td><td>A string that, when appended to the value of the URI attribute of the associated Resource element, followed by a pound sign (“#”), locates the part of interest within the associated resource.</td></tr>
<tr><td>id</td><td>ID</td><td>A unique ID value.</td></tr>
<tr><td>Source</td><td colspan="3"><pre>
<xsd:element name="Anchor">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Condition" minOccurs="0"
        maxOccurs="unbounded"/>
      <xsd:element ref="Descriptor" minOccurs="0"
        maxOccurs="unbounded"/>
</pre></td></tr>
</table>

	<xsd:element ref="Reference" minOccurs="0" maxOccurs="1"/> </xsd:sequence> <xsd:attribute name="precedence" type="xsd:unsignedInt"/> <xsd:attribute name="fragment" type="xsd:string"/> <xsd:attributeGroup ref="ID_ATTRS"/> </xsd:complexType> </xsd:element>

Table 10 — ANCHOR element syntax

Example (informative):

This example is a single ITEM showing how multiple ANCHORS could be used on a single RESOURCE within a COMPONENT. The first *anchor* attaches a description "The whole session" to the beginning of the audio clip. The second *anchor* attaches a different description "Jim's killer drum solo" to a time-point of 17 minutes and 30 seconds past the beginning of the clip (using a fictitious media_time() function as a URI fragment). The assigned precedence values tell a user interface to display the first *anchor* before the second one when displaying the list of *anchor*s, and to make the first *anchor* the default.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
  <Item>
    <Component>
        <Resource ref="JimsGarageBand.mp3" mimeType="audio/mp3" />
        <Anchor precedence="200">
          <Descriptor>
            <Statement mimeType="text/plain">The whole session</Statement>
          </Descriptor>
        </Anchor>
        <Anchor precedence="100" fragment="media_time(17:30)">
          <Descriptor>
            <Statement mimeType="text/plain">
              Jim's killer drum solo
            </Statement>
          </Descriptor>
        </Anchor>
    </Component>
  </Item>
</DIDL>
```

7.2.13 <Choice>

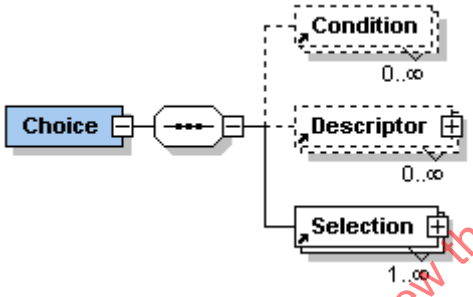
A CHOICE element represents a *Choice*. As such, it encapsulates a set of related SELECTIONS that can affect the configuration of an ITEM. The optional MINSELECTIONS and MAXSELECTIONS attributes specify the number of SELECTIONS that must be made for a CHOICE to be validly resolved. For example, if MINSELECTIONS and MAXSELECTIONS are omitted, then the CHOICE is multiple, meaning that any number of SELECTIONS may be made, including zero. If the MINSELECTIONS and MAXSELECTIONS attributes are both set to '1', then the CHOICE is single, meaning that exactly one SELECTION must be chosen.

Validation Rules:

— The value of the MAXSELECTIONS attribute must be no less than the value of the MINSELECTIONS attribute.

— The value of the MINSELECTIONS attribute must be no larger than the number of SELECTION children.

— The values specified in the DEFAULT attribute must each match the SELECT_ID value of one of the SELECTIONS within this CHOICE. The number of individual values in the DEFAULT attribute may not be less than the value of the MINSELECTIONS attribute, nor more than the value of the MAXSELECTIONS attribute.

<table>
<tr><td>Diagram</td><td> 
</td></tr>
<tr><td>Children</td><td><Condition> <Descriptor> <Selection></td></tr>
<tr><td>Used by</td><td><Item></td></tr>
<tr><td>Attributes</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>minSelections</td><td>nonNegativeInteger</td><td>Minimum number of selections that must be made. If not present, there is no minimum number.</td></tr>
<tr><td>maxSelections</td><td>PositiveInteger</td><td>Maximum number of selections that must be made. If not present, there is no maximum number.</td></tr>
<tr><td>default</td><td>IDREFS</td><td>A list of ID values defined in select_id attributes of selection elements, indicating the set of default selections for this choice.</td></tr>
<tr><td>choice_id</td><td>ID</td><td>Serves as the target for an Assertion element.</td></tr>
</table>
</td></tr>
<tr><td>Source</td><td><pre>
<xsd:element name="Choice">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Condition" minOccurs="0"
        maxOccurs="unbounded"/>
</pre></td></tr>
</table>

	<xsd:element ref="Descriptor" minOccurs="0" maxOccurs="unbounded"/> <xsd:element ref="Selection" maxOccurs="unbounded"/> </xsd:sequence> <xsd:attribute name="minSelections" type="xsd:nonNegativeInteger"/> <xsd:attribute name="maxSelections" type="xsd:positiveInteger"/> <xsd:attribute name="default" type="xsd:IDREFS"/> <xsd:attribute name="choice_id" type="xsd:ID"/> </xsd:complexType> </xsd:element>

Table 11 — CHOICE element syntax

7.2.14 <Selection>

A SELECTION element represents a *Selection*. As such, it defines a specific decision about a particular CHOICE. The SELECT_ID attribute value identifies the *Predicate* embodied by the SELECTION, and relates it to one or more CONDITIONS somewhere within an ITEM. At configuration time*, if the SELECTION is chosen, its *predicate* becomes true; if it is rejected, its *predicate* becomes false; if it is left unresolved, its *predicate* is left undecided.

Note that SELECTIONS and entire CHOICES may be made conditional (i.e. may have one or more CONDITION child elements). This makes it possible to implement complex decision trees in which certain selections may make certain subsequent CHOICES or SELECTIONS redundant. For example, a DIDL document might contain a CHOICE on whether to include a supplemental video clip, followed by a CHOICE on the encoding preference of the video clip. If, during configuration time, the video clip SELECTION is rejected, then the encoding preference CHOICE should be skipped.

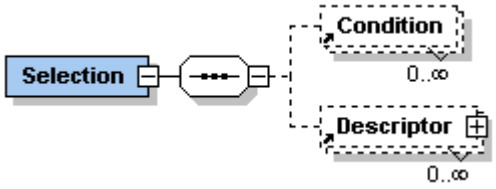
<table>
<tr><td>Diagram</td><td colspan="3"> 
</td></tr>
<tr><td>Children</td><td colspan="3"><Condition> <Descriptor></td></tr>
<tr><td>Used by</td><td colspan="3"><Choice></td></tr>
<tr><td rowspan="2">Attributes</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>select_id</td><td>ID</td><td>An ID value that can be referenced in a Condition element.</td></tr>
<tr><td>Source</td><td colspan="3"><pre>
<xsd:element name="Selection">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Condition" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element ref="Descriptor" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="select_id" type="xsd:ID" use="required"/>
    </xsd:complexType>
</xsd:element>
</pre></td></tr>
</table>

Table 12 — SELECTION element syntax

* The time when some software agent decides it is the appropriate time to make decisions on the SELECTIONS defined within a CHOICE

7.2.15 <Condition>

A CONDITION element represents a *Condition*. As such, it denotes the parent element as being conditional on a set of predicate tests. The REQUIRE attribute lists the set of *predicate*s that must become true, and the EXCEPT attribute lists the set of *predicate*s that must become false, in order for the CONDITION to be satisfied. Each *predicate* is identified by the value of the SELECT_ID attribute in a SELECTION element.

A set of CONDITION elements defines a Boolean combination of predicate tests. Multiple tests within a CONDITION are combined as a conjunction (an AND relationship). Multiple CONDITION elements within a given parent are combined as a disjunction (an OR relationship).

Validation Rules:

— Each ID value specified in the REQUIRE and EXCEPT attributes must match a SELECT_ID attribute value defined in a SELECTION element located somewhere within an ITEM element that is an ancestor of the CONDITION.

— Empty CONDITIONS are not permitted. Therefore, it is not valid for a CONDITION element to have neither a REQUIRE attribute nor an EXCEPT attribute.

Diagram			
Used by	<Anchor> <Choice> <Component> <Descriptor> <Item> <Selection>		
Attributes	**Name**	**Type**	**Description**
	require	IDREFS	A list of ID values matching SELECT_ID attribute of SELECTION element(s), indicating the SELECTION(s) that must be asserted for this CONDITION to evaluate to true.
	except	IDREFS	A list of ID values matching SELECT_ID attribute of SELECTION element(s), indicating the SELECTION(s) that must be asserted for this CONDITION to evaluate to false.
Source	<xsd:element name="Condition"> <xsd:complexType> <xsd:attribute name="require" type="xsd:IDREFS"/> <xsd:attribute name="except" type="xsd:IDREFS"/> </xsd:complexType> </xsd:element>		

Table 13 — CONDITION element syntax

Example (informative):

This example shows a simple CHOICE. The CHOICE specifies two SELECTIONS and specifies that exactly one SELECTION must be chosen. The DESCRIPTOR for the CHOICE describes the *choice* in a human-readable format, and the DESCRIPTORS for the SELECTIONS do likewise. If the user chooses the MP3 selection, then the MP3_FORMAT *predicate* becomes true and the WMA_FORMAT *predicate* becomes false, so the first COMPONENT is retained and the second one is discarded. Likewise, if the user chooses the WMA selection, then the WMA_FORMAT *predicate* becomes true and the MP3_FORMAT *predicate* becomes false, so the first is discarded and the second one is retained.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
  <Item>
```

```
    <Choice minSelections="1" maxSelections="1">
      <Descriptor>
        <Statement mimeType="text/plain">
          What format would you prefer?
        </Statement>
      </Descriptor>
      <Selection select_id="MP3_FORMAT">
        <Descriptor>
          <Statement mimeType="text/plain">I want MP3</Statement>
        </Descriptor>
      </Selection>
      <Selection select_id="WMA_FORMAT">
        <Descriptor>
          <Statement mimeType="text/plain">I want WMA</Statement>
        </Descriptor>
      </Selection>
    </Choice>
    <Component>
      <Condition require="MP3_FORMAT"/>
      <Resource ref="clip.mp3" mimeType="audio/mp3"/>
    </Component>
    <Component>
      <Condition require="WMA_FORMAT"/>
      <Resource ref="clip.wma" mimeType="audio/wma"/>
    </Component>
  </Item>
</DIDL>
```

