ContentDirectory:1 Service Template Version 1.01

For UPnP™ Version 1.0
Status: Standardized DCP
Date: June 25, 2002

This Standardized DCP has been adopted as a Standardized DCP by the Steering Committee of the UPnP™ Forum, pursuant to Section 2.1(c)(ii) of the UPnP™ Forum Membership Agreement. UPnP™ Forum Members have rights and licenses defined by Section 3 of the UPnP™ Forum Membership Agreement to use and reproduce the Standardized DCP in UPnP™ Compliant Devices. All such use is subject to all of the provisions of the UPnP™ Forum Membership Agreement.

THE UPNP™ FORUM TAKES NO POSITION AS TO WHETHER ANY INTELLECTUAL PROPERTY RIGHTS EXIST IN THE STANDARDIZED DCPS. THE STANDARDIZED DCPS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". THE UPNP™ FORUM MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE STANDARDIZED DCPS, INCLUDING BUT NOT LIMITED TO ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, OF REASONABLE CARE OR WORKMANLIKE EFFORT, OR RESULTS OR OF LACK OF NEGLIGENCE.

Authors	Company
Kirt Debique	Microsoft
Tatsuya Igarashi	Sony
Sho Kou	Sony
Jean Moonen	Philips
John Ritchie	Intel
Gerrie Schults	HP
Mark Walker	Intel

Contents

List of Tables

1. Overview and Scope

This service template is compliant with the UPnP Device Architecture version 1.0. It defines a service type referred to herein as Content Directory Service (CDS).

1.1. Introduction

Many devices within the home network contain various types of content that other devices would like to access (e.g. music, videos, still images, etc). As an example, a "Media Server" device might contain a significant portion of the homeowner's audio, video, and still-image library. In order for the homeowner to enjoy this content, the homeowner must be able to browse the objects stored on the Media Server, select a specific one, and cause it to be "played" on an appropriate rendering device (e.g. an audio player for music objects, a TV for video content, an Electronic Picture Frame for still-images, etc).

For maximum convenience, it is highly desirable to allow the homeowner to initiate these operations from a variety of UI devices. In most cases, these UI devices will either be a UI built into the rendering device, or it will be a stand-alone UI device such as a wireless PDA or tablet. In any case, it is unlikely that the homeowner will interact directly with the device containing the content (i.e. the homeowner won't have to walk over to the server device). In order to enable this capability, the service device needs to provide a uniform mechanism for UI devices to browse the content on the server and to obtain detailed information about individual content objects. This is the purpose of the Content Directory Service

The Content Directory Service additionally provides a lookup/storage service that allows clients (e.g. UI devices) to locate (and possibly store) individual objects (e.g. songs, movies, pictures, etc) that the (server) device is capable of providing. For example, this service can be used to enumerate a list of songs stored on an MP3 player, a list of still-images comprising various slide-shows, a list of movies stored in a DVD-Jukebox, a list of TV shows currently being broadcast (a.k.a an EPG), a list of songs stored in a CD-Jukebox, a list of programs stored on a PVR (Personal Video Recorder) device, etc. Nearly any type of content can be enumerated via this Content Directory service. For those devices that contain multiple types of content (e.g. MP3, MPEG2, JPEG, etc), a single instance of the Content Directory Service can be used to enumerate all objects, regardless of their type.

2. Service Modeling Definitions

2.1. Service Type

The following service type identifies a service that is compliant with this template:

urn:**schemas-upnp-org**:**service**:ContentDirectory:1

Content Directory Service (CDS) is used herein to refer to this service type.

2.2. References

This section lists the normative references used in this document and includes the tag inside square brackets that is used for each such reference:

[DEVICE] - UPnP Device Architecture, version 1.0.

[XML] - "Extensible Markup Language (XML) 1.0 (Second Edition)", T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, eds. W3C Recommendation, 6 October 2000. Available at: http://www.w3.org/TR/2000/REC-xml-20001006.

[EBNF] ISO/IEC 14977, *Information technology - Syntactic metalanguage - Extended BNF*, December 1996.

[DIDL] ISO/IEC CD 21000-2:2001**,** *Information Technology - Multimedia Framework - Part 2: Digital Item Declaration*, July 2001.

[RFC 2396] Tim Berners-Lee, et. al. *RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax.*. 1998. Available at: http://www.ietf.org/rfc/rfc2396.txt

2.3. Terms

Table 1: Terms

Term	Description
object	Any data entity that can be returned by a Content Directory Service from a browsing or searching action. The Content Directory Service defines a class system to represent the different types of objects that are 'managed' by the CDS. The base class, from which all other classes are derived, is named **object**. The class **object** cannot be instantiated.
property	A property represents a CDS or client-defined characteristic of an **object**. The Content Directory Service expresses properties in XML as either elements or attributes. When expressed as an element, the property is addressed via its property/element tag name (including namespace other than DIDL-Lite) e.g. **dc:title**, **upnp:class**. When expressed as an attribute, the property is addressed with its parent expression, the @ delimeter and its property name e.g. **res@size**, **res@protocolInfo**, **upnp:class@name**. One exception is if the property is expressed as an attribute of an element which is a top-level object tag (**item**, **container**, etc.) in which case it is simply addressed with the @ delimeter and its property name e.g. **@id**, **@parentID**, **@restricted, @childCount** etc. Properties declared in this specification come from one of three metadata namespaces: DIDL-Lite, Dublin Core (dc) or UPnP (upnp). Their data types and meanings are defined in Section 6.
class	A **class** is used to assign a type to an **object**, and identifies the minimum required and optional set of properties that must be present on that object. Classes are organized in a hierarchy with certain classes being derived from others as in a typical object oriented system. At the root of the class hierarchy is the **object** base class. Examples are object.item.audioItem.musicTrack and object.container.album.musicAlbum. See section 2.4 for a definition of the format of the class specification for an object.
item	**item** is a first-level class derived directly from **object**. An **item** most often represents a single piece of AV data, such as a CD track, a movie or an audio file. Items may be playable, meaning they have information that can be played on a rendering device. Any object which derives from the item class is expressed via the DIDL-Lite **item** structure.

Term	Description
container	**container** is a first-level class derived directly from **object**. A **container** represents a collection of objects. Containers can represent the physical organization of objects (storage containers) or logical collections. Logical collections can have formal definitions of their contents or they can be arbitrary collections. Containers can be either homogeneous, containing objects that are all of the same class, or heterogeneous, containing objects of mixed class. Containers can contain other containers. Any object derived from the container class is expressed via the DIDL-Lite **container** structure. A CDS is required to maintain a **ContainerUpdateID** for each of its containers. This value is maintained internally, does not appear in any XML expression of the container, and cannot be used in a search or sort criterion.
container modification	A **container** is considered modified when any of the following occurs: A property of the **container** is added, removed or changed in value. A direct child element, whether **object**-derived or ordinary element, is added to or removed from the **container**. A direct, non-**container**-derived, child object has one of its properties or child elements added, removed or changed. A direct **container**-derived child element has **one** of its properties or non-**object**-derived child elements added, removed or changed. *Note to implementors: since '**ContainerUpdateID'** is not a formal property of a container, a modification to a direct child container that affects that child's '**ContainerUpdateID'** does not propagate upward to the parent container.*
XML fragment	In this document, XML fragment refers to a string that represents one element from a valid XML document. Individual uses of 'XML fragment' do not always specify the exact XML context that would be required to validate the fragment. If a "qualifying name" is given, the name defines the root element tag of the fragment. For example, 'DIDL-Lite XML fragment' means a string of the form "<DIDL-Lite …>…</DIDL-Lite>". In addition, any AV-defined XML fragment is permitted to be fully compliant XML. Any extraneous headers/tags should be gracefully ignored by the code processing the fragment.
CDS	Content Directory Service
ContainerUpdateID	An unsigned integer associated with each container. The integer value is incremented each time the container is modified (see the entry in this 'Terms' table for the precise definition of 'container modification'). Upon reaching the value 2^{32}-1, the next update rolls the value back to zero. Initial value of **ContainerUpdateID** for any newly created container is unspecified, but recommended to be zero. Implementers should maintain the same value for each container's **ContainerUpdateID** through power cycles and any other disappearance/reappearance on the network.

2.3.1. Notation: Strings Embedded in Other Strings

Some string variables and arguments described in this document contain substrings that must be independently identifiable and extractable for other processing. This requires the definition of appropriate substring delimiters and an escaping mechanism so that these delimiters can also appear as ordinary characters in the string and/or its independent substrings. This document uses embedded strings in two contexts—Comma Separated Value (CSV) lists (see Section 2.5.1.1 below) and property values in search criteria strings. Escaping conventions use the backslash character, '\' (UTF-8 character code 0x5C), as follows:

a. Backslash (‘\’) is represented as ‘\\’ in both contexts.
b. Comma (‘,’) is
    1. represented as ‘\,’ in individual substring entries in CSV lists
    2. not escaped in search strings
c. Double quote (‘"’) is
    1. not escaped in CSV lists
    2. not escaped in search strings when it appears as the start or end delimiter of a property value
    3. represented as ‘\"’ in search strings when it appears as a character that is part of the property value

2.3.2. Notation: Extended Backus-Naur Form

Extended Backus-Naur Form is used in this document for a formal syntax description of certain constructs. The usage here is according to the reference [EBNF].

Typographic conventions for EBNF

Symbol names in `monospace` font are non-terminal symbols. Character sequences between `'single quotes'` are terminal strings and must appear literally in valid expressions. Character sequences between (*comment delimiters*) are English language definitions or supplementary explanations of their associated symbols. White space in the EBNF is used to separate elements of the EBNF, not to represent white space in search strings. White space usage in actual search strings is described explicitly in the EBNF. Finally, the EBNF uses the following four operators:

Operator	Semantics
::=	**definition** — the non-terminal symbol on the left is defined by one or more alternative sequences of terminals and/or non-terminals to its right.
|	**alternative separator** — separates sequences on the right that are independently allowed definitions for the non-terminal on the left.
*	**“null” repetition** — means the expression to its left may occur zero or more times
+	**“non-null” repetition** — means the expression to its left must occur at least once and may occur more times

2.4. Class Hierarchy

The ContentDirectory service exposes a class hierarchy which is used to type all objects that can be retrieved from it. The ContentDirectory service exposes a class hierarchy which is used to type all objects that can be retrived from it. Each class is named using a string of the form described in 2.4.1 below. Each class definition includes a list of properties. Some properties are required while others are optional. Some properties are ‘multi-valued’ for a class, meaning that, in an XML instance of the class, the property may occur more than once. A class that is derived from another class must include all the required properties of the base class. The definition of a subclass may make some optional properties of the base class required. Each property will be expressed in XML as either an XML Element or XML Attribute.

This section will describe three classes, **object**, **object.item** and **object.container** which make up the base hierarchy from which all other classes (UPnP- or vendor-defined) derive, see section 7.

2.4.1. Class name syntax

Class name syntax is formally described using EBNF as described in section 2.3.2.

```
className     ::= baseName  |  derivedName

baseName      ::= 'object'

derivedName   ::= ( baseName  |  derivedName )  '.'  shortName

shortName     ::= (* valid XML name, excluding the characters
                      '.' — UTF-8 code 0x2E and
                      ':' — UTF-8 code 0x3A *)
```

2.4.2. Base Properties

These properties are used by the base classes. Where a property is defined as having an "element" XML structure, all properties that are defined as attributes of that XML element are implicitly included (see section 6 for details on those properties). The base properties are defined as follows:

Table 2: Base properties

Property Name	XML Structure	Namespace	Property Type	Property Description
id	Element attribute	DIDL-Lite	String	An identifier for the object. The value of each object id property must be unique with respect to the Content Directory.
title	Element	Dublin Core	String	Name of the object
creator	Element	Dublin Core	String	Primary content creator or owner of the object
res	Element	DIDL-Lite	URI	Resource, typically a media file, associated with the object. Values must be properly escaped URIs as described in [RFC 2396].
class	Element	UPnP	String	Class of the object.
searchable	Element attribute	DIDL-Lite	Boolean	When *true*, the ability to perform a **Search()** action under a **container** is enabled, otherwise a **Search()** under that **container** will return no results. The default value of this attribute when it is absent on a container is **false**
searchClass	Element	UPnP	String	See section 6 for details
createClass	Element	UPnP	String	See section 6 for details
parentID	Element attribute	DIDL-Lite	String	id property of object's parent. The parentID of the Content Directory 'root' container must be set to the reserved value of "-1". No other parentID attribute of any other Content Directory object may take this value.

Property Name	XML Structure	Namespace	Property Type	Property Description
refID	Element attribute	DIDL-Lite	String	id property of the item being referred to.
restricted	Element attribute	DIDL-Lite	Boolean	When *true*, ability to modify a given object is confined to the Content Directory Service. Control point metadata write access is disabled.
writeStatus	Element	UPnP	String	When present, controls the modifiability of the resources of a given object. Ability of a Control Point to change **writeStatus** of a given resource(s) is implementation dependent. Allowed values are: *WRITABLE, PROTECTED, NOT_WRITABLE, UNKNOWN, MIXED.*
childCount	Element attribute	DIDL-Lite	Integer	Child count for the object. Applies to containers only.