7.2.16 <Reference>

A REFERENCE represents a reference to one of the following DIDL elements: CONTAINER, ITEM, COMPONENT, DESCRIPTOR, ANCHOR, or ANNOTATION. It can represent an "internal reference" (i.e., a reference to another element located somewhere within the parent document), or an "external reference" (i.e., a reference to an element in an external DIDL document).

Semantically, a reference links the contents of the referenced element (the element identified by the value of the REFERENCE'S TARGET attribute) to the existing contents of the referring element (the REFERENCE'S parent element). In addition, the values of any attributes not specified in the referring element are inherited from the referent element.

When used for internal referencing, the REFERENCE allows a document author to maintain a single source for an element that occurs in more than one place within a single DIDL document. For example, in a document representing a photo album, an author may want to include identical authorship information for each of the individual photos within the album.

The use of REFERENCE as an external reference allows lengthy or complex DIDL documents to be split up into multiple discrete documents. This can be used, for example, for load-balancing or other techniques to make more efficient use of computing/network resources. Note that it is possible for a referent element in an external document to contain internal references to elements declared in a local DECLARATIONS element within its parent document. In this situation, when linking the reference, the required element declarations in the external document must be copied to the DECLARATIONS element within the root DIDL element of the referring element.

It should be noted that, when internal or external references are processed by a DIDL processor that creates a new DID with actual copies of the referred elements, such processor must ensure the uniqueness of all ID type attribute values that are introduced by doing so. This should be done by checking all ID/IDREF values in the target of the reference. If necessary, some ID/IDREF values in the target should be renamed before including them in the new DID.

The TARGET attribute is a URI expression that identifies the element being referenced. Since it identifies a specific element, the TARGET attribute must contain a URI fragment identifier preceded by an optional base URI. If the base URI part is omitted, then the reference is understood to be an internal reference. The URI fragment part shall be formatted according to the "bare name" form of XPointer [5], or some other URI fragment identifier format. The "bare name" form of XPointer is defined as follows:

```
#{idref}
```

where `{idref}` is a valid IDREF for the document. An example of a complete TARGET URI is:

```
http://www.acme.com/document1.xml#REF_TARGET
```

Validation Rules:

— The TARGET attribute must evaluate to a single XML element node (since it is possible for full XPointer expressions to evaluate to a set of nodes).

— The fully qualified name of the referent element must match the fully qualified name of the referring element.

— The TARGET must evaluate to a single DIDL element node.

— The reference must be acyclic. In other words, the REFERENCE must not, either directly or indirectly reference its parent element (e.g. through a chain of references or by referencing an ancestor element).

Diagram	Reference		
Used by	<Anchor> <Component> <Container> <Descriptor> <Item> <Annotation>		
Attributes	**Name**	**Type**	**Description**
	target	anyURI	A URI expression identifying the referent element.
Source	<xsd:element name="Reference"> <xsd:complexType> <xsd:attribute name="target" type="xsd:anyURI" use="required"/> </xsd:complexType> </xsd:element>		

Table 14 — REFERENCE element syntax

Example (informative):

This example shows how REFERENCE elements are used to reference elements within the parent document (internal references). In this case, the referenced element (the target of the REFERENCE) is declared in a DECLARATIONS element. The declared element is a DESCRIPTOR, which is logically duplicated (via a REFERENCE) in each of the child ITEMS within the photo album ITEM.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
  <Declarations>
    <Descriptor id="PHOTO_INFO">
      <Statement mimeType="text/plain">
        Taken with my new SnazzyCam
      </Statement>
    </Descriptor>
  </Declarations>
  <Item>
    <Descriptor>
      <Statement mimeType="text/plain">Photo Album #1</Statement>
    </Descriptor>
    <Item>
      <Descriptor><Reference target="#PHOTO_INFO"/></Descriptor>
      <Component>
        <Resource ref="myFirstPicture.jpg" mimeType="image/jpg"/>
      </Component>
    </Item>
    <Item>
      <Descriptor><Reference target="#PHOTO_INFO"/></Descriptor>
      <Component>
        <Resource ref="mySecondPic.bmp" mimeType="image/bmp" />
      </Component>
    </Item>
  </Item>
</DIDL>
```

Example (informative):

This example shows how REFERENCE elements can be used as external references, to create composite DIDL documents from two or more discrete documents. Note that the DESCRIPTORS make contextual sense in album.xml, but they would probably not make sense in the context of photo99.xml or photo00.xml.

album.xml:

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
  <Container>
    <Item id=”x”>
      <Descriptor>
        <Statement mimeType="text/plain">Last year’s photo</Statement>
      </Descriptor>
      <Reference target="photo99.xml#thePhoto"/>
    </Item>
    <Item id=”y”>
      <Descriptor>
        <Statement mimeType="text/plain">This year’s photo</Statement>
      </Descriptor>
      <Reference target="photo00.xml#thePhoto"/>
    </Item>
  </Container>
</DIDL>
```

photo99.xml:

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
  <Item id="thePhoto">
    <Component>…</Component>
  </Item>
</DIDL>
```

photo00.xml:

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
  <Item id="thePhoto">
    <Component>…</Component>
  </Item>
</DIDL>
```

7.2.17 <Annotation>

An ANNOTATION element represents an *Annotation*. As such, it allows additional DESCRIPTORS and/or ANCHORS to be logically added to an element in one or more DIDL elements without affecting the original contents of the element. This allows, for example, an end-user to associate bookmarks and commentary to a digitally signed created work, without invalidating the signature. It also allows ASSERTIONS to be made on the *predicates* associated with the SELECTIONS of a given CHOICE.

Within ANNOTATIONS, a REFERENCE performs the same functionality as it does in other contexts of DIDL (such as ITEMS or DESCRIPTORS): a REFERENCE within an ANNOTATION is linked (at runtime) to the ANNOTATION itself before it is logically attached to the ANNOTATION TARGET.

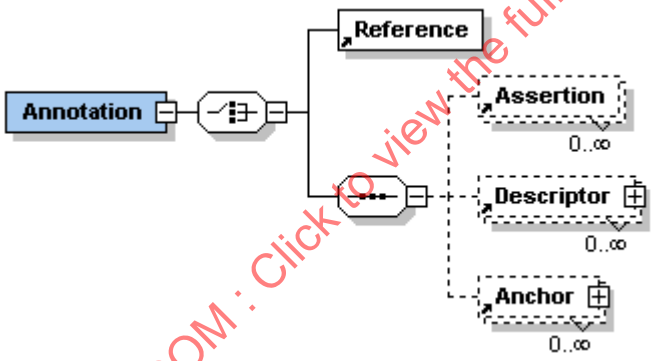
Validation Rules:

— The values given in the TARGET attribute must each correspond to some descendant element of the parent element, or that of the parent element itself.

— The contents of an ANNOTATION must conform to the content model of the targeted element(s). For example, ANCHORS can be included only if the TARGET attribute values each match the ID value of a COMPONENT.

— If an ANNOTATION contains an ASSERTION, then its TARGET attribute values must each match the ID attribute value of an ITEM.

— An ANNOTATION that does not contain a REFERENCE child element must have a TARGET attribute.

<table>
<tr><td>Diagram</td><td> 
</td></tr>
<tr><td>Children</td><td><Reference> <Assertion> <Descriptor> <Anchor></td></tr>
<tr><td>Used by</td><td><Item> <Declarations></td></tr>
<tr><td>Attributes</td><td colspan="2"><table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>target</td><td>anyURI</td><td>Identifies the elements being annotated.</td></tr>
<tr><td>id</td><td>ID</td><td>A unique ID value.</td></tr>
</table></td></tr>
<tr><td>Source</td><td><pre>
<xsd:element name="Annotation">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="Reference"/>
      <xsd:sequence>
        <xsd:element ref="Assertion" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element ref="Descriptor" minOccurs="0"
</pre></td></tr>
</table>

	maxOccurs="unbounded"/> <xsd:element ref="Anchor" minOccurs="0" maxOccurs="unbounded"/> </xsd:sequence> </xsd:choice> <xsd:attribute name="target" type="xsd:anyURI"/> <xsd:attributeGroup ref="ID_ATTRS"/> </xsd:complexType> </xsd:element>

Table 15 — ANNOTATION element syntax

Example (informative):