2.4.3. Class 'object' (Base Class)

This is the root class of the entire content directory class hierarchy. It can not be instantiated, in the sense that no XML fragment returned by a **Browse()** or **Search()** action can be of class **object**. The **object** class defines properties that are common to both atomic media items, as well as logical collections of these items. The **object** class contains the following properties:

Table 3: Object properties

Property Name	Required	Multiple Values
id	yes	no
parentID	yes	no
title	yes	no
creator	no	no
res	no	yes
class	yes	no
restricted	yes	no
writeStatus	no	no

2.4.4. Class 'item' : 'object'

This is a derived class of **object** used to represent "atomic" content objects, i.e., object that don't contain other objects, for example, a music track on an audio CD. The XML expression of any instance of a class that is derived from **item** is the <**item**> tag. The **item** class identifies the properties specified on its base class **object**, as well as the following additional properties:

Table 4: Item properties

Property Name	Required	Multiple Values
refID	no	no

2.4.5. Class 'container' : 'object'

This is a derived class of **object** used to represent containers e.g. a music album. The XML expression of any instance of a class that is derived from **container** is the <**container**> tag. The **container** class identifies the properties specified on its base class **object**, as well as the following additional properties:

Table 5: Container properties

Property Name	Required	Multiple Values
childCount	no	no
createClass	no	yes
searchClass	no	yes
searchable	no	no

2.5. State Variables

Unlike most other service templates, the Content Directory Service is primarily 'action' based. The service's state variables exist primarily to support argument passing of the service's actions. Information is not exposed directly through explicit state variables. Rather, a client retrieves Content Directory Service information via the return parameters of the actions defined below. The majority of state variables defined below exist simply to enable the various actions of this service.

Reader Note: For first-time reader, it may be more helpful to read the action definitions before reading the state variable definitions.

2.5.1. Derived data types

This section defines some derived data types that are represented as UPnP string data types with special syntax.

2.5.1.1. Comma Separated Value (CSV) Lists

The UPnP ContentDirectory Service uses variables that represent lists, or one-dimensional arrays, of values. Examples include the lists of active resource transfer ids and tag/value pairs for UpdateObject. The UPnP Device Architecture, Version 1.0 [DEVICE], does not provide for either an array type or a list type, so a list type is defined here. Lists may either be homogeneous (all values are the same type) or heterogeneous (values of different types are allowed). Lists may also consist of repeated occurrences of homogeneous or heterogeneous subsequences, all of which have the same syntax and semantics (same number of values, same value types and in the same order). The data type of a homogeneous list is *string (CSV x)*, where *x* is the UPnP type of the individual values. The data type of a heterogeneous list is of the form *string (CSV x, y, z)*, where *x*, *y* and *z* are the UPnP types of individual values. If the number of values in the heterogeneous list is too large to show each type individually, that variable type is represented as

string (CSV heterogeneous), and the variable description includes additional information as to the expected sequence of values appearing in the list and their corresponding types. The data type of a repeated subsequence list is *string (CSV {x, y, z})*, where *x*, *y* and *z* are the types of the individual values in the subsequence and the subsequence may be repeated zero or more times.

- A list is represented as a UPnP string type.
- Commas separate values within a list.
- Integer values are represented in CSVs with the same syntax as the integer data type specified in [DEVICE] (i.e., optional leading sign, optional leading zeroes, numeric ASCII)
- Boolean values are represented in CSVs as either '0' for false or '1' for true. These values are a subset of the defined boolean data type values specified in [DEVICE]: '0', 'false', 'no', '1', 'true', 'yes'.
- Escaping conventions for the comma and backslash characters are defined in section 2.3.1.
- White space before, after, or interior to any numeric data type is not allowed.
- White space before, after, or interior to any other data type is part of the value.

Table 6: CSV Examples

Type refinement of string	Value	Comments
CSV string	**+artist,-date**	List of 2 property sort criteria.
CSV int	**1,-5,006,0,+7**	List of 5 integers.
CSV boolean	**0,1,1,0**	List of 4 booleans
CSV string	**Smith\, Fred,Jones\, Davey**	List of 2 names, "Smith, Fred" and "Jones, Davey"
CSV i4,string,ui2	**-29837, string with leading blanks,0**	Note that the second value is " string with leading blanks"
CSV i4	**3, 4**	Illegal CSV. White space is not allowed as part of an integer value.
CSV string	**„**	List of 3 empty string values
CSV heterogeneous	**Alice,Marketing,5,Susan,R&D,21,David,Finance,7**	List of unspecified number of people and associated attributes. Each person is described by 3 elements—a name *string*, a department *string* and years-of-service *ui2*.

Table 7: State variables

Variable Name	Req. or Opt. [1]	Data Type	Allowed Value	Default Value	Eng. Units
TransferIDs	O	string (CSV ui4)	see section 2.5.2	n/a	n/a
A_ARG_TYPE_ObjectID	R	string	see section 2.5.3	n/a	n/a
A_ARG_TYPE_Result	R	string	see section 2.5.4	n/a	n/a
A_ARG_TYPE_SearchCriteria	O	string	see section 2.5.5	n/a	n/a

Variable Name	Req. or Opt. [1]	Data Type	Allowed Value	Default Value	Eng. Units
A_ARG_TYPE_BrowseFlag	R	string	BrowseMetadata , BrowseDirectChildren	n/a	n/a
A_ARG_TYPE_Filter	R	string (CSV string)	see section 2.5.7	n/a	n/a
A_ARG_TYPE_SortCriteria	R	string (CSV string)	see section 2.5.8	n/a	n/a
A_ARG_TYPE_Index	R	ui4	see section 2.5.9	n/a	n/a
A_ARG_TYPE_Count	R	ui4	see section 2.5.10	n/a	n/a
A_ARG_TYPE_UpdateID	R	ui4	see section 2.5.11	n/a	n/a
A_ARG_Type_TransferID	O	ui4	see section 2.5.12	n/a	n/a
A_ARG_Type_TransferStatus	O	string	see section 2.5.13	n/a	n/a
A_ARG_Type_TransferLength	O	string	see section 2.5.14	n/a	n/a
A_ARG_Type_TransferTotal	O	string	see section 2.5.15	n/a	n/a
A_ARG_TYPE_TagValueList	O	string (CSV string)	see section 2.5.16	n/a	n/a
A_ARG_TYPE_URI	O	uri	see section 2.5.17	n/a	n/a
SearchCapabilities	R	string (CSV string)	see section 2.5.18	n/a	n/a
SortCapabilities	R	string (CSV string)	see section 2.5.19	n/a	n/a
SystemUpdateID	R	ui4	see section 2.5.20	n/a	n/a
ContainerUpdateIDs	O	string (CSV {string, ui4})	see section 2.5.21	“”	n/a

[1] R = Required, O = Optional, X = Non-standard.

2.5.2. TransferIDs

TransferIDs is a CSV list of type **A_ARG_TYPE_TransferID**. It is evented to notify clients when file transfers initiated by **ImportResource** or **ExportResource** started or finished. When a file transfer starts, its transfer id is added to the **TransferIDs** list. When the transfer ends, its id is removed from **TransferIDs**.

This state variable is used for eventing only.

2.5.3. A_ARG_TYPE_ObjectID

This variable is used in conjunction with those actions that include an **ObjectID** parameter. **ObjectID** parameters uniquely identify individual objects within the Content Directory Service.

2.5.4. A_ARG_TYPE_Result

This variable is used in conjunction with those actions that include a **Result** parameter. The structure of the result is a DIDL-Lite XML fragment:

- Optional XML header e.g. <?xml version="1.0" ?>
- **<DIDL-Lite>** is the root tag.
- **<container>** is the tag representing **container** objects.
- **<item>** is the tag representing **item** objects.
- Tags in the Dublin Core (dc) and UPnP (upnp) namespaces represent object metadata.

See the DIDL-Lite schema in Appendix A for more details on the structure. The available metadata tags for properties are described in section 2.3 on Service Base Classes (for the base properties), and in the Appendix (for AV working group defined extended properties).

Control points may validate the document fragment returned by prepending the appropriate XML headers on the start of the document fragment.

Note that since the DIDL-Lite format of **Result** is based on XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a **SOAP** response message. In addition, when a variable of type **A_ARG_TYPE_Result** is employed as a parameter in a comma-separated list, commas (‘,’) that appear within XML CDATA must be escaped (as \,), see section 2.3.1.

2.5.5. A_ARG_TYPE_SearchCriteria

A_ARG_TYPE_SearchCriteria is the related state variable for the **SearchCriteria** parameter used in search actions. The **SearchCriteria** parameter gives one or more search criteria to be used for querying the Content Directory.

2.5.5.1. SearchCriteria String Syntax

SearchCriteria string syntax is described here formally using EBNF as described in section 2.3.2.

```
searchCrit    ::= searchExp  |  asterisk

searchExp     ::= relExp  |
                  searchExp  wChar+  logOp  wChar+  searchExp  |
                  ‘(’  wChar*  searchExp  wChar*  ‘)’

logOp         ::= ‘and’  |  ‘or’

relExp        ::= property  wChar+  binOp  wChar+  quotedVal  |
                  property  wChar+  existsOp  wChar+  boolVal

binOp         ::= relOp  |  stringOp

relOp         ::= ‘=’  |  ‘!=’  |  ‘<’  |  ‘<=’  |  ‘>’  |  ‘>=’

stringOp      ::= ‘contains’  |  ‘doesNotContain’  |  ‘derivedfrom’

existsOp      ::= ‘exists’

boolVal       ::= ‘true’  |  ‘false’

quotedVal     ::= dQuote  escapedQuote  dQuote
```

```
wChar          ::= space  |  hTab  |  lineFeed  |
                   vTab  |  formFeed  |  return

property       ::=      (*  property name as defined in section 2.4 *)

escapedQuote   ::=      (* double-quote escaped string as defined in section 2.3.1 *)

hTab           ::=      (* UTF-8 code 0x09, horizontal tab character *)

lineFeed       ::=      (* UTF-8 code 0x0A, line feed character *)

vTab           ::=      (* UTF-8 code 0x0B, vertical tab character *)

formFeed       ::=      (* UTF-8 code 0x0C, form feed character *)

return         ::=      (* UTF-8 code 0x0D, carriage return character *)

space          ::= ' '  (* UTF-8 code 0x20, space character *)

dQuote         ::= '"'  (* UTF-8 code 0x22, double quote character *)

asterisk       ::= '*'  (* UTF-8 code 0x2A, asterisk character *)
```

2.5.5.2. SearchCriteria String Semantics and Examples

- **Operator precedence**

  Precedence, highest to lowest, is:

  ```
  dQuote
  ( )
  binOp, existsOp
  and
  or
  ```

  Examples:

  '*s1* and *s2* or *s3* or *s4* and *s5*' is equivalent to:
  ' ((*s1* and *s2*) or *s3*) or (*s4* and *s5*)'

  '*s1* and *s2* or (*s3* or *s4*) and *s5*' is equivalent to:
  ' (*s1* and *s2*) or ((*s3* or *s4*) and *s5*)'

- **Return all**. The special value `'*'` means "find everything", or "return all objects that exist beneath the selected starting container".

- **Property existence testing**. Property existence is queried for by using the `'exists'` operator. Strictly speaking, `'exists'` could be a unary operator. This searchCriteria syntax makes it a binary operator to simplify search string parsing—there are no unary operators. The string "actor exists true" is true for every object that has at least one occurrence of the actor property. It is false for any object that has no actor property. Similarly, the string "actor exists false" is false for every object that has at least one occurrence of the actor property. It is true for any object that has no actor property.

- **Property omission**. Any property value query (as distinct from an existence query) applied to an object that does not have that property, evaluates to false.

- **Class derivation testing**. Existence of objects whose class is derived from some base class specification is queried for by using the `'derivedfrom'` operator. For example

'upnp:class derivedfrom "object.item"' is true for all objects whose class is "object.item", or whose class name begins with "object.item.".

- **Numeric comparisons**. When the operator in a `relExp` is a `relOp`, and both the `escapedQuote` value and the actual property value are sequences of decimal digits or sequences of decimal digits preceded by either a '+' or '-' sign (i.e., integers), the comparison is done numerically. For all other combinations of operators and property values, the comparison is done by treating both values as strings, converting a numeric value to its string representation in decimal if necessary.
  Note: The CDS is not expected to recognize any kind of numeric data other than decimal integers, composed only of decimal digits with the optional leading sign.

- **String comparisons**. All operators when applied to strings use case-insensitive comparisons.

2.5.6. A_ARG_TYPE_BrowseFlag

This variable is used in conjunction with the browse actions. A **BrowseFlag** parameter specifies a browse option to be used for browsing the Content Directory. Valid values are:

- *BrowseMetadata* - this indicates that the properties of the object specified by the **ObjectID** parameter will be returned in the result.

- *BrowseDirectChildren* - this indicates that first level objects under the object specified by **ObjectID** parameter will be returned in the result, as well as the metadata of all objects specified.

2.5.7. A_ARG_TYPE_Filter

This variable is used in conjunction with those actions that include a **Filter** parameter. The comma-separated list of property specifiers (including namespaces) indicates which metadata properties are to be returned in the results from browsing or searching.

Both properties represented in CDS query results as XML elements, as well as properties represented as element attributes, may be included in the comma-separated list.

If the **Filter** parameter is equal to "*", all properties are returned.

As a rule, all required properties are returned, but no optional properties will be returned unless explicitly requested in the filter.

A CDS must always respond to **Search()** and **Browse()** requests with valid DIDL-Lite in the **Result** parameter. Individual properties specified in the comma-separated filter list that would result in an invalid DIDL-Lite **Result** are selectively ignored by the CDS.

By the same token, individual properties NOT specified in the comma-separated **Filter** list that are required for a valid DIDL-Lite **Result** are automatically included. For example, since **title** is a required property for both **item** and **container** objects, the <**dc:title**> element would automatically be included in all <**item**>s and <**container**>s in the **Result**.

2.5.8. A_ARG_TYPE_SortCriteria

This variable is used in conjunction with those actions that include a **SortCriteria** parameter. **A_ARG_TYPE_SortCriteria** is CSV list of *signed* property names, where *signed* means preceded by '+' or '-' sign. The '+' and '-' indicate the sort is in ascending or descending order, respectively, with regard to the value of its associated property. Properties appear in the list in order of descending sort priority. For example, a value of "*+upnp:artist,-dc:date,+dc:title*" would sort first on artist in ascending order, then within each artist by date in descending order (most recent first) and finally by title in ascending order. An empty string indicates no sorting requested.

Note that only properties available in **SortCapabilities** can be sorted on.

2.5.9. A_ARG_TYPE_Index

This variable is used in conjunction with those actions that include an Index parameter. Index parameters specify an offset into an arbitrary list of objects. A value of *0* represents the first object in the list.

2.5.10.A_ARG_TYPE_Count

This variable is used in conjunction with those actions that include a **Count** parameter. **Count** parameters specify an ordinal number of arbitrary objects.

2.5.11.A_ARG_TYPE_UpdateID

This variable is used in conjunction with any action that includes an **UpdateID** parameter.

A_ARG_TYPE_UpdateID is the related state variable for the return parameter **UpdateID** used in **Browse()** and **Search()** actions. The return value will either be the **SystemUpdateID** (sec. 2.5.20 below) or a **ContainerUpdateID** (see Terms table, sec. **Error! Reference source not found.**)

2.5.12. A_ARG_TYPE_TransferID

This variable is used in conjunction with those actions that include a **TransferID** parameter. **TransferID** parameters uniquely identify individual file transfers initiated by the **ImportResource()** or the **ExportResource()** action of the Content Directory Service. **TransferID** is a unique value assigned by the device.

2.5.13. A_ARG_TYPE_TransferStatus

This variable is used in conjunction with those actions that include a **TransferStatus** parameter. This variable may assume one of the enumerated values: *IN_PROGRESS*, *STOPPED*, *ERROR*, or *COMPLETED*, indicating the status of a file transfer.

2.5.14.A_ARG_TYPE_TransferLength

This variable is used in conjunction with those actions that include a **TransferLength** parameter. It has the string type of data representing a numerical value that may exceed 32 bits in size.

2.5.15. A_ARG_TYPE_TransferTotal

This variable is used in conjunction with those actions that include a **TransferTotal** parameter. It has the string type of data representing a numerical value that may exceed 32 bits in size.

2.5.16.A_ARG_TYPE_TagValueList

A_ARG_TYPE_TagValueList is a CSV list of pairs of XML fragments. Each fragment is either an empty placeholder or a well-formed XML element. Note that commas (‘,’) that appear within XML CDATA in the fragments must be escaped (as \,), see section 2.3.1.

2.5.17.A_ARG_TYPE_URI

This variable is used in conjunction with any action that includes a URI parameter. A_ARG_TYPE_URI variables utilized as IN or OUT parameters in CDS actions must be properly escaped URIs as described in [RFC 2396].

2.5.18. SearchCapabilities

SearchCapabilities is a CSV list of property names that can be used in search queries. An empty string indicates that the CDS does not support any kind of searching. A wildcard ('*') indicates that the device supports search queries using all tags present in the CDS.

2.5.19. SortCapabilities

SortCapabilities is a CSV list of tags that the CDS can use to sort **Search()** or **Browse()** results. An empty string indicates that the device does not support any kind of sorting. A wildcard ('*') indicates that the device supports sorting using all tags present in the Content Directory.

2.5.20. SystemUpdateID

This required variable changes whenever anything in the Content Directory changes. A change could be a new or removed object, or a change in the metadata of an object. This variable is evented and the event is moderated at a maximum rate of 0.5 Hz (once every 2 seconds). The actual value of **SystemUpdateID** is unspecified. Clients should only check for equality with previous values of **SystemUpdateID**.

Note that the (optional) **ContainerUpdateIDs** variable provides more information about the scope of the change, since it takes advantage of the **ContainerUpdateID** values maintained for each container.

2.5.21. ContainerUpdateIDs

This optional state variable is an unordered CSV list of ordered pairs. Each pair consists of a **ContainerID** and a **ContainerUpdateID**, in that order. There can be at most one occurrence in **ContainerUpdateIDs** of an ordered pair with any specific **ContainerID**. The initial value of **ContainerUpdateIDs** is the empty string.

Each time a container is modified (see 'container modification' in table 2.3), its **ContainerUpdateID** is incremented and the ordered pair of the **ContainerID** and **ContainerUpdateID** are concatenated to the list **ContainerUpdateIDs**. If the **ContainerID** already appears in **ContainerUpdateIDs**, the new ordered pair is *not* added to the list. Instead, the corresponding **ContainerUpdateID** that is already in **ContainerUpdateIDs** is replaced by the new **ContainerUpdateID** value. A subscribing control point only sees the last value of **ContainerUpdateID** prior to the event.

ContainerUpdateIDs is a moderated evented state variable and is *only* used for eventing. There is no action that returns the value of **ContainerUpdateIDs**.

ContainerUpdateIDs is not a history list of container changes. Its evented value will never show the same '**ContainerID**,**ContainerUpdateID**' pair value twice. The net effect for a ContentDirectory Service implementation is that the first time a new value of a changed value of a **ContainerUpdateID** is added to **ContainerUpdateIDs** *after* its value has been evented, the value of **ContainerUpdateIDs** is cleared (set to the empty string) before the newly changed **'ContainerID**,**ContainerUpdateID**' is added to the list.

Example: Table 1 shows a time-ordered sequence of activities on a ContentDirectory Service for container modifications.

Table 1 ContainerUpdateIDs Example

Activity on CDS	**ContainerID**	New value of **ContainerUpdateID**	New value of **ContainerUpdateIDs** (the open close double quote marks are to clearly mark the variable value—they are not part of the value)
Initialization	—	—	“”
container modified	musicAlbum15	53	“musicAlbum15,53”
container modified	photoAlbum28	427	“musicAlbum15,53,photoAlbum28,427”
container modified	musicAlbum15	54	“musicAlbum15,54,photoAlbum28,427”
container modified	musicAlbum11	12	“musicAlbum15,54,photoAlbum28,427, musicAlbum11,12”
ContainerUpdateIDs is evented	—	—	— *(value not changed)*
New control point signs up for events	—	—	— *(no change to value, the special event value unicast to new control point includes the full set of 3 pairs)*
container modified	musicAlbum01	97	“musicAlbum01,97”

If this optional variable is not supported, a control point may use the **Browse()** or **Search()** actions to query the **ContainerUpdateID** for an individual container. For indications of CDS-wide change, use the evented variable **SystemUpdateID**, or the action **GetSystemUpdateID()**.

2.6. Eventing and Moderation

Table 8: Event moderation

Variable Name	Evented	Moderated Event	Max Event Rate [1]	Logical Combination	Min Delta per Event [2]
TransferIDs	Yes	No	N/A	N/A	N/A
A_ARG_TYPE_ObjectID	No	No	N/A	N/A	N/A
A_ARG_TYPE_Result	No	No	N/A	N/A	N/A
A_ARG_TYPE_SearchCriteria	No	No	N/A	N/A	N/A
A_ARG_TYPE_SortCriteria	No	No	N/A	N/A	N/A

Variable Name	Evented	Moderated Event	Max Event Rate [1]	Logical Combination	Min Delta per Event [2]
A_ARG_TYPE_UpdateID	No	No	N/A	N/A	N/A
A_ARG_TYPE_BrowseFlag	No	No	N/A	N/A	N/A
A_ARG_TYPE_Filter	No	No	N/A	N/A	N/A
A_ARG_TYPE_Index	No	No	N/A	N/A	N/A
A_ARG_TYPE_Count	No	No	N/A	N/A	N/A
A_ARG_Type_TransferID	No	No	N/A	N/A	N/A
A_ARG_Type_TransferStatus	No	No	N/A	N/A	N/A
A_ARG_Type_TransferLength	No	No	N/A	N/A	N/A
A_ARG_Type_TransferTotal	No	No	N/A	N/A	N/A
A_ARG_TYPE_TagValueList	No	No	N/A	N/A	N/A
A_ARG_TYPE_URI	No	No	N/A	N/A	N/A
SearchCapabilities	No	No	N/A	N/A	N/A
SortCapabilities	No	No	N/A	N/A	N/A
SystemUpdateID	Yes	Yes	2	N/A	N/A
ContainerUpdateIDs	Yes	Yes	2	N/A	N/A

[1] Determined by N, where Rate = (Event)/(N seconds).
[2] (N) * (allowedValueRange Step).

2.7. Actions

The following tables and subsections define the various CDS actions.

Except where noted, if an invoked action returns an error, the state of the device will be unaffected.

Table 9: Actions

Name	Req. or Opt. [1]
GetSearchCapabilities	R
GetSortCapabilities	R
GetSystemUpdateID	R
Browse	R
Search	O
CreateObject	O
DestroyObject	O
UpdateObject	O
ImportResource	O
ExportResource	O
StopTransferResource	O
GetTransferProgress	O

Name	Req. or Opt. [1]
DeleteResource	O
CreateReference	O
Non-standard actions implemented by an UPnP vendor go here.	X

[1] R = Required, O = Optional, X = Non-standard.

2.7.1. GetSearchCapabilities

This action returns the searching capabilities that are supported by the device.

2.7.1.1. Arguments

Argument	Direction	Related State Variable
SearchCaps	OUT	SearchCapabilities

2.7.1.2. Effect on State

This action has no effect on the device's current state.

2.7.1.3. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.

2.7.2. GetSortCapabilities

Returns the CSV list of meta-data tags that can be used in sortCriteria

2.7.2.1. Arguments

Argument	Direction	Related State Variable
SortCaps	OUT	SortCapabilities

2.7.2.2. Effect on State

This action has no effect on the device's current state.

2.7.2.3. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.

Error Code	Error Description	Description
501	Action failed	See UPnP Device Architecture section on Control.

2.7.3. GetSystemUpdateID

This action returns the current value of state variable **SystemUpdateID**. It can be used by clients that want to 'poll' for any changes in the Content Directory (as opposed to subscribing to events).

2.7.3.1. Arguments

Argument	Direction	Related State Variable
Id	OUT	SystemUpdateID

2.7.3.2. Effect on State

This action has no effect on the device's current state.

2.7.3.3. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.

2.7.4. Browse

This action allows the caller to incrementally browse the *native* hierarchy of the Content Directory objects exposed by the Content Directory Service, including information listing the classes of objects available in any particular object container.

2.7.4.1. Arguments

Argument	Direction	Related State Variable
ObjectID	IN	A_ARG_TYPE_ObjectID
BrowseFlag	IN	A_ARG_TYPE_BrowseFlag
Filter	IN	A_ARG_TYPE_Filter
StartingIndex	IN	A_ARG_TYPE_Index
RequestedCount	IN	A_ARG_TYPE_Count
SortCriteria	IN	A_ARG_TYPE_SortCriteria
Result	OUT	A_ARG_TYPE_Result
NumberReturned	OUT	A_ARG_TYPE_Count
TotalMatches	OUT	A_ARG_TYPE_Count

Argument	Direction	Related State Variable
UpdateID	OUT	A_ARG_TYPE_UpdateID

2.7.4.2. Argument Descriptions

Argument	Type	Description
ObjectID	string	Object currently being browsed. An **ObjectID** value of zero corresponds to the root object of the Content Directory.
BrowseFlag	string	See section 2.5.6.
Filter	string	See section 2.5.7.
StartingIndex	ui4	Starting zero based offset to enumerate children under the container specified by **ObjectID**. Must be 0 if BrowseFlag is equal to BrowseMetadata.
RequestedCount	ui4	Requested number of entries under the object specified by **ObjectID**. **RequestedCount** =*0* indicates request all entries.
SortCriteria	string	See section 2.5.8.
Result	string	See section 2.5.4.
NumberReturned	ui4	Number of objects returned in this result. If **BrowseMetadata** is specified in the **BrowseFlags**, then **NumberReturned** = 1
TotalMatches	ui4	If *BrowseMetadata* is specified in the **BrowseFlags** then **TotalMatches** = 1, else if *BrowseDirectChildren* is specified in the **BrowseFlags** then **TotalMatches** = total number of objects in the container specified for the **Browse()** action (independent of the starting index specified by the **StartingIndex** argument). If the CDS cannot compute **TotalMatches** and **NumberReturned** is not equal to zero, then **TotalMatches** = 0. If the CDS cannot compute **TotalMatches** and **NumberReturned** is equal to zero, then the CDS should return an error code 720.
UpdateID	ui4	**ContainerUpdateID** (see Terms, sec. 2.5.21)of the container being described if a container is specified in **ObjectID**. If the control point has an **UpdateID** for the container that is not equal to the **UpdateID** last returned, then the control point should refresh all its state relative to that container. If the **ObjectID** is zero, then the **UpdateID** returned is **SystemUpdateID (see sec. 2.5.20).**

2.7.4.3. Effect on State

This action has no effect on the current state of the device.

2.7.4.4. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.

Error Code	Error Description	Description
501	Action failed	See UPnP Device Architecture section on Control.
701	No such object	The specified **ObjectID** is invalid.
709	Unsupported or invalid sort criteria	The sort criteria specified is not supported or is invalid
720	Cannot process the request	Cannot process the request.

2.7.5. Search

This action allows the caller to search the content directory for objects that match some search criteria. The search criteria are specified as a query string operating on properties with comparison and logical operators.

2.7.5.1. Arguments

Argument	Direction	Related State Variable
ContainerID	IN	A_ARG_TYPE_ObjectID
SearchCriteria	IN	A_ARG_TYPE_SearchCriteria
Filter	IN	A_ARG_TYPE_Filter
StartingIndex	IN	A_ARG_TYPE_Index
RequestedCount	IN	A_ARG_TYPE_Count
SortCriteria	IN	A_ARG_TYPE_SortCriteria
Result	OUT	A_ARG_TYPE_Result
NumberReturned	OUT	A_ARG_TYPE_Count
TotalMatches	OUT	A_ARG_TYPE_Count
UpdateID	OUT	A_ARG_TYPE_UpdateID

2.7.5.2. Argument Description

The **Filter**, **StartingIndex**, **RequestedCount**, **SortCriteria** input arguments are the same as the corresponding input parameters for the **Browse()** action. The **Result** and **UpdateID** output arguments are the same as the corresponding output parameters for the **Browse()** action.