This example shows how an ANNOTATION element can be used to logically add elements to an Item without actually modifying its contents. In this case, the element being annotated is the first child ITEM of the outermost ITEM. At rendering time, the application software would logically add the DESCRIPTOR to the display of the ITEM.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
  <Item>
    <Item id="PHOTO_1">
      <Component>
        <Resource ref="myFirstPicture.jpg" mimeType="image/jpg" />
      </Component>
    </Item>
    <Item>
      <Component>
        <Resource ref="mySecondPic.bmp" mimeType="image/bmp" />
      </Component>
    </Item>
    <Annotation target="PHOTO_1">
      <Descriptor>
        <Statement mimeType="text/plain">This photo is really cool!</Statement>
      </Descriptor>
    </Annotation>
  </Item>
</DIDL>
```

7.2.18 <Assertion>

An ASSERTION element represents an *Assertion*. As such, it allows a set of *predicates* in a particular CHOICE to be asserted as true or false. This captures a particular state of a set of SELECTIONS within a CHOICE to be instantiated without modifying the original document. ASSERTIONS are always part of an ANNOTATION to an ITEM. The TARGET attribute of the ASSERTION identifies the affected CHOICE.

ASSERTIONS may either fully or partially resolve the given CHOICE. The *predicates* represented by the missing SELECT_ID values are left undecided. It is possible to continue resolving a partially resolved CHOICE by simply assigning true or false values to some or all of the undecided *predicates* to arrive at a new ASSERTION.

Validation Rules:

— The associated CHOICE element must be a descendant of the ITEM whose ID attribute value matches the parent ANNOTATION'S TARGET attribute value.

— The number of true *predicates* (i.e. the number of SELECT_ID values listed in the TRUE attribute) must be less than or equal to the MAXSELECTIONS attribute value in the associated CHOICE.

— The total number of SELECT_ID values defined in the associated CHOICE, minus the number of SELECT_ID values listed in the FALSE attribute, must be greater than or equal to the MINSELECTIONS attribute value in the associated CHOICE.

Diagram			
Used by	<Annotation>		
Attributes	**Name**	**Type**	**Description**
	target	anyURI	Identifies the CHOICE that this ASSERTION affects. Must contain an ID value that matches a CHOICE_ID attribute within the descendants of the parent of the ANNOTATION that contains this ASSERTION.
	true	NMTOKENS	The set of ID values corresponding to the SELECT_ID attributes of SELECTIONS that are to be asserted as true.
	false	NMTOKENS	The set of ID values corresponding to the SELECT_ID attributes of SELECTIONS that are to be asserted as false.
Source	<xsd:element name="Assertion"> <xsd:complexType> <xsd:attribute name="target" type="xsd:anyURI" use="required"/> <xsd:attribute name="true" type="xsd:NMTOKENS" use="optional"/> <xsd:attribute name="false" type="xsd:NMTOKENS" use="optional"/> </xsd:complexType> </xsd:element>		

Table 16 — ASSERTION element syntax

Example (informative):

This example shows an ASSERTION can be used to "save" a configuration within a document. In this case, the ASSERTION is targeting the format CHOICE, and is asserting the MP3_FORMAT SELECTION. Since the CHOICE specifies a MAXSELECTIONS value of 1, the application software can completely resolve the CHOICE with the given ASSERTION.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
  <Item id="THE_Item">
    <Choice choice_id="FORMAT_Choice" minSelections="1" maxSelections="1">
      <Descriptor>
        <Statement mimeType="text/plain">What format would you prefer?</Statement>
      </Descriptor>
      <Selection select_id="MP3_FORMAT">
        <Descriptor>
          <Statement mimeType="text/plain">I want MP3</Statement>
        </Descriptor>
      </Selection>
      <Selection select_id="WMA_FORMAT">
        <Descriptor>
          <Statement mimeType="text/plain">I want WMA</Statement>
        </Descriptor>
      </Selection>
    </Choice>
    . . .
    <Annotation target="THE_Item">
      <Assertion target="FORMAT_Choice" true="MP3_FORMAT"/>
    </Annotation>
  </Item>
</DIDL>
```