Argument	Type	Description
ContainerID	string	Unique identifier of the container in which to begin searching. A **ContainerID** value of zero corresponds to the root object of the Content Directory.
NumberReturned	ui4	Number of Content Directory objects returned in the Result

Argument	Type	Description
TotalMatches	ui4	Total number of Content Directory objects that match the search criteria (specified by the **SearchCriteria** argument, and independent of the starting index specified by the **StartingIndex** argument) under the object specified by the **ContainerID** argument. If the CDS cannot compute **TotalMatches** and **NumberReturned** is not equal to zero, then **TotalMatches** = 0. If the CDS cannot compute **TotalMatches** and **NumberReturned** is equal to zero, then the CDS should return an error code 720.
SearchCriteria	string	See section 2.5.5.

2.7.5.3. Effect on State

This action has no effect on the current state of the device.

2.7.5.4. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
708	Unsupported or invalid search criteria	The search criteria specified is not supported or is invalid
709	Unsupported or invalid sort criteria	The sort criteria specified is not supported or is invalid
710	No such container	The specified **ContainerID** is invalid or identifies an object that is not a **container**.
720	Cannot process the request	Cannot process the request

2.7.6. CreateObject

This action creates a new object in the container identified by **ContainerID**. The new object is created with the required **id** attribute set to "" and the required **restricted** attribute set to *false*. The actual value of the **id** attribute is supplied by the Content Directory Service.

The other elements of the new object are initialized according to the specified child elements. In addition, the Content Directory Service may create additional elements, for example, to ensure consistency across the whole directory. The unique id assigned to the newly created object is returned in the output parameter **ObjectID**. The complete object description is returned in output parameter **Result** in DIDL-Lite form.

When the new object is required to have one or more child <**res**> elements, the <**res**> elements may be generated in one of two ways:

- **The control point specifies the value of the <res> element.** The <**res**> element must identify a pre-existing resource, for example, an Internet radio station. When <**res**> value is present, then the resource is available for streaming and no additional **ImportResource()** action is needed.

- **The control point does not specify the value of the <res> element**. In this case it is the responsibility of the Content Directory Service to create a new value for the **importUri** attribute of the <**res**> element value for the purpose of importing the resource at a later time. The **res** element returned to the control point (as part of the **Result** output parameter) will have *no* value yet (actually set to ""), and is therefore not yet accessible for control points. After the control point has completely imported the actual resource content, via **ImportResource()** or HTTP POST, the Content Directory Service will add a new <**res**> value, and the content is then accessible. The Content Directory may subsequently decide to remove the **importUri** attribute, or keep it for the purpose of updating resource contents in the future.

In both cases, if the control point knows the MIME-type of the resource being added, the **protocolInfo** attribute of the <**res**> element should be set to '*:*:MIME-type:*' (e.g., *:*:audio/m3u:*). Otherwise, it should be set to '*:*:*:*'. It is the responsibility of the Content Directory Service to fill in the appropriate values for the *protocol*, *network* and *additionalInfo* fields of the **protocolInfo** attribute (e.g., http:*:audio/m3u:*) when it knows them (typically after importing the resource), such that compatibility between server and renderer devices can be checked for this resource.

Three examples of <**res**> elements are shown below:

- object has a resource that can be accessed:

<res protocolInfo="rtsp:*:audio/m3u:*"/>rtsp://10.0.0.1/contentdir?id=10</res>

- object has no resource and can import a resource.

<res protocolInfo="*:*:audio:*" importUri="http://10.0.0.1/postdir?id=10"> </res>

- object has a resource and the resource can be replaced.

<res protocolInfo="rtsp:*:audio/m3u:*" importUri="http://10.0.0.1/postdir?id=10"/>

rtsp://10.0.0.1/contentdir?id=10

</res>

In CDS implementations that don't allow resources to be placed directly under container objects, attempting to create a container object with resource property may generate a 'bad metadata' error. The ability to place resources directly under containers is vendor dependent.

Items that are actually references to other, existing Content Directory items are generated with the **CreateReference()** action. Reference items cannot be generated using **CreateObject()**.

2.7.6.1. Arguments

Argument	Direction	Related State Variable
ContainerID	IN	A_ARG_TYPE_ObjectID
Elements	IN	A_ARG_TYPE_Result
ObjectID	OUT	A_ARG_TYPE_ObjectID
Result	OUT	A_ARG_TYPE_Result

2.7.6.2. Effect on State

Updates the **SystemUpdateID** and the **ContainerUpdateID** for this container. Also, if creation of this object causes other containers to change by the autoupdate process, their **ContainerUpdateID**'s are updated as well.

2.7.6.3. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
710	No such container	The container identified by **ContainerID** does not exist.
713	Restricted parent object	Create object failed because **restricted** attribute of parent container is set to *true*.
712	Bad metadata	Cannot create object in the specified container with the specified meta-data.

2.7.7. DestroyObject

This action destroys the specified object when permitted. If the object is a container, all of its child objects are also deleted, recursively. Each deleted object becomes invalid and all references to it are also deleted.

The Content Directory Service is allowed (but not required) to delete a resource when it detects, with absolute certainty, that there are no references to it left anywhere in the Content Directory Service after the **DestroyObject()** action.

2.7.7.1. Arguments

Argument	Direction	Related State Variable
ObjectID	IN	A_ARG_TYPE_ObjectID

2.7.7.2. Effect on State

Updates the **SystemUpdateID** and the **ContainerUpdateID** of the parent **container**. Also, if deletion of this object causes other containers to change by the autoupdate process, their **ContainerUpdateID**'s are updated as well.

2.7.7.3. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
701	No such object	The specified **ObjectID** is invalid.
711	Restricted object	Destroy object failed because **restricted** attribute is set to *true*.
713	Restricted parent object	Operation failed because **restricted** attribute of parent object is set to *true*.

Error Code	Error Description	Description
715	Resource access denied	Cannot access the local resource

2.7.8. UpdateObject

This action modifies, deletes or inserts object metadata. The object to be updated is specified by **ObjectID**. **CurrentTagValue** is a CSV list of XML fragments. Each fragment is either the complete, exact, current text of an existing metadata element of the object or an empty placeholder. **NewTagValue** is also a CSV list of XML fragments, each of which is the complete new text of a metadata element for the object or an empty placeholder. The two tag/value lists must have the same number of entries. Each entry in **CurrentTagValue** represents metadata to be modified. The corresponding entry in **NewTagValue** represents the new, replacement metadata for the element identified by **CurrentTagValue**.

Identification of existing metadata to be modified must be exact. The **CurrentTagValue** entry text must exactly match the text of the existing element. This is easily done by copying the text from a previously returned **Search()** or **Browse()** result, and it allows **UpdateObject()** to perform a simple string compare between a **CurrentTagValue** entry and the current information. The matched existing element is then replaced by the corresponding entry in **NewTagValue**. The tag in the **NewTagValue** entry must be the same tag as in the **CurrentTagValue** entry. Using **UpdateObject()** to replace one piece of metadata by a completely different kind of metadata is not allowed. This must be accomplished by first deleting the old metadata and then inserting the new meatadata.

If an entry in **CurrentTagValue** is an empty string, it matches the “null” element, and the net effect is to insert the corresponding **NewTagValue** entry as a new metadata element. Conversely, if an entry in **CurrentTagValue** is not empty but the corresponding **NewTagValue** entry is, the net effect is to remove the current metadata element. If changing or removing a property would result in an object that no longer satisfies the requirements as specified by the object’s class, the **UpdateObject()** action is illegal. Examples include:

- Removing a required property, unless the property appears multiple times and this single removal leaves the object with a valid set of required occurrences.
- Changing the value of the **date** property to a person’s name.
- Changing the object’s class.

The Content Directory Service is allowed (but not required) to delete a resource when it detects, with absolute certainty, that there are no reamaining references to the resource anywhere in the Content Directory Service after the **UpdateObject()** action.

In CDS implementations that don't allow resources to be placed directly under a **container**, attempting to add a resource property may generate a 'bad metadata' error. The ability to place resources directly under containers is vendor dependent.

When the two parameter lists contain two or more entries, the multiple update request is performed as an atomic operation. In other words, all modifications to the object will be made before any change is visible to an external observer. Either the entire request succeeds or the object’s metadata is not changed. A partial update will never occur.

Example:

```
<item>
        <dc:title>My Favorite Song</dc:title>
```

```
        <upnp:artist>Singer1</upnp:artist>
        <dc:publisher>Acme Records</dc:publisher>
        <dc:date> 1990-01-01 </dc:date>
</item>
```

Items that are actually references to other, existing Content Directory items are generated with the **CreateReference()** action. Reference items cannot be generated using **UpdateObject()**.

Table 10: Update examples

Operation	Current Tag Value	New Tag Value	Notes:
Change the title of the Song	<dc:title>My Favorite Song</dc:title>	<dc:title>My Second Favorite Song</dc:title>	
Delete the date property	<dc:date> 1990-01-01 </dc:date>	(Empty String)	
Insert a genre tag	(Empty String)	<upnp:genre> Swing </upnp:genre>	
Change the artist's name from Singer1 to Singer2	<upnp:artist>Singer1 </upnp:artist>	<upnp:artist >Singer2 </upnp:artist >	The entire top-level tag (i.e. <upnp:artist>) is included in both **CurrentTagValue** and **NewTagValue.**
Change the title, insert another genre, and delete the publisher property	<dc:title>My Favorite Song</dc:title>,, <dc:publisher>Acme Music</dc:publisher >	<dc:title>My Third Favorite Song</dc:title>, <upnp:genre> Jazz</upnp:genre>,	In **CurrentTagValue**, note the double-comma placeholder just after </dc:title>. In **NewTagValue**, note that the trailing comma at the end is a placeholder for the deleted <dc:publisher> tag.

2.7.8.1. Arguments

Argument	Direction	Related State Variable
ObjectID	IN	A_ARG_TYPE_ObjectID
CurrentTagValue	IN	A_ARG_TYPE_TagValueList
NewTagValue	IN	A_ARG_TYPE_TagValueList

2.7.8.2. Dependency on State

The specified object must already exist. When modifying or deleting a specific tag, that tag must already exist.

2.7.8.3. Effect on State

This action will change the state of the metadata of the specified object. It also updates the **SystemUpdateID** and the **ContainerUpdateID** for the parent **container**. Finally, if changing this object causes other containers to change by the autoupdate process, their **ContainerUpdateID**'s are updated as well.

2.7.8.4. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
701	No such object	The specified **ObjectID** is invalid.
702	Invalid currentTagValue	The tag/value pair(s) listed in **CurrentTagValue** do not match the current state of the CDS. The specified data is likely out of date.
703	Invalid newTagValue	The specified value for the **NewTagValue** parameter is invalid.
704	Required tag	Unable to delete a required tag. See Appendix.
705	Read only tag	Unable to update a read only tag. See Appendix.
706	Parameter Mismatch	The number of tag/value pairs (including empty placeholders) in **CurrentTagValue** and **NewTagValue** do not match.
711	Restricted object	Operation failed because **restricted** attribute of object is set to *true*.
712	Bad metadata	Operation fails because it would result in invalid or disallowed metadata in current object.
713	Restricted parent object	Operation failed because **restricted** attribute of parent object is set to *true*.

2.7.9. ImportResource

This action transfers a file from a remote source resource, specified by **SourceURI**, to a local destination resource, specified by **DestinationURI**, in the Content Directory Service. When the Content Directory Service identifies the destination resource in CDS, the action will return a unique **TransferID** in the response and start HTTP GET. A client can monitor the progress of file transfer by using **GetTransferProgress()**.

The **DestinationURI** should correspond to an **importURI** attribute of a <**res**> element present in the CDS. This <**res**> element might already have accessible content, or not. In the first case, **ImportResource()** actually updates the resource contents. In the second case, **ImportResource()** assigns the first content to a object. Note that it is up to the Content Directory Service to determine if a specified <**res**> element is actually allowed to be updated.

2.7.9.1. Arguments

Argument	Direction	Related State Variable
SourceURI	IN	A_ARG_TYPE_URI
DestinationURI	IN	A_ARG_TYPE_URI
TransferID	OUT	A_ARG_TYPE_TransferID

2.7.9.2. Effect on State

When the file transfer is started, the **TransferID** returned by **ImportResource()** is added into the status **TransferIDs**. When the file transfer is finished, **TransferID** is removed from the status **TransferIDs**.

2.7.9.3. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
713	Restricted parent object	Operation failed because the **restricted** attribute of parent object is set to *true*.
714	No such source resource	Cannot identify the specified source resource
715	Source resource access denied	Cannot access the specified source resource
716	Transfer busy	Another file transfer is not accepted
718	No such destination resource	Cannot identify the specified destination resource
719	Destination resource access denied	Cannot access the specified destination resource

2.7.10. ExportResource

This action transfers a file from a local source resource, specified by **SourceURI**, to a remote destination resource, specified by **DestinationURI**. When the CDS identifies the source resource, the action will return a unique **TransferID** in the response and start the HTTP POST. A client can monitor the progress of file transfer by using the **GetTransferProgress()** action.

2.7.10.1.Arguments

Argument	Direction	Related State Variable
SourceURI	IN	A_ARG_TYPE_URI
DestinationURI	IN	A_ARG_TYPE_URI
TransferID	OUT	A_ARG_TYPE_TransferID

2.7.10.2.Effect on State

When the file transfer is started, the **TransferID** returned by **ExportResource()** is added into the status **TransferIDs**. When the file transfer is finished, **TransferID** is removed from the status **TransferIDs**.