8 The Digital Item Declaration XML Schema Definition

```
<?xml version="1.0"?>
<!--================================================================-->
<!--================================================================-->
<!--                                                                -->
<!--           Schema for DIDL XML Document Type                    -->
<!--                                                                -->
<!--================================================================-->
<!--================================================================-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS"
            targetNamespace="urn:mpeg:mpeg21:2002:01-DIDL-NS"
            version="0.01">
   <!--================================================================

    Basic Principles that apply to all element types:

    1) Any element with an attribute named 'id' of type ID may
       have a Reference child.

    2) Elements with an attribute of type ID *may not* have any
       attributes which are required.  This is because
       attributes should be inheritable from a Reference
       if they are not specified, which is not possible
       if they are required.

    3) If an element has a Reference child
       then only Descriptor (and possibly Condition) elements may
       precede it, and no elements may follow it.

   ================================================================-->
   <xsd:attributeGroup name="ID_ATTRS">
      <xsd:attribute name="id" type="xsd:ID" use="optional"/>
   </xsd:attributeGroup>

   <!--================================================================

    DIDL element may contain exactly one Container or Item.

   ================================================================-->
   <xsd:element name="DIDL">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="Declarations" minOccurs="0"/>
            <xsd:choice>
               <xsd:element ref="Container"/>
               <xsd:element ref="Item"/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <!--================================================================

    A Declarations element contains any number of Items, Descriptors,
    Components, and Anchors, in any order.
```

```
    ============================================================-->
    <xsd:element name="Declarations">
        <xsd:complexType>
            <xsd:choice maxOccurs="unbounded">
                <xsd:element ref="Item"/>
                <xsd:element ref="Descriptor"/>
                <xsd:element ref="Component"/>
                <xsd:element ref="Annotation"/>
                <xsd:element ref="Anchor"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>

    <!--============================================================

     Container element may contain any number of Container elements
     followed by any number of Items.

    ============================================================-->
    <xsd:element name="Container">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Descriptor" minOccurs="0"
                    maxOccurs="unbounded"/>
                <xsd:choice>
                    <xsd:element ref="Reference"/>
                    <xsd:sequence>
                        <xsd:element ref="Container" minOccurs="0"
                            maxOccurs="unbounded"/>
                        <xsd:element ref="Item" minOccurs="0"
                            maxOccurs="unbounded"/>
                    </xsd:sequence>
                </xsd:choice>
            </xsd:sequence>
            <xsd:attributeGroup ref="ID_ATTRS"/>
        </xsd:complexType>
    </xsd:element>

    <!--============================================================

     Item element contains any number Choice elements,
     followed by at least one Item or Component element.
     An Item can be conditional.

    ============================================================-->
    <xsd:element name="Item">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Condition" minOccurs="0"
                    maxOccurs="unbounded"/>
                <xsd:element ref="Descriptor" minOccurs="0"
                    maxOccurs="unbounded"/>
                <xsd:element ref="Choice" minOccurs="0"
                    maxOccurs="unbounded"/>
                <xsd:choice>
                    <xsd:element ref="Reference"/>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element ref="Item"/>
                        <xsd:element ref="Component"/>
                    </xsd:choice>
```

```
            </xsd:choice>
            <xsd:element ref="Annotation" minOccurs="0"
               maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attributeGroup ref="ID_ATTRS"/>
      </xsd:complexType>
   </xsd:element>

   <!--============================================================

    A Descriptor contains descriptive data about its parent
    element.

    The Descriptor can be resource-based (comprised of a single
    Component), or text-based (comprised of a single Statement).
    A Descriptor can be conditional.

   ============================================================-->
   <xsd:element name="Descriptor">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="Condition" minOccurs="0"
               maxOccurs="unbounded"/>
            <xsd:element ref="Descriptor" minOccurs="0"
               maxOccurs="unbounded"/>
            <xsd:choice>
               <xsd:element ref="Reference"/>
               <xsd:element ref="Component"/>
               <xsd:element ref="Statement"/>
            </xsd:choice>
         </xsd:sequence>
         <xsd:attributeGroup ref="ID_ATTRS"/>
      </xsd:complexType>
   </xsd:element>

   <!--============================================================

    A Statement contains textual descriptive data within a
    Descriptor.

    Attribs:
    MimeType  - A string identifying the type of metadata

   ============================================================-->
   <xsd:element name="Statement">
      <xsd:complexType mixed="true">
         <xsd:sequence>
            <xsd:any namespace="##any" processContents="lax" minOccurs="0"/>
         </xsd:sequence>
         <xsd:attribute name="mimeType" type="xsd:string"/>
      </xsd:complexType>
   </xsd:element>

   <!--============================================================

    Component element contains one or more Resource elements, followed by any
    number of Anchor elements.
    A Component can be conditional.

   ============================================================-->
```

```
   <xsd:element name="Component">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="Condition" minOccurs="0"
               maxOccurs="unbounded"/>
            <xsd:element ref="Descriptor" minOccurs="0"
               maxOccurs="unbounded"/>
            <xsd:choice>
               <xsd:element ref="Reference"/>
               <xsd:element ref="Resource" minOccurs="1"
                  maxOccurs="unbounded"/>
            </xsd:choice>
            <xsd:element ref="Anchor" minOccurs="0"
               maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attributeGroup ref="ID_ATTRS"/>
      </xsd:complexType>
   </xsd:element>

   <!--============================================================

    An Anchor element indicates a point of interest in the
    resource of the parent Component.
    An Anchor can be conditional.

    Attribs:
    precedence  - An unsigned integer value indicating the position
                  that this start point should occupy relative to the
                  other start points in this title.  The highest
                  precedence anchor is the default entry
                  point.

    fragment  - The fragment identifier that locates the start
                point position within the resource. This
                string, when appended to the SRC attribute
                of the parent, plus a '#', becomes the full
                URI for the start point.

   ============================================================-->
   <xsd:element name="Anchor">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="Condition" minOccurs="0"
               maxOccurs="unbounded"/>
            <xsd:element ref="Descriptor" minOccurs="0"
               maxOccurs="unbounded"/>
            <xsd:element ref="Reference" minOccurs="0" maxOccurs="1"/>
         </xsd:sequence>
         <xsd:attribute name="precedence" type="xsd:unsignedInt"/>
         <xsd:attribute name="fragment" type="xsd:string"/>
         <xsd:attributeGroup ref="ID_ATTRS"/>
      </xsd:complexType>
   </xsd:element>
   <!--============================================================

    Condition element contains no children.  It indicates a
    selection condition for the parent file.  Multiple Condition
    tags indicate an 'OR' relationship, in that only one Condition
    needs to be satisfied for the parent element to be retrieved,
    included, etc.  The individual IDs in the require attribute of
```

```
     a Condition tag have an 'AND' relationship in that selection
     of all of the IDs referenced are required to satisfy that
     Condition.

     Attribs:
     require   - Indicates the Selection(s) that must be affirmed
                 for this Condition to be "satisfied".

     except    - Indicates the Selection(s) that must be denied
                 for this Condition to be "satisfied".

    ============================================================-->
    <xsd:element name="Condition">
       <xsd:complexType>
          <xsd:attribute name="require" type="xsd:IDREFS"/>
          <xsd:attribute name="except" type="xsd:IDREFS"/>
       </xsd:complexType>
    </xsd:element>

    <!--============================================================

     Choice element contains one or more Selections.
     A Choice can be conditional.

     Attribs:

     MinSelections   - Minimum number of Selections that must be made. If
                       not present, there is no minimum.

     MaxSelections   - Maximum number of Selections that can be made. If
                       not present, there is no maximum.

     default   - Indicates one or more default selections to use in the absence
                 of info to make a more specific decision.  Must conform to the
                 requirements of the minSelections and/or maxSelections
                 attributes if present.

     choice_id   - Serves as a target for Assertion elements.

    ============================================================-->
    <xsd:element name="Choice">
       <xsd:complexType>
          <xsd:sequence>
             <xsd:element ref="Condition" minOccurs="0"
                 maxOccurs="unbounded"/>
            <xsd:element ref="Descriptor" minOccurs="0"
                 maxOccurs="unbounded"/>
             <xsd:element ref="Selection" maxOccurs="unbounded"/>
          </xsd:sequence>
          <xsd:attribute name="minSelections" type="xsd:nonNegativeInteger"/>
          <xsd:attribute name="maxSelections" type="xsd:positiveInteger"/>
          <xsd:attribute name="default" type="xsd:IDREFS"/>
          <xsd:attribute name="choice_id" type="xsd:ID"/>
       </xsd:complexType>
    </xsd:element>

    <!--============================================================

     Selection element contains no children.
     A Selection can be conditional.
```

```
 Attrib: select_id - Uniquely identifies the Selection

============================================================-->

<xsd:element name="Selection">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="Condition" minOccurs="0"
            maxOccurs="unbounded"/>
         <xsd:element ref="Descriptor" minOccurs="0"
            maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="select_id" type="xsd:ID" use="required"/>
   </xsd:complexType>
</xsd:element>

<!--============================================================

 Resource element contains or points to binary data. The
 contained data can be binary or any valid XML element.

 Attribs:
 mimeType  - An identifier of a recognized scheme indicating the type
             of the resource.
 ref  - A URI from which the resource data can be obtained
 localPath - Specifies the required relative location for a cached version

============================================================-->
<xsd:element name="Resource">
   <xsd:complexType mixed="true">
      <xsd:attribute name = "mimeType" use = "required"
         type = "xsd:string"/>
      <xsd:attribute name="ref" type="xsd:anyURI"/>
      <xsd:attribute name="localPath" type="xsd:anyURI"/>
   </xsd:complexType>
   <!-- "mixed" content model allows for embedded resources -->
</xsd:element>

<!--============================================================

 Reference contains no child elements

 Attrib: target - Points to the referenced element

============================================================-->
<xsd:element name="Reference">
   <xsd:complexType>
      <xsd:attribute name="target" type="xsd:anyURI" use="required"/>
   </xsd:complexType>
</xsd:element>
<!--============================================================

 Annotation contains any number of Assertions followed by
 any number of Descriptors followed by any number of Anchors

 Attrib: target - Points to the element being annotated

 Restrictions:
   1. The target must reference an element within the
```

```
            parent Item, or can reference the parent Item itself.
         2. The contents of an Annotation must conform with the
            content model of the targeted element. For example,
            Anchors can be included only if the target references
            a Component.

   ============================================================-->
   <xsd:element name="Annotation">
      <xsd:complexType>
         <xsd:choice>
            <xsd:element ref="Reference"/>
            <xsd:sequence>
               <xsd:element ref="Assertion" minOccurs="0"
                  maxOccurs="unbounded"/>
               <xsd:element ref="Descriptor" minOccurs="0"
                  maxOccurs="unbounded"/>
               <xsd:element ref="Anchor" minOccurs="0"
                  maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:choice>
         <xsd:attribute name="target" type="xsd:anyURI"/>
         <xsd:attributeGroup ref="ID_ATTRS"/>
      </xsd:complexType>
   </xsd:element>

   <!--============================================================

    Assertion contains no child elements

    Attribs:

       true   - The set of selection IDs which are asserted as true
       false  - The set of selection IDs which are asserted as false

   ============================================================-->
   <xsd:element name="Assertion">
      <xsd:complexType>
         <xsd:attribute name="target" type="xsd:anyURI" use="required"/>
         <xsd:attribute name="true" type="xsd:NMTOKENS" use="optional"/>
         <xsd:attribute name="false" type="xsd:NMTOKENS" use="optional"/>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

9 Example Digital Items expressed in DIDL (informative)

9.1 Example 1: Using MPEG-7 descriptors in conjunction with a Choice

This example shows how MPEG-7 descriptors can be used to express the information required for a user agent to resolve a CHOICE.

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001">
    <Declarations>
        <!-- Define a Descriptor that will be reused in multiple places in this Item.
             In this case, it is the invariant parts of the media profile of the content. -->
        <Descriptor id="COMMON_PROFILE_DESC">
            <Statement mimeType="text/xml">
                <mpeg7:Mpeg7>
                    <mpeg7:DescriptionUnit xsi:type="mpeg7:MediaProfileType">
                        <mpeg7:MediaFormat>
                            <mpeg7:Content>audiovisual</mpeg7:Content>
                            <mpeg7:Medium href="urn:mpeg:mpeg7:cs:MPEG7MediumCS:1.1"/>
                            <mpeg7:FileFormat href="urn:mpeg:mpeg7:cs:MPEG7FileFormatCS:3"/>
                            <mpeg7:AudioCodingFormat
                              href="urn:mpeg:mpeg7:cs:MPEG7AudioCodingFormatCS:4.3.1"/>
                        </mpeg7:MediaFormat>
                    </mpeg7:DescriptionUnit>
                </mpeg7:Mpeg7>
            </Statement>
        </Descriptor>
    </Declarations>
    <Item>
        <Choice minSelections="1" maxSelections="1">
            <!-- Each Selection contains the parts of the MediaFormat that vary between the
                 three alternative resources for the content. In this example, the client
                 machine will have the opportunity to balance the allocation of bandwidth on
                 its network connection with the quality level desired by the end user. -->
            <Selection select_id="HI_QUALITY">
                <Descriptor id="HI_QUALITY_DESC">
                    <Statement mimeType="text/xml">
                        <mpeg7:Mpeg7>
                            <mpeg7:DescriptionUnit xsi:type="MediaProfileType">
                                <mpeg7:MediaFormat>
                                    <mpeg7:VisualCodingFormat
                                      href="urn:mpeg:mpeg7:cs:MPEG7VisualCodingFormatCS:3.1.4"/>
                                    <mpeg7:BitRate>384000</mpeg7:BitRate>
                                </mpeg7:MediaFormat>
                                <mpeg7:MediaQuality>
                                    <mpeg7:QualityRating ratingType="objective">
                                        <mpeg7:RatingValue>89.4</mpeg7:RatingValue>
                                        <mpeg7:RatingMetric>
                                            <mpeg7:QualityRatingScheme
                                    href="urn:mpeg:mpeg7:cs:MPEG-7QualityRatingSchemeCS:2.3"/>
                                            <mpeg7:RatingStyle>higherBetter</mpeg7:RatingStyle>
                                        </mpeg7:RatingMetric>
                                    </mpeg7:QualityRating>
                                </mpeg7:MediaQuality>
                            </mpeg7:DescriptionUnit>
                        </mpeg7:Mpeg7>
                    </Statement>
                </Descriptor>
            </Selection>
            <Selection select_id="MED_QUALITY">
                <Descriptor id="MED_QUALITY_DESC">
                    <Statement mimeType="text/xml">
                        <mpeg7:Mpeg7>
                            <mpeg7:DescriptionUnit xsi:type="MediaProfileType">
```

```
                            <mpeg7:MediaFormat>
                                <mpeg7:VisualCodingFormat
                                  href="urn:mpeg:mpeg7:cs:MPEG7VisualCodingFormatCS:3.1.3"/>
                                <mpeg7:BitRate>128000</mpeg7:BitRate>
                            </mpeg7:MediaFormat>
                            <mpeg7:MediaQuality>
                                <mpeg7:QualityRating ratingType="objective">
                                    <mpeg7:RatingValue>65.3</mpeg7:RatingValue>
                                    <mpeg7:RatingMetric>
                                        <mpeg7:QualityRatingScheme
                                href="urn:mpeg:mpeg7:cs:MPEG-7QualityRatingSchemeCS:2.3"/>
                                        <mpeg7:RatingStyle>higherBetter</mpeg7:RatingStyle>
                                    </mpeg7:RatingMetric>
                                </mpeg7:QualityRating>
                            </mpeg7:MediaQuality>
                        </mpeg7:DescriptionUnit>
                    </mpeg7:Mpeg7>
                </Statement>
            </Descriptor>
        </Selection>
        <Selection select_id="LO_QUALITY">
            <Descriptor id="LO_QUALITY_DESC">
                <Statement mimeType="text/xml">
                    <mpeg7:Mpeg7>
                        <mpeg7:DescriptionUnit xsi:type="MediaProfileType">
                            <mpeg7:MediaFormat>
                                <mpeg7:VisualCodingFormat
                                  href="urn:mpeg:mpeg7:cs:MPEG7VisualCodingFormatCS:3.1.2"/>
                                <mpeg7:BitRate>64000</mpeg7:BitRate>
                            </mpeg7:MediaFormat>
                            <mpeg7:MediaQuality>
                                <mpeg7:QualityRating ratingType="objective">
                                    <mpeg7:RatingValue>35.6</mpeg7:RatingValue>
                                    <mpeg7:RatingMetric>
                                        <mpeg7:QualityRatingScheme
                                href="urn:mpeg:mpeg7:cs:MPEG-7QualityRatingSchemeCS:2.3"/>
                                        <mpeg7:RatingStyle>higherBetter</mpeg7:RatingStyle>
                                    </mpeg7:RatingMetric>
                                </mpeg7:QualityRating>
                            </mpeg7:MediaQuality>
                        </mpeg7:DescriptionUnit>
                    </mpeg7:Mpeg7>
                </Statement>
            </Descriptor>
        </Selection>
    </Choice>
    <!-- In each of the following Components, the relevant Descriptors are included by
         reference (the Reference element). -->
    <Component>
        <Condition require="HI_QUALITY"/>
        <Descriptor>
            <Reference target="#HI_QUALITY_DESC"/>
        </Descriptor>
        <Descriptor>
            <Reference target="#COMMON_PROFILE_DESC"/>
        </Descriptor>
        <Resource ref="rtsp://telemedia1:/v11.mp4" mimeType="video/mp4v-es"/>
    </Component>
    <Component>
        <Condition require="MED_QUALITY"/>
        <Descriptor>
            <Reference target="#MED_QUALITY_DESC"/>
        </Descriptor>
        <Descriptor>
            <Reference target="#COMMON_PROFILE_DESC"/>
        </Descriptor>
        <Resource ref="rtsp://telemedia1/v12.mp4" mimeType="video/mp4v-es"/>
    </Component>
```

```
        <Component>
            <Condition require="LO_QUALITY"/>
            <Descriptor>
                <Reference target="#LO_QUALITY_DESC"/>
            </Descriptor>
            <Descriptor>
                <Reference target="#COMMON_PROFILE_DESC"/>
            </Descriptor>
            <Resource ref="rtsp://telemedia1/v13.mp4" mimeType="video/mp4v-es"/>
        </Component>
    </Item>
</DIDL>
```