2.7.10.3.Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
714	No such source resource	Cannot identify the specified source resource
715	Source resource access denied	Cannot access the specified source resource
716	Transfer busy	Another file transfer is not accepted
718	No such destination resource	Cannot identify the specified destination resource
719	Destination resource access denied	Cannot access the specified destination resource

2.7.11. StopTransferResource

This action stops the file transfer initiated by the **ImportResource()** or **ExportResource()** action. The file transfer, identified by **TransferID**, is halted immediately.

2.7.11.1.Arguments

Argument	Direction	Related State Variable
TransferID	IN	A_ARG_TYPE_TransferID

2.7.11.2.Effect on State

When the file transfer is finished, **TransferID** is removed from the status **TransferIDs**.

2.7.11.3.Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
717	No such file transfer	The file transfer specified by **TransferID** does not exist

2.7.12. GetTransferProgress

This action is used to query the progress of the file transfer initiated by the **ImportResource()** or the **ExportResource()** action. Progress of the file transfer, specified by **TransferID**, will be returned in the

response. **TransferStatus** indicates the status of the file transfer, it can be either *IN_PROGRESS*, *STOPPED*, *ERROR*, or *COMPLETED*. **TransferLength** specifies the length in bytes that has been transferred. **TransferTotal** specifies the total length of file in bytes that should be transferred in this file transfer. If the CDS cannot determine the total length, **TransferTotal** is set to zero. If the file transfer is started, the status is changed to *IN_PROGRESS*. If the file transfer is finished, the status is changed to either *STOPPED*, *ERROR*, or *COMPLETED* depending on the result of the file transfer. CDS maintains the status of a file transfer for at least 30 seconds after the file transfer has finished, ensuring that a client can query the result of file transfer.

2.7.12.1.Arguments

Argument	Direction	Related State Variable
TransferID	IN	A_ARG_TYPE_TransferID
TransferStatus	OUT	A_ARG_TYPE_TransferStatus
TransferLength	OUT	A_ARG_TYPE_TransferLength
TransferTotal	OUT	A_ARG_TYPE_TransferTotal

2.7.12.2. Effect on State

2.7.12.3. None.

2.7.12.4. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
717	No such file transfer	The file transfer specified by **TransferID** does not exist

2.7.13.DeleteResource

This action uses the specified **ResourceURI** to locate and delete all of the corresponding <**res**> elements in the CDS. Whether or not the resource identified by **ResourceURI** is actually deleted is implementation dependent.

2.7.13.1. Arguments

Argument	Direction	Related State Variable
ResourceURI	IN	A_ARG_TYPE_URI

2.7.13.2. Effect on State

Updates **SystemUpdateID**. Also updates **ContainerUpdateId** for each container in which a <**res**> element is removed from an object.

2.7.13.3. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
713	Restricted parent object	Operation failed because the **restricted** attribute of the parent object is set to *true*.
714	No such resource	Cannot identify the specified resource
715	Resource access denied	Cannot access the specified resource

2.7.14. CreateReference

This action creates a reference to an existing **item**, specified by the **ObjectID** argument, in the parent container, specified by the **ContainerID** argument. Both the **ContainerID** and **ObjectID** must already exist in the CDS. A unique, new object id is assigned to the newly created reference item and returned in the **NewID** output parameter.

A reference **item** may appear with the same meta data as that contained in the original **item** when browsed or searched, or may it may contain additional metadata. Reference items are distinguished from non-reference items by the addition of the **refID** attribute and value in the <**item**> tag.

2.7.14.1. Arguments

Argument	Direction	Related State Variable
ContainerID	IN	A_ARG_TYPE_ObjectID
ObjectID	IN	A_ARG_TYPE_ObjectID
NewID	OUT	A_ARG_TYPE_ObjectID

2.7.14.2.Effect on State

A new reference object is added to the specified parent object.

2.7.14.3. Errors

Error Code	Error Description	Description
402	Invalid args	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
701	No such object	The specified **ObjectID** is invalid.
710	No such container	The specified **ContainerID** is invalid or identifies an object that is not a **container**.
713	Restricted parent object	Operation failed because the **restricted** attribute of parent object is set to *true*.

2.7.15. Non-Standard Actions Implemented by an UPnP Vendor

To facilitate certification, non-standard actions implemented by a UPnP vendor should be included in the device's service template. The UPnP Device Architecture lists naming requirements for non-standard actions (cf. section on Description).

2.7.16. Common Error Codes

The following table lists error codes common to actions for this service type. If a given action results in multiple errors, the most specific error should be returned.

Table 11: Common error codes

Error Code	Error Description	Description
401	Invalid Action	See UPnP Device Architecture section on Control.
402	Invalid args	See UPnP Device Architecture section on Control.
404	Invalid Var	See UPnP Device Architecture section on Control.
501	Action failed	See UPnP Device Architecture section on Control.
600-699	TBD	Common action errors. Defined by UPnP Forum Technical Committee.
701	No such object	The specified **ObjectID** is invalid.
702	Invalid CurrentTagValue	The tag/value pair(s) listed in **CurrentTagValue** do not match the current state of the CDS. The specified data is likely out of date.
703	Invalid NewTagValue	The specified value for the **NewTagValue** parameter is invalid.
704	Required tag	Unable to delete a required tag. See Appendix.
705	Read only tag	Unable to update a read only tag. See Appendix.
706	Parameter Mismatch	The number of tag/value pairs (including empty placeholders) in **CurrentTagValue** and **NewTagValue** do not match.
708	Unsupported or invalid search criteria	The search criteria specified is not supported or is invalid
709	Unsupported or invalid sort criteria	The sort criteria specified is not supported or is invalid
710	No such container	The specified **ContainerID** is invalid or identifies an object that is not a **container**.
711	Restricted object	Operation failed because the **restricted** attribute of object is set to *true*.
712	Bad metadata	Operation fails because it would result in invalid or disallowed metadata in current object.
713	Restricted parent object	Operation failed because the **restricted** attribute of parent object is set to *true*.
714	No such source resource	Cannot identify the specified source resource
715	Source resource access denied	Cannot access the specified source resource
716	Transfer busy	Another file transfer is not accepted

Error Code	Error Description	Description
717	No such file transfer	The file transfer specified by **TransferID** does not exist
718	No such destination resource	Cannot identify the specified destination resource
719	Destination resource access denied	Cannot access the specified destination resource
720	Cannot process the request	Cannot process the request
800-899	*TBD*	*(Specified by UPnP vendor.)*

2.8. Theory of Operation (Informative)

2.8.1. Introduction

This section walks through several scenarios to illustrate the various actions supported by the Content Directory. These include browsing, searching, object creation, update, and deletion, property creation, update and deletion, content transfer, playlist manipulation, Internet content representation, and asynchronous behavior.

2.8.2. Content setup for Browsing and Searching

The following illustrates the logical structure of a Content Directory Service which exposes a physical directory structure on a PC like file system. The content includes music and photos organized into a few directory folders. The logical directory hierarchy is as follows:

- Name=”Content”
  - Name=”My Music”
    - Name=”Singles Soundtrack - Various Artists.musicalbum”
      - Name=”Would - Alice In Chains.wma”, Size=”90000”
      - Name=”Chloe Dancer - Mother Love Bone.wma”, Size=”200000”
      - Name=”State Of Love And Trust - Pearl Jam.wma”, Size=”70000”
      - Name=”Drown - Smashing Pumpkins.mp3”, Size=”140000”
    - Name=”Brand New Day - Sting.musicalbum”
      - Name=”A Thousand Years - Sting.wma”, Size=”100000”
      - Name=”Desert Rose - Sting.wma”, Size=”50000”
      - Name=”Big Lie Small World - Sting.mp3”, Size=”80000”
  - Name=”My Photos”
    - Name=”Mexico Trip.photoalbum”

- Name=”Sunset on the beach - 10/20/2001.jpg”, Size=”20000”
- Name=”Playing in the pool - 10/25/2001.jpg”, Size=”25000”

▪ Name=”Christmas.photoalbum”

- Name=”John and Mary by the fire - 12/24/2001.jpg”, Size=”22000”
- Name=”Christmas Tree loaded with presents - 12/25/2001.jpg”, Size=”10000”

- Name=”Album Art”
  - Name=”Brand New Day.albumart”,Size=”20000”
  - Name=”Singles Soundtrack.albumart”,Size=”20000”

2.8.3. Browsing

The **Browse()** action is designed to allow the control point to navigate the “native” content hierarchy exposed by the CDS. This hierarchy could map onto an explicit physical hierarchy or a logical one. In addition, the **Browse()** action enables the following features while navigating the hierarchy:

- **Metadata only browsing**. The metadata associated with a particular object can be retrieved.
- **Children object browsing**. The direct children of an object derived from a container can be retrieved.
- **Incremental navigation** i.e. the full hierarchy is never returned in one call since this is likely to flood the resources available to the control point (memory, network bandwidth, etc.). Also within a particular hierarchy level, the control point can restrict the number (and the starting offset) of objects returned in the result.
- **Sorting**. The result can be requested in a particular sort order. The available sort orders are expressed in the return value of the **GetSortCapabilities()** action.
- **Filtering**. The result data can be filtered to only include a subset of the properties available on the object (see section 2.5.7). Note that certain properties may not be filtered out in order to maintain the validity of the result data fragment. If a non-filterable property is left out of the filter set, it will still be included in the result.

The following examples illustrate the typical **Browse()** request-response interaction between a control point and a CDS. It assumes the content setup specified in section 2.8.2

2.8.3.1. Retrieving sort capabilities

When it connects to the content directory, the control point determines which properties can be used as sort criteria in a **Browse()** or **Search()** request. It does this via the **GetSortCapabilities** action:

request: `GetSortCapabilities()`

response: `GetSortCapabilities(“dc:title,dc:creator,dc:date,res@size”)`

2.8.3.2. Browsing the root level metadata

The control point needs to retrieve the root level metadata for the content directory. It does this via the following Browse action:

```
request: Browse( “0”, “BrowseMetadata”, “*”, 0, 0, “” )


response: Browse(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
   <container id=”0” parentID=”-1” childCount=”2” restricted=”true”
searchable="true">
      <dc:title>My multimedia stuff</dc:title>
      <upnp:class>object.container.storageFolder</upnp:class>
      <upnp:storageUsed>907000</upnp:storageUsed>
      <upnp:writeStatus>WRITABLE</upnp:writeStatus>
      <upnp:searchClass includeDerived=”false” >
         object.container.album.musicAlbum
      </upnp:searchClass>
      <upnp:searchClass includeDerived=”false” >
         object.container.album.photoAlbum
      </upnp:searchClass>
      <upnp:searchClass includeDerived=”false” >
         object.item.audioItem.musicTrack
      </upnp:searchClass>
        <upnp:searchClass includeDerived=”false” >
           object.item.imageItem.photo
        </upnp:searchClass>
        <upnp:searchClass name=”Vendor Album Art” includeDerived=”true”>
                  object.item.imageItem.photo.vendorAlbumArt
        </upnp:searchClass>
    </container>
</DIDL-Lite>”, 1, 1, 10 )
```

Note that the response contains the DIDL-Lite fragment with the metadata corresponding to the root container of the CDS (**container id**=*0*), and the other out parameters **NumberReturned**, **TotalMatches**, and **ContainerUpdateID**, respectively.

2.8.3.3. Browsing the children of the root level

The control point needs to retrieve the children of the root-level container. The control point can display 3 items at a time, so it restricts the number of children returned in the result. It does this via the following **Browse()** action:

```
request: Browse( “0”, “BrowseDirectChildren”, “*”, 0, 3, “” )


response: Browse(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
<container id=”1” parentID=”0” childCount=”2” restricted=”false”>
           <dc:title>My Music</dc:title>
           <upnp:class>object.container.storageFolder</upnp:class>
           <upnp:storageUsed>730000</upnp:storageUsed>
           <upnp:writeStatus>WRITABLE</upnp:writeStatus>
```

```
            <upnp:searchClass includeDerived=”false”>
                  object.container.album.musicAlbum
            </upnp:searchClass>
            <upnp:searchClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:searchClass>
            <upnp:createClass includeDerived=”false”>
                  object.container.album.musicAlbum
            </upnp:createClass>
      </container>
      <container id=”2” parentID=”0” childCount=”2” restricted=”false”>
            <dc:title>My Photos</dc:title>
            <upnp:class>object.container.storageFolder</upnp:class>
            <upnp:storageUsed>177000</upnp:storageUsed>
            <upnp:writeStatus>WRITABLE</upnp:writeStatus>
            <upnp:searchClass includeDerived=”false”>
                  object.container.album.photoAlbum
            </upnp:searchClass>
            <upnp:searchClass includeDerived=”false”>
                  object.item.imageItem.photo
            </upnp:searchClass>
            <upnp:createClass includeDerived=”false”>
                  object.container.album.photoAlbum
            </upnp:createClass>
      </container>
      <container id=”30” parentID=”0” childCount=”2” restricted=”false”>
            <dc:title>Album Art</dc:title>
            <upnp:class>object.container.storageFolder</upnp:class>
            <upnp:storageUsed>40000</upnp:storageUsed>
            <upnp:writeStatus>WRITABLE</upnp:writeStatus>
            <upnp:searchClass name=”Vendor Album Art”
               includeDerived=”true”>
                  object.item.imageItem.photo.vendorAlbumArt
            <upnp:searchClass>
            <upnp:createClass includeDerived=”true”>
                  object.item.imageItem.photo.vendorAlbumArt
            </upnp:createClass>
      </container>
</DIDL-Lite>”, 2, 2, 10 )
```