9.2 Example 2: Expressing the same set of metadata in different descriptor formats

This example shows how different descriptor formats can be used to express overlapping metadata sets. This allows a wider range of applications to access the metadata. If a particular application understands one format but not the other, it can simply ignore the other descriptor and still make full use of the Digital Item. This example shows MPEG-7 and RDF-based Dublin Core descriptors. Also note that the COMPONENT in this Digital Item has more than one RESOURCE element, indicating multiple equivalent mechanisms for retrieving the same resource data.

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS"
    xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
    xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/">
    <Item>
        <Descriptor>
            <Statement mimeType="text/xml">
                <mpeg7:Mpeg7>
                    <mpeg7:DescriptionUnit xsi:type="mpeg7:CreationInformationType">
                        <mpeg7:Creation>
                            <mpeg7:Title>When the Thistle Blooms</mpeg7:Title>
                            <mpeg7:Creator>
                                <mpeg7:Role href="urn:mpeg:mpeg7:cs:MPEG7RoleCS:PERFORMER"/>
                                <mpeg7:PersonGroup>
                                    <mpeg7:Name>Always Red</mpeg7:Name>
                                </mpeg7:PersonGroup>
                            </mpeg7:Creator>
                            <mpeg7:Creator>
                                <mpeg7:Role href="urn:mpeg:mpeg7:cs:MPEG7RoleCS:PUBLISHER"/>
                                <mpeg7:Organization>
                                    <mpeg7:Name>PDQ Records</mpeg7:Name>
                                </mpeg7:Organization>
                            </mpeg7:Creator>
                        </mpeg7:Creation>
                    </mpeg7:DescriptionUnit>
                </mpeg7:Mpeg7>
            </Statement>
        </Descriptor>
        <Descriptor>
            <Statement mimeType="text/xml">
                <RDF:Description>
                    <dc:title>When the Thistle Blooms</dc:title>
                    <dc:creator>Always Red</dc:creator>
                    <dc:publisher>PDQ Records</dc:publisher>
                </RDF:Description>
            </Statement>
        </Descriptor>
        <Component>
            <Resource ref="rtsp://telemedia1:/v11.mp4" mimeType="audio/mp4a-latm"/>
            <Resource ref="urn:doi:10.1000-1" mimeType="audio/mp4a-latm"/>
        </Component>
    </Item>
</DIDL>
```

9.3 Example 3: A digital music album

This example shows how a digital music album might be expressed in DIDL. It shows, among other things, how multiple instances of metadata of various formats can coexist within a single Digital Item, and how a single Digital Item can be made configurable in various ways.