2.8.3.4. Browsing the children of the My Music folder

The control point needs to retrieve the children of the *My Music* folder. The control point can display 3 items at a time, so it specifies the number of children returned in the result. In addition, it specifies the result to be sorted in ascending order by the creator property. It does this via the following **Browse()** action:

```
request: Browse( “1”, “BrowseDirectChildren”, “*”, 0, 3, “+dc:creator” )
```

```
response: Browse(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <container id=”4” parentID=”1” childCount=”3” restricted=”false”>
            <dc:title>Brand New Day</dc:title>
            <dc:creator>Sting</dc:creator>
            <upnp:class>object.container.album.musicAlbum</upnp:class>
```

```
            <upnp:searchClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:searchClass>
            <upnp:createClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:createClass>
      </container>
      <container id=”3” parentID=”1” childCount=”4” restricted=”false”>
            <dc:title>Singles Soundtrack</dc:title>
            <dc:creator>Various Artists</dc:creator>
            <upnp:class>object.container.album.musicAlbum</upnp:class>
            < upnp:searchClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:searchClass>
            <upnp:createClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:createClass>
      </container>
</DIDL-Lite>”, 2, 2, 21 )
```

2.8.3.5. Browsing the children of the Singles Soundtrack music album

The control point needs to retrieve the children of the Singles Soundtrack music album. The control point can display 3 items at a time, so it restricts the number of children returned in each result. In addition, it specifies the result to be sorted in ascending order by the title property. It does this via the following **Browse()** action:

```
request: Browse( “3”, “BrowseDirectChildren”, “*”, 0, 3, “+dc:title” )
```

```
response: Browse(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <item id=”6” parentID=”3” restricted="false">
            <dc:title>Chloe Dancer</dc:title>
            <dc:creator>Mother Love Bone</dc:creator>
            <upnp:class>object.item.audioItem.musicTrack</upnp:class>
            <res protocolInfo="http-get:*:audio/x-ms-wma:*”
size=”200000”>
                  http://10.0.0.1/getcontent.asp?id=6
            </res>
      </item>
      <item id=”8” parentID=”3” restricted="false">
            <dc:title>Drown</dc:title>
            <dc:creator>Smashing Pumpkins</dc:creator>
            <upnp:class>object.item.audioItem.musicTrack</upnp:class>
            <res protocolInfo="http-get:*:audio/mpeg:*” size=”140000”>
                  http://10.0.0.1/getcontent.asp?id=8
            </res>
      </item>
      <item id=”7” parentID=”3” restricted="false">
            <dc:title>State Of Love And Trust</dc:title>
            <dc:creator>Pearl Jam</dc:creator>
            <upnp:class>object.item.audioItem.musicTrack</upnp:class>
            <res protocolInfo="http-get:*:audio/x-ms-wma:*”
size=”70000”>
                  http://10.0.0.1/getcontent.asp?id=7
```

```
            </res>
      </item>
</DIDL-Lite>", 3, 4, 18 )
```

request: Browse("3", "BrowseDirectChildren", "*", 3, 3, "+dc:title")

```
response: Browse(
"<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
      xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/">
      <item id="5" parentID="3" restricted="false">
            <dc:title>Would</dc:title>
            <dc:creator>Alice In Chains</dc:creator>
            <upnp:class>object.item.audioItem.musicTrack</upnp:class>
            <res protocolInfo="http-get:*:audio/x-ms-wma:*"
size="90000">
                  http://10.0.0.1/getcontent.asp?id=5
            </res>
      </item>
</DIDL-Lite>", 1, 4, 18 )
```

2.8.3.6. Browsing the children of the Album Art folder

The control point needs to retrieve the children of the Album Art folder. The control point can display 3 items at a time so it restricts the number of children returned in the result. It does this via the following **Browse()** action:

request: Browse("30", "BrowseDirectChildren", "*", 0, 3, "")

```
response: Browse(
"<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
      xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/">
      <item id="31" parentID="30" restricted="false">
            <dc:title>Brand New Day</dc:title>
            <upnp:class name="Vendor Album Art">
                  object.item.imageItem.photo.vendorAlbumArt
            </upnp:class>
            <res protocolInfo="http-get:*:image/jpeg:*" size="20000">
                  http://10.0.0.1/getcontent.asp?id=31
            </res>
      </item>
      <item id="32" parentID="30" restricted="false">
            <dc:title>Singles Soundtrack</dc:title>
            <upnp:class name="Vendor Album Art">
                  object.item.imageItem.photo.vendorAlbumArt
            </upnp:class>
            <res protocolInfo="http-get:*:image/jpeg:*" size="20000">
                  http://10.0.0.1/getcontent.asp?id=32
            </res>
      </item>
</DIDL-Lite>", 2, 2, 50 )
```

2.8.4. Searching

The **Search()** action is designed to allow a control point to search for objects in the content directory that match a given search criteria (see section 2.5.5). In addition, the **Search()** action supports the following features:

- **Incremental result retrieval** i.e. in the context of a particular request the control point can restrict the number (and the starting offset) of objects returned in the result.
- **Sorting**. The result can be requested in a particular sort order. The available sort orders are expressed in the return value of the GetSortCapabilities action.
- **Filtering**. The result data can be filtered to only include a subset of the properties available on the object (see section 2.5.7). Note that certain properties may not be filtered out in order to maintain the validity of the result data fragment. If a non-filterable property is left out of the filter set, it will still be included in the result.

The following examples illustrate the typical **Search()** request-response interaction between a control point and a CDS. It assumes the content setup specified in section 2.8.2

2.8.4.1. Retrieving search capabilities

When it connects to the CDS, the control point determines which properties can be used in the search criteria in a **Search()** request. It does this via the **GetSearchCapabilities()** action:

```
request: GetSearchCapabilities()

response:
GetSearchCapabilities("dc:title,dc:creator,dc:date,upnp:class,res@size")
```

2.8.4.2. Search for all content by Sting

Search for all objects where **dc:creator** is *Sting* and sort the result in ascending order by **dc:title**. The control point can only display 3 items at a time so it restricts the number requested. The following **Search()** action is used:

```
request: Search( "0", "dc:creator = "Sting"", "*", 0, 3, "+dc:title" )


response: Search(
"<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
      xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/">
      <item id="9" parentID="4" restricted="false">
            <dc:title>A Thousand Years</dc:title>
            <dc:creator>Sting</dc:creator>
            <upnp:class>object.item.audioItem.musicTrack</upnp:class>
            <res protocolInfo="http-get:*:audio/x-ms-wma:*"
size="100000">
                  http://10.0.0.1/getcontent.asp?id=9
            </res>
      </item>
      <item id="11" parentID="4" restricted="false">
            <dc:title>Big Lie, Small World</dc:title>
            <dc:creator>Sting</dc:creator>
            <upnp:class>object.item.audioItem.musicTrack</upnp:class>
            <res protocolInfo="http-get:*:audio/mpeg:*" size="70000">
```

```
                  http://10.0.0.1/getcontent.asp?id=11
            </res>
      </item>
      <container id=”4” parentID=”1” childCount=”3” restricted=”false”
searchable="true">
            <dc:title>Brand New Day</dc:title>
            <dc:creator>Sting</dc:creator>
            <upnp:class>object.container.album.musicAlbum</upnp:class>
            <upnp:searchClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:searchClass>
            <upnp:createClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:createClass>
      </container>
</DIDL-Lite>”, 3, 4, 10 )

request: Search( “0”, “dc:creator = “Sting””, “*”, 3, 3, “+dc:title” )


response: Search(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <item id=”10” parentID=”4” restricted="false">
            <dc:title>Desert Rose</dc:title>
            <dc:creator>Sting</dc:creator>
            <upnp:class>object.item.audioItem.musicTrack</upnp:class>
            <res protocolInfo="http-get:*:audio/x-ms-wma:*”
size=”50000”>
                  http://10.0.0.1/getcontent.asp?id=10
            </res>
      </item>
</DIDL-Lite>”, 1, 4, 10 )
```

2.8.4.3. Search for all photos taken during October

Search for all photo objects whose **dc:date** is in October and sort the result in ascending order by **dc:date**. The control point can only display 3 items at a time so it restricts the number requested. The following **Search()** action is used:

```
request: Search( “0”, “upnp:class = “object.item.imageItem.photo” and "(
dc:date >= "2001-10-01" and dc:date <= "2001-10-31" )”, “*”, 0, 3,
“+dc:date” )


response: Search(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <item id=”14” parentID=”12” restricted="false">
            <dc:title>Sunset on the beach</dc:title>
            <dc:date>2001-10-20</dc:date>
            <upnp:class>object.item.imageItem.photo</upnp:class>
            <res protocolInfo="http-get:*:image/jpeg:*” size=”20000”>
                  http://10.0.0.1/getcontent.asp?id=14
            </res>
      </item>
      <item id=”15” parentID=”12” restricted="false">
```

```
            <dc:title>Playing in the pool</dc:title>
            <dc:date>2001-10-25</dc:date>
            <upnp:class>object.item.imageItem.photo</upnp:class>
            <res protocolInfo="http-get:*:image/jpeg:*" size="25000">
                  http://10.0.0.1/getcontent.asp?id=15
            </res>
      </item>
</DIDL-Lite>", 2, 2, 10 )
```

2.8.4.4. Search for all objects in the My Photos folder containing the word Christmas

Search for all objects where the title contains Christmas under the *My Photos* folder. The control point can only display 3 items at a time so it restricts the number requested. The result is sorted in ascending order by **dc:title**. The following **Search()** action is used:

```
request: Search( "2", "dc:title contains "Christmas"", "*", 0, 3,
"+dc:title" )
```

```
response: Search(
"<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
      xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/">
   <container id="13" parentID="2" restricted="false" searchable="true">
            <dc:title>Christmas</dc:title>
            <upnp:class>object.container.album.photoAlbum</upnp:class>
            <upnp:searchClass includeDerived="false">
                  object.item.imageItem.photo
            </upnp:searchClass>
            <upnp:createClass includeDerived="false">
                  object.item.imageItem.photo
            </upnp:createClass>
      </container>
      <item id="17" parentID="13" restricted="false">
            <dc:title>Christmas tree loaded with presents</dc:title>
            <dc:date>2001-12-25</dc:date>
            <upnp:class>object.item.imageItem.photo</upnp:class>
            <res protocolInfo="http-get:*:image/jpeg:*" size="25000">
                  http://10.0.0.1/getcontent.asp?id=17
            </res>
      </item>
</DIDL-Lite>", 2, 2, 47 )
```

2.8.4.5. Search for all album objects in the Content Directory

Search for all objects that are derived from **object.container.album**. The following **Search()** action is used:

```
request: Search( "0", "upnp:class derivedfrom "object.container.album"",
"*", 0, 4, "" )
```

```
response: Search(
"<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
      xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/">
```

```
      <container id=”3” parentID=”1” childCount=”4” restricted=”false”
searchable="true">
            <dc:title>Singles Soundtrack</dc:title>
            <dc:creator>Various Artists</dc:creator>
            <upnp:class>object.container.album.musicAlbum</upnp:class>
            < upnp:searchClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:searchClass>
            <upnp:createClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:createClass>
      </container>
      <container id=”4” parentID=”1” childCount=”3” restricted=”false”
searchable="true">
            <dc:title>Brand New Day</dc:title>
            <dc:creator>Sting</dc:creator>
            <upnp:class>object.container.album.musicAlbum</upnp:class>
            <upnp:searchClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:searchClass>
            <upnp:createClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:createClass>
      </container>
      <container id=”12” parentID=”2” restricted=”false”
searchable="true">
            <dc:title>Mexico Trip</dc:title>
            <upnp:class>object.container.album.photoAlbum</upnp:class>
            <upnp:searchClass includeDerived=”false” >
                  object.item.imageItem.photo
            </upnp:searchClass>
            <upnp:createClass includeDerived=”false”>
                  object.item.imageItem.photo
            </upnp:createClass>
      </container>
      <container id=”13” parentID=”2” restricted=”false”
searchable="true">
            <dc:title>Christmas</dc:title>
            <upnp:class>object.container.album.photoAlbum</upnp:class>
            <upnp:searchClass includeDerived=”false” >
                  object.item.imageItem.photo
            </upnp:searchClass>
            <upnp:createClass includeDerived=”false”>
                  object.item.imageItem.photo
            </upnp:createClass>
      </container>
</DIDL-Lite>”, 4, 4, 10 )
```

2.8.5. Browsing, Searching, and References

Using the content setup above, the following examples illustrate creation of a reference, the result of a search where the result contains a reference, and deletion of a reference.

2.8.5.1. Creating a reference to a photo in the Mexico Trip album inside the Christmas album

A reference to an existing **item** is created via the following action:

request: `CreateReference( “13”, “15” )`

```
response: CreateReference( “20” )
```

2.8.5.2. Search for all photos taken during October

Search for all photo objects whose **dc:date** is in October and sort the result in ascending order by **dc:date**. The control point can only display 3 items at a time so it restricts the number requested. The following **Search()** action is used:

```
request: Search( “0”, “upnp:class = “object.item.imageItem.photo” and (
dc:date >= "2001-10-01" and dc:date <= "2001-10-31" )”, “*”, 0, 3,
“+dc:date” )


response: Search(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <item id=”14” parentID=”12” restricted="false">
            <dc:title>Sunset on the beach</dc:title>
            <dc:date>2001-10-20</dc:date>
            <upnp:class>object.item.imageItem.photo</upnp:class>
            <res protocolInfo="http-get:*:image/jpeg:*” size=”20000”>
                  http://10.0.0.1/getcontent.asp?id=14
            </res>
      </item>
      <item id=”15” parentID=”12” restricted="false">
            <dc:title>Playing in the pool</dc:title>
            <dc:date>2001-10-25</dc:date>
            <upnp:class>object.item.imageItem.photo</upnp:class>
            <res protocolInfo="http-get:*:image/jpeg:*” size=”25000”>
                  http://10.0.0.1/getcontent.asp?id=15
            </res>
      </item>
      <item id=”20” refID=”15” parentID=”13” restricted="false">
            <dc:title>Playing in the pool</dc:title>
            <dc:date>2001-10-25</dc:date>
            <upnp:class>object.item.imageItem.photo</upnp:class>
            <res protocolInfo="http-get:*:image/jpeg:*” size=”25000”>
                  http://10.0.0.1/getcontent.asp?id=15
            </res>
      </item>
</DIDL-Lite>”, 3, 3, 10 )
```

2.8.5.3. Deletion of the reference to the photo in the Mexico trip album

A reference is deleted via the following action:

```
request: DestroyObject( “20” )

response: DestroyObject( )
```

2.8.6. Browsing, Searching, and Filtering

Using the content setup above, the following examples illustrate filtering of properties. In each case, a **Search()** is performed for a particular item id, but with filtering of different properties.