```
<?xml version="1.0"?>
<!-- This is a Digital Item Declaration for the (fictitious) musical album "Always Red". -->
<DIDL   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:cr="http://www.music-ratings.org/Content-Ratings-Scheme"
        xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
        xmlns:profile="urn:mpeg:mpeg21:example: Profile-Specs"
        xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS">
    <Declarations>
        <Descriptor id="ALBUM_RATING">
            <Descriptor>
                <!-- A Descriptor always says something about its parent. In this
                 case, the Descriptor is describing the parent Descriptor. -->
                <Statement mimeType="text/plain">
                    Content ratings provided by Parents of Teens, Inc.
                </Statement>
            </Descriptor>
            <Statement mimeType="text/xml">
                <cr:rating>
                    <cr:violence>None</cr:violence>
                    <cr:explicit-language>None</cr:explicit-language>
                    <cr:sex>None</cr:sex>
                </cr:rating>
            </Statement>
        </Descriptor>
    </Declarations>
    <Container>
        <!-- This Container is acting as a delivery package for a particular
             consumer. The package contains information about the package, in
             this case, the recipient's name and the distributor's name. -->
        <Descriptor>
            <Statement mimeType="text/plain">
                This is a package for John Q. Consumer. This package was provided
                by Digital Music Unlimited.
            </Statement>
        </Descriptor>
        <Item>
            <!-- Define a set of descriptive information associated with the
                 outermost item - the album as a whole. -->
            <Descriptor>
                <!-- The following statement is an example of an application-specific
                     data instance that can be included, in this case, a format
                     specific to the Digital Music Unlimited service. This example
                     happens to be encoded in plain text, but it is possible to encode
                     such data in any format compatible with well-formed XML. -->
                <Statement mimeType="application/vnd.dmu.content-organizer-hints">
                    DMU 9876:: Item Type="Music Album";
                </Statement>
            </Descriptor>
            <Descriptor>
                <Reference target="#ALBUM_RATING"/>
            </Descriptor>
            <Descriptor id="PERFORMING_GROUP">
                <Statement mimeType="text/xml">
                    <rdf:RDF>
                        <rdf:Description>
                            <dc:creator>Once Blue</dc:creator>
                        </rdf:Description>
                    </rdf:RDF>
                </Statement>
            </Descriptor>
```

```
            <Descriptor>
                <Statement mimeType="text/xml">
                    <rdf:RDF>
                        <rdf:Description>
                            <dc:title>Always Red</dc:title>
                            <dc:creator>Jack Jake (vocals)</dc:creator>
                            <dc:creator>Jane Juno (vocals, tambourine, finger snaps)
                                </dc:creator>
                            <dc:contributor>Joe Jump (acoustic &amp;amp; electric guitars, piano)
                                </dc:contributor>
                            <dc:contributor>Jeff Jelly (acoustic bass)  </dc:contributor>
                            <dc:contributor>Jim Jinks (drums, marimba, shaker, carob pods)
                                </dc:contributor>
                            <dc:subject>Record Album: Always Red by Always Red </dc:subject>
                            <dc:publisher>Acme Records </dc:publisher>
                            <dc:identifier>CD-ID: a409c80c</dc:identifier>
                            <dc:source>ASIN: B002U0A</dc:source>
                            <dc:date>1995-10-24</dc:date>
                        </rdf:Description>
                    </rdf:RDF>
                </Statement>
            </Descriptor>
            <Descriptor id="RIGHTS">
                <Statement mimeType="text/xml">
                    <rdf:RDF>
                        <rdf:Description>
                            <dc:rights>
                                Copyright 1995, Acme Records, All Rights Reserved.
                                Unauthorized duplication is a violation of
                                applicable laws.
                            </dc:rights>
                        </rdf:Description>
                    </rdf:RDF>
                </Statement>
            </Descriptor>
            <Descriptor id="ICON_FILE">
                <Descriptor>
                    <Statement mimeType="application/vnd.dmu.content-organizer-hints">
                        DMU 9876:: Descriptor Type="Item Icon";
                    </Statement>
                </Descriptor>
                <Component>
                    <Resource ref="http://www.dmu.com/always_red/B002U0A.t.jpg"
                        mimeType="image/jpeg"/>
                </Component>
            </Descriptor>
            <!-- This is the outermost Item, which represents the musical album
                 as a whole. -->
            <Choice choice_id="PLATFORM_Choice" minSelections="1" maxSelections="1">
                <!-- This choice allows the item to be configured for a specific
                     target platform: Windows, Linux, or Mac. -->
                <Selection select_id="PLATFORM_WINDOWS">
                    <Descriptor>
                        <Statement mimeType="text/xml">
                            <profile:operating-system>Win32</profile:operating-system>
                        </Statement>
                    </Descriptor>
                </Selection>
                <Selection select_id="PLATFORM_LINUX">
                    <Descriptor>
                        <Statement mimeType="text/xml">
                            <profile:operating-system>Linux</profile:operating-system>
                        </Statement>
                    </Descriptor>
                </Selection>
                <Selection select_id="PLATFORM_MAC">
                    <Descriptor>
                        <Statement mimeType="text/xml">
```

```
                <profile:operating-system>MacOS</profile:operating-system>
            </Statement>
            </Descriptor>
        </Selection>
    </Choice>
    <Choice choice_id="ALL_SONGS" >
        <!-- This choice allows the user to decide whether to filter out
             some of the songs on the album. The plain-text statements
             contain text that can be used in GUI dialogs. -->
        <Selection select_id="PICK_SONGS">
            <Descriptor>
            <Statement mimeType="text/plain">
                    I want to choose among the individual songs.
                </Statement>
            </Descriptor>
        </Selection>
    </Choice>
    <Choice choice_id="SONG_PICKER" default="SONG1 SONG2 SONG3 SONG4">
        <!-- This choice presents the user with the list of songs to choose
             from. It is conditional upon the PICK_SONGS selection from the
             previous choice, so it will be processed only if the PICK_SONGS
             selection is made. -->
        <Condition require="PICK_SONGS"/>
        <Descriptor>
            <Statement mimeType="text/plain">
                Choose the songs you would like:
            </Statement>
        </Descriptor>
        <Selection select_id="SONG1">
            <Descriptor>
                <!-- This descriptor contains a reference to another
                     descriptor. The effect is that the contents of the
                     referenced descriptor (which defines the title of the
                     first song) are logically duplicated within this,
                     the referring descriptor. This allows the actual value
                     of the title to be kept in a single location within
                     the document. -->
                <Reference target="#SONG1_TITLE"/>
            </Descriptor>
        </Selection>
        <Selection select_id="SONG2">
            <Descriptor>
                <Reference target="#SONG2_TITLE"/>
            </Descriptor>
        </Selection>
        <Selection select_id="SONG3">
            <Descriptor>
                <Reference target="#SONG3_TITLE"/>
            </Descriptor>
        </Selection>
        <Selection select_id="SONG4">
            <Descriptor>
                <Reference target="#SONG4_TITLE"/>
            </Descriptor>
        </Selection>
    </Choice>
    <Choice choice_id="BITRATE_Choice" default="LOW_BITRATE" minSelections="1"
        maxSelections="1">
        <Descriptor>
            <Statement mimeType="text/plain">
                Please select the fidelity you would prefer.
            </Statement>
        </Descriptor>
        <Selection select_id="LOW_BITRATE">
            <Descriptor>
                <Statement mimeType="text/plain">
                    Low (128 Kbits/sec, $10.00).
                </Statement>
```