2.8.6.1. Result when requesting all properties

The following **Search()** action is used:

```
request: Search( "0", "@id = “20”", "*", 0, 3, "" )

response: Search(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <item id=”18” parentID=”13” restricted="false">
            <dc:title>John and Mary by the fire</dc:title>
            <dc:date>2001-12-24</dc:date>
            <upnp:class>object.item.imageItem.photo</upnp:class>
            <res protocolInfo="http-get:*:image/jpeg:*” size=”22000”>
                  http://10.0.0.1/getcontent.asp?id=18
            </res>
      </item>
</DIDL-Lite>”, 1, 1, 10 )
```

2.8.6.2. Result when requesting required properties, and leaving out properties expressed as elements e.g. dc:date, res

The following **Search()** action is used:

```
request: Search( “0”, “@id = “20””,
“@id,@parentID,@restricted,dc:title,upnp:class”, 0, 3, “” )


response: Search(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <item id=”18” parentID=”13” restricted="false">
            <dc:title>John and Mary by the fire</dc:title>
            <upnp:class>object.item.imageItem.photo</upnp:class>
      </item>
</DIDL-Lite>”, 1, 1, 10 )
```

Note that only the properties requested are returned in the search result.

2.8.6.3. Result when requesting required properties, and leaving out properties expressed as attributes e.g. res@size

The following **Search()** action is used:

```
request: Search( “0”, “@id = “20””,
“@id,@parentID,@restricted,dc:title,dc:date,upnp:class,res,res@protocolI
nfo”, 0, 3, “” )


response: Search(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <item id=”18” parentID=”13” restricted="false">
            <dc:title>John and Mary by the fire</dc:title>
            <dc:date>2001-12-24</dc:date>
            <upnp:class>object.item.imageItem.photo</upnp:class>
```

```
            <res protocolInfo="http-get:*:image/jpeg:*”>
                  http://10.0.0.1/getcontent.asp?id=18
            </res>
      </item>
</DIDL-Lite>”, 1, 1, 10 )
```

Note that only the properties requested are returned in the search result.

2.8.6.4. Result when leaving out required properties e.g. @id, @parentID, @restricted, upnp:class, dc:title, and leaving out properties expressed as attributes e.g. res@size

The following **Search()** action is used:

```
request: Search( “0”, “@id = “20””, “dc:date,res,res@protocolInfo”, 0,
3, “” )
```

```
response: Search(
“<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <item id=”18” parentID=”13” restricted="false">
            <dc:title>John and Mary by the fire</dc:title>
            <dc:date>2001-12-24</dc:date>
            <upnp:class>object.item.imageItem.photo</upnp:class>
            <res protocolInfo="http-get:*:image/jpeg:*”>
                  http://10.0.0.1/getcontent.asp?id=18
            </res>
      </item>
</DIDL-Lite>”, 1, 1, 10 )
```

Note that the required properties, and the properties requested are returned in the search result.

2.8.7. Object Creation

2.8.7.1. Creating New Objects

The **CreateObject()** action is used to create a new object in the specified container. In general, Control Points invoke the **CreateObject()** with meta-data that includes the **upnp:class** element for the newly created object. The Content Directory Service will create an object according to the specified meta-data and additional meta-data may be added by the CDS. The action returns **ObjectID** and meta-data of the created object. Note that all required elements must exist in the **Result**.

2.8.7.2. Example: Creating a new MusicTrack in the Album1 (id=10)

Invoke **CreateObject ()** with the **ContainerID** parameter set to *10* and the **Elements** parameter set to meta-data that contained **upnp:class** of class **id**=*MusicTrack*.

```
  request: CreateObject(“10”,

      "<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite”>
      <item id="" restricted="false">
            <dc:title>New Track</dc:title>
```

```
            <upnp:class>
                  object.item.audioItem.musicTrack
            </upnp:class>
      </item>
      </DIDL-Lite>")

  response: CreateObject("12",

      "<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
      xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/">
        <item id="12" parentID="10" restricted="false">
            <dc:title>New Track</dc:title>
            <dc:creator></dc:creator>
            <res importUri="http:/pc/item?id=12"
                  protocolInfo="*:*:audio:*"/>
            <upnp:class>
                  object.item.audioItem.musicTrack
            </upnp:class>
            <upnp:genre></upnp:genre>
            <upnp:album>Album1</upnp:album>
        </item>
      </DIDL-Lite>")
```

2.8.8. File Transfer of a resource in Objects

The **ImportResource()** action and **ExportResource()** action are used for a file transfer of resources in an object. A <**res**> element identifies a resource of the object and contains the uri locator of the resource to access. The **protocolInfo** attribute indicates the protocol for exporting the resource and HTTP protocol is recommended for the purpose of a file transfer. The **importUri** attribute indicates the uri of the resource used in the **ImportResource()** action.

Shown below are three cases of <**res**> elements. Generally when an object is created, it has no resource. If the object has resources to be access, the <**res**> element has a uri value and designates a value for the **protocolInfo** attribute. Note that the <**res**> value and the **protocolInfo** attribute value may be updated by the file transfer from a remote resource to a local resource in the object.

2.8.8.1. case A

The <**res**> element contains no resource. A Control Point can import a resource to the object, and the value of the **importUri** attribute is used in the **ImportResource()** action. For example,

```
<res protocolInfo="*:*:audio:*"
    importUri="http://10.0.0.1/cd/import?id=3"/>
```

2.8.8.2. case B

The <**res**> element has a resource to be accessed by Control Points. The Control Point cannot replace the resource of the <**res**> element. For example,

```
<res protocolInfo="http-get:*:audio/m3u:*">
      http://10.0.0.1/cd/content?id=3
</res>
```

2.8.8.3. case C

The <**res**> element has assigned a resource. The Control Point can replace the assigned resource using the **ImportResource()** action. For example,

```
<res protocolInfo="http-get:*:audio/m3u:*"
   importUri="http://10.0.0.1/cd/import?id=3">
      http://10.0.0.1/cd/content?id=3
</res>
```

2.8.8.4. Transferring using the ImportResource() action

The destination resource, e.g. http://10.0.0.1/cd/import?id=3, is located in the CDS and the source resource, e.g. http://server/song.mp3", is located in another CDS, or is any resource identified by uri. If a control point wants to create a new object that imports a file into a resource, it can invoke the **CreateObject()** action before the file transfer.

For example, the <**res**> element is as shown below before a file transfer.

```
<res protocolInfo="*:*:audio:*"
   importUri="http://10.0.0.1/cd/import?id=3"/>
```

A control point invokes **ImportResource()** and the CDS identifies the local destination resource.

request: ImportResource("http://server/song.mp3", "http::://10.0.0.1/cd/import?id=3")

response: ImportResource(1234)

The CDS initiates the HTTP::GET to the remote source resource and begins receiving data which is directed to the local destination resource. Note that the specified scheme of a remote source resource is used by the CDS.

GET /song.mp3 HTTP/1.1

HTTP/1.1 200 OK

The control point may query the progress

request: GetTransferProgress(1234)

response: GetTransferProgress("IN_PROGRESS", 43852, 125327)

Finally, HTTP::GET has finished successfully.

The control point can query the result of file transfer.

request: GetTransferProgress(1234)

response: GetTransferProgress("COMPLETED", 125327, 125327)

If a control point receives events from the CDS, the control point receives two events of the **TransferIDs** status variable.

A file transfer starts at step 1)

event: TransferIDs="1234".

A file transfer ends at step 4) successfully or when it fails by an error or is stopped by the **StopTransferResource()** action.

event:TransferIDs="".

After the file transfer, <**res**> element is as shown below.

```
<res protocolInfo="http-get:*:audio/m3u:*">
```

```
      http://10.0.0.1/cd/content?id=3
</res>
```

2.8.8.5. Transfer using direct HTTP::POST

If the source resource is located in the control point, it is obviously easy for a control point to access the destination resource directly.

A control point initiates HTTP::POST at the destination resource and begins sending data from the local source resource. Note that the specified scheme of a remote destination resource is used by the control point.

POST /cd/content?id=3 HTTP/1.1

HTTP/1.1 200 OK

2.8.8.6. Transfer using the ExportResource() action

The source resource, e.g."http://10.0.0.1/cd/content?id=3", is located locally, and the destination resource, e.g."http://server/content?id=6", is located remotely, or is generally any resource identified by URI.

For example, the <**res**> element is as shown below before a file transfer.

```
<res protocolInfo="http-get:*:audio/m3u:*″>
      http://10.0.0.1/cd/content?id=3
</res>
```

1) A control point invokes **ExportResource()** and the CDS identifies the local source resource.

   `request`: ExportResource("http://10.0.0.1/cd/content?id=3","http://server/content?id=6")

   `response`: ExportResource(1235)

2) The CDS initiates the HTTP::POST at the remote destination resource and begins sending data from the local source resource. Note that the specified scheme of a remote destination resource is used by the CDS.

   POST content?id=6 HTTP/1.1

   HTTP/1.1 200 OK

The subsequent steps are as same as the case of **ImportResource().**

2.8.8.7. Transfer using direct HTTP::GET

If the destination resource is located in the control point, it is obviously easy for a control point to access the source resource directly.

A control point initiates HTTP::GET at the remote source resource and begins receiving data which is directed to the local destination resource. Note that the specified scheme of a remote source resource is used by the control point.

2.8.9. Playlist Manipulation

2.8.9.1. Playlist file representation in CDS

A playlist file is represented as an object of the playlist class (object.item.playlist). The format of the playlist is indicated by the MIME type section of the **res@protocolInfo** property on the playlist object. If a search were performed for all objects of class object.item.playlist in the content directory, it would return a result of the following form:

```
<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <item id=”2” parentID=”1” restricted="false">
            <dc:title>Playlist of John and Mary's music</dc:title>
            <dc:creator>John Jones</dc:creator>
            <upnp:class>object.item.playlist</upnp:class>
            <res protocolInfo="http-get:*:audio/m3u:*”>
                http://pc/k.m3u
            </res>
      </item>
</DIDL-Lite>
```

2.8.9.2. Playlist file generation from container derived object resources

Objects derived from the container class (object.container) may contain objects derived from the item (object.item) or container classes. An example of such a class is the musicAlbum class (object.container.album.musicAlbum). It is desired to allow a control point to set up a rendering session of all the items in the music album. This may be accomplished by having the album object expose a resource property (**res**), and the URI of the resource generates a playlist file in a format that is understood by the media renderer. The content of the playlist file is a list of all the items in the container.

Note: the order of the items in the playlist file is defined by the generator of the playlist, but should match the order of the items as returned from the **Browse()** action on that container. The following example illustrates:

- A **Browse()** of a musicAlbum object’s metadata would return a result of the following form:

```
<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp=”urn:schemas-upnp-org:metadata-1-0/upnp/”
      xmlns=”urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/”>
      <container id=”1” parentID=”0” resricted="false"
searchable="true">
            <dc:title>Brand New Day</dc:title>
            <dc:creator>Sting</dc:creator>
            <upnp:class>
                  object.container.album.musicAlbum
            </upnp:class>
            <upnp:searchClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:searchClass>
            <upnp:createClass includeDerived=”false”>
                  object.item.audioItem.musicTrack
            </upnp:createClass>
            <res protocolInfo="http-get:*:audio/m3u:*”>
                  http://pc/genm3u?containerID="1"
            </res>
      </container>
</DIDL-Lite>
```

- A **Browse()** of that musicAlbum object's direct children would return a result of the following form:

```
<DIDL-Lite xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
      xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/">
      <item id="2" parentID="1" restricted="false">
            <dc:title>A Thousand Years</dc:title>
            <dc:creator>Sting</dc:creator>
            <upnp:class>
                  object.item.audioItem.musicTrack
            </upnp:class>
            <res protocolInfo="http-get:*:audio/x-ms-wma:*">
                  http://pc/getcontent?contentID="2"
            </res>
      </item>
      <item id="3" parentID="1">
            <dc:title>Desert Rose</dc:title>
            <dc:creator>Sting</dc:creator>
            <upnp:class>
                  object.item.audioItem.musicTrack
            </upnp:class>
            <res protocolInfo="http-get:*:audio/x-ms-wma:*">
                  http://pc/getcontent?contentID="3"
            </res>
      </item>
</DIDL-Lite>
```

- The control point would use the content of the <**res**> element on the musicAlbum **container** object to set the URI on the AVTransport Service of the media renderer. The media renderer would issue an HTTP GET on the URI `"http://pc/genm3u?containerID="1""` to retrieve the generated M3U resource which would have the following content:

```
http://pc/getcontent?contentID="2"
http://pc/getcontent?contentID="3"
```

2.8.10.Internet Content Representation

A CDS implementation will always reside on a UPnP device. However, various URIs present as metadata inside the Content Directory are allowed to point to locations, e.g., web servers, that are outside the UPnP network. For example, an Internet Radio station may be represented by an object in a CDS hosted by a UPnP MediaServer device.

In order to be compatible with as many renderer (player) devices in the UPnP home network as possible, a MediaServer device may be able to perform protocol and/or format conversion of content. Protocol and format information is exposed via the DIDL-lite <**res**> element. MediaServer devices that can serve content using multiple protocols will generally have multiple <**res**> elements for a single object. For example, consider an Internet video resource using RTSP/RTP/UDP. To accomodate MediaRenderer devices that can only play via HTTP, a MediaServer could provide protocol translation, and offer the following meta data:

```
<item id="InternetStream1" restricted="false">
      <res protocolInfo="rtsp-rtp-udp:*:MPV:*">
            rtsp://internet-server/stream1.m2v
      </res>
      <res protocolInfo="http-get:*:video/mpeg:*">
```

```
            http://upnp-device/stream1.m2v
      </res>
</item>
```

MediaRenderer devices that can deal with RTSP/RTP/UDP streams can play from the Internet server directly, whereas MediaRenderer devices that can only deal with HTTP streams would stream the same content over HTTP via the MediaServer device that acts as a translating proxy.

2.8.11.Vendor Metadata Extensions

Vendors may extend DIDL-Lite metadata by placing blocks of vendor-specific metadata into <**desc**> blocks. In DIDL-Lite, a <**desc**> element identifies a **descriptor**. The required **nameSpace** attribute identifies the namespace of the contained metadata. <**desc**> elements may appear as child elements of <**DIDL-Lite**> root elements, <**container**>, and <**item**> elements. The contents of each <desc> must be associated with only one namespace.

A **descriptor** is employed to associate blocks of other XML-based metadata with a given CDS object. Examples of other XML-based metadata include DIG35, MPEG7, RDF, XrML, and etc. **descriptor** blocks could also be employed to contain vendor-specific content ratings information, digitally signed rights descriptions, and etc.

Allowing the <**desc**> to contain only elements from the namespace defined by the **nameSpace** attribute allows control points vendors to selectively deploy support for a given namespace using parser 'plug-in' techniques. <**desc**> blocks designating unfamiliar namespaces are ignored by the control point.

3. XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
    <name>GetSearchCapabilities</name>
      <argumentList>
        <argument>
          <name>SearchCaps</name>
          <direction>out</direction>
         <relatedStateVariable>SearchCapabilities</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>GetSortCapabilities</name>
      <argumentList>
        <argument>
          <name>SortCaps</name>
          <direction>out</direction>
          <relatedStateVariable>SortCapabilities</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>GetSystemUpdateID</name>
      <argumentList>
        <argument>
          <name>Id</name>
          <direction>out</direction>
          <relatedStateVariable>SystemUpdateID</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>Browse</name>
      <argumentList>
        <argument>
          <name>ObjectID</name>
          <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_ObjectID</relatedStateVariable>
        </argument>
        <argument>
          <name>BrowseFlag</name>
          <direction>in</direction>
      <relatedStateVariable>A_ARG_TYPE_BrowseFlag</relatedStateVariable>
        </argument>
        <argument>
          <name>Filter</name>
          <direction>in</direction>
```

```
          <relatedStateVariable>A_ARG_TYPE_Filter</relatedStateVariable>
        </argument>
        <argument>
          <name>StartingIndex</name>
          <direction>in</direction>
          <relatedStateVariable>A_ARG_TYPE_Index</relatedStateVariable>
        </argument>
        <argument>
          <name>RequestedCount</name>
          <direction>in</direction>
          <relatedStateVariable>A_ARG_TYPE_Count</relatedStateVariable>
        </argument>
        <argument>
          <name>SortCriteria</name>
          <direction>in</direction>
    <relatedStateVariable>A_ARG_TYPE_SortCriteria</relatedStateVariable>
        </argument>
        <argument>
          <name>Result</name>
          <direction>out</direction>
          <relatedStateVariable>A_ARG_TYPE_Result</relatedStateVariable>
        </argument>
        <argument>
          <name>NumberReturned</name>
          <direction>out</direction>
          <relatedStateVariable>A_ARG_TYPE_Count</relatedStateVariable>
        </argument>
        <argument>
          <name>TotalMatches</name>
          <direction>out</direction>
          <relatedStateVariable>A_ARG_TYPE_Count</relatedStateVariable>
        </argument>
        <argument>
          <name>UpdateID</name>
          <direction>out</direction>
        <relatedStateVariable>A_ARG_TYPE_UpdateID</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>Search</name>
      <argumentList>
        <argument>
          <name>ContainerID</name>
          <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_ObjectID</relatedStateVariable>
        </argument>
        <argument>
          <name>SearchCriteria</name>
          <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_SearchCriteria</relatedStateVariable>
        </argument>
        <argument>
          <name>Filter</name>
          <direction>in</direction>
          <relatedStateVariable>A_ARG_TYPE_Filter</relatedStateVariable>
```

```
      </argument>
      <argument>
        <name>StartingIndex</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_Index</relatedStateVariable>
      </argument>
      <argument>
        <name>RequestedCount</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_Count</relatedStateVariable>
      </argument>
      <argument>
        <name>SortCriteria</name>
        <direction>in</direction>
    <relatedStateVariable>A_ARG_TYPE_SortCriteria</relatedStateVariable>
      </argument>
      <argument>
        <name>Result</name>
        <direction>out</direction>
        <relatedStateVariable>A_ARG_TYPE_Result</relatedStateVariable>
      </argument>
      <argument>
        <name>NumberReturned</name>
        <direction>out</direction>
        <relatedStateVariable>A_ARG_TYPE_Count</relatedStateVariable>
      </argument>
      <argument>
        <name>TotalMatches</name>
        <direction>out</direction>
        <relatedStateVariable>A_ARG_TYPE_Count</relatedStateVariable>
      </argument>
      <argument>
        <name>UpdateID</name>
        <direction>out</direction>
       <relatedStateVariable>A_ARG_TYPE_UpdateID</relatedStateVariable>
      </argument>
    </argumentList>
  </action>
  <action>
  <name>CreateObject</name>
    <argumentList>
      <argument>
        <name>ContainerID</name>
        <direction>in</direction>
      <relatedStateVariable>A_ARG_TYPE_ObjectID</relatedStateVariable>
      </argument>
      <argument>
        <name>Elements</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_Result</relatedStateVariable>
      </argument>
      <argument>
        <name>ObjectID</name>
        <direction>out</direction>
      <relatedStateVariable>A_ARG_TYPE_ObjectID</relatedStateVariable>
      </argument>
```

```
      <argument>
        <name>Result</name>
        <direction>out</direction>
        <relatedStateVariable>A_ARG_TYPE_Result</relatedStateVariable>
      </argument>
    </argumentList>
  </action>
  <action>
  <name>DestroyObject</name>
    <argumentList>
      <argument>
        <name>ObjectID</name>
        <direction>in</direction>
      <relatedStateVariable>A_ARG_TYPE_ObjectID</relatedStateVariable>
      </argument>
    </argumentList>
  </action>
  <action>
  <name>UpdateObject</name>
    <argumentList>
      <argument>
        <name>ObjectID</name>
        <direction>in</direction>
      <relatedStateVariable>A_ARG_TYPE_ObjectID</relatedStateVariable>
      </argument>
      <argument>
        <name>CurrentTagValue</name>
        <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_TagValueList</relatedStateVariable>
      </argument>
      <argument>
        <name>NewTagValue</name>
        <direction>in</direction>
  <relatedStateVariable>A_ARG_TYPE_TagValueList</relatedStateVariable>
      </argument>
    </argumentList>
  </action>
  <action>
  <name>ImportResource</name>
    <argumentList>
      <argument>
        <name>SourceURI</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_URI</relatedStateVariable>
      </argument>
      <argument>
        <name>DestinationURI</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_URI</relatedStateVariable>
      </argument>
      <argument>
        <name>TransferID</name>
        <direction>out</direction>
   <relatedStateVariable>A_ARG_TYPE_TransferID</relatedStateVariable>
      </argument>
    </argumentList>
```

```
    </action>
    <action>
    <name>ExportResource</name>
      <argumentList>
        <argument>
          <name>SourceURI</name>
          <direction>in</direction>
          <relatedStateVariable>A_ARG_TYPE_URI</relatedStateVariable>
        </argument>
        <argument>
          <name>DestinationURI</name>
          <direction>in</direction>
          <relatedStateVariable>A_ARG_TYPE_URI</relatedStateVariable>
        </argument>
        <argument>
          <name>TransferID</name>
          <direction>out</direction>
      <relatedStateVariable>A_ARG_TYPE_TransferID</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>StopTransferResource</name>
      <argumentList>
        <argument>
          <name>TransferID</name>
          <direction>in</direction>
      <relatedStateVariable>A_ARG_TYPE_TransferID</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>GetTransferProgress</name>
      <argumentList>
        <argument>
          <name>TransferID</name>
          <direction>in</direction>
      <relatedStateVariable>A_ARG_TYPE_TransferID</relatedStateVariable>
        </argument>
        <argument>
          <name>TransferStatus</name>
          <direction>out</direction>
  <relatedStateVariable>A_ARG_TYPE_TransferStatus</relatedStateVariable>
        </argument>
        <argument>
          <name>TransferLength</name>
          <direction>out</direction>
  <relatedStateVariable>A_ARG_TYPE_TransferLength</relatedStateVariable>
        </argument>
        <argument>
          <name>TransferTotal</name>
          <direction>out</direction>
   <relatedStateVariable>A_ARG_TYPE_TransferTotal</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
```

```
    <action>
    <name>DeleteResource</name>
      <argumentList>
        <argument>
          <name>ResourceURI</name>
          <direction>in</direction>
          <relatedStateVariable>A_ARG_TYPE_URI</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>CreateReference</name>
      <argumentList>
        <argument>
          <name>ContainerID</name>
          <direction>in</direction>
     <relatedStateVariable>A_ARG_TYPE_ObjectID</relatedStateVariable>
        </argument>
        <argument>
          <name>ObjectID</name>
          <direction>in</direction>
     <relatedStateVariable>A_ARG_TYPE_ObjectID</relatedStateVariable>
        </argument>
        <argument>
          <name>NewID</name>
          <direction>out</direction>
     <relatedStateVariable>A_ARG_TYPE_ObjectID</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    Declarations for other actions added by UPnP vendor (if any) go here
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="yes">
      <name>TransferIDs</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_ObjectID</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_Result</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_SearchCriteria</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_BrowseFlag</name>
      <dataType>string</dataType>
      <allowedValueList>
        <allowedValue>BrowseMetadata</allowedValue>
        <allowedValue>BrowseDirectChildren</allowedValue>
      </allowedValueList>
```

```
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_Filter</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_SortCriteria</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_Index</name>
  <dataType>ui4</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_Count</name>
  <dataType>ui4</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_UpdateID</name>
  <dataType>ui4</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_TransferID</name>
  <dataType>ui4</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_TransferStatus</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>COMPLETED</allowedValue>
    <allowedValue>ERROR</allowedValue>
    <allowedValue>IN_PROGRESS</allowedValue>
    <allowedValue>STOPPED</allowedValue>
  </allowedValueList>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_TransferLength</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_TransferTotal</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_TagValueList</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_URI</name>
  <dataType>uri</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>SearchCapabilities</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
```

```
      <name>SortCapabilities</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>SystemUpdateID</name>
      <dataType>ui4</dataType>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>ContainerUpdateIDs</name>
      <dataType>string</dataType>
    </stateVariable>
    Declarations for other state variables added by UPnP vendor (if any)
    go here
  </serviceStateTable>
</scpd>
```

4. Test

No semantic tests have been specified for this service.

5. Appendix A - DIDL-Lite

This section contains a complete XML schema for the DIDL-Lite element set. The UPnP, Dublin Core and XML namespaces are imported into the DIDL-Lite schema.

DIDL-Lite is derived from a subset of DIDL, a Digital Item Declaration Language recently developed within ISO/MPEG21 [DIDL].

The attached DIDL-Lite schema may be copied at the 'cut' lines and saved into a file for use in a validating parser or instance document editing tool.

It is anticipated that few if any, UPnP A/V Control Points or Content Directories will employ schema-based validation in the implementation of A/V functionality. In any case, the schema serves as the normative reference for the format of DIDL-Lite fragments.

The schema, however, may have a use in testing and certifying the UPnP A/V standard compliance of UPnP A/V Control Points and UPnP A/V Content Directory Services (see section 4.0).

This DIDL-Lite schema has been constructed using the May 2, 2001 W3C XML Schema Recommendation.

----------(cut here) ----------

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/" xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:didl-lite="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">DIDL-Lite schema for UPnP A/V Content Directory Services, version 1.0.</xsd:documentation>
    </xsd:annotation>
    <xsd:import namespace="urn:schemas-upnp-org:metadata-1-0/upnp/" schemaLocation="upnp.xsd"/>
    <xsd:import namespace="http://purl.org/dc/elements/1.1/" schemaLocation="simpledc20020312.xsd"/>
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="http://www.w3.org/2001/xml.xsd"/>
    <!--============================================================

     'DIDL-Lite' is the root element of DIDL-Lite documents.

     Attributes:
     xml:lang    use: optional

    The 'xml:lang' attribute may optionally be used to specify the language of text
    in the DIDL-Lite document.
    ============================================================-->
    <xsd:group name="allowed-under-DIDL-Lite">
        <xsd:annotation>
            <xsd:documentation>This group defines the elements allowed under the DIDL-Lite root</xsd:documentation>
        </xsd:annotation>
        <xsd:choice maxOccurs="unbounded">
            <xsd:element ref="didl-lite:item"/>
            <xsd:element ref="didl-lite:container"/>
            <xsd:element ref="didl-lite:desc"/>
        </xsd:choice>
    </xsd:group>
    <xsd:element name="DIDL-Lite" type="didl-lite:rootType"/>
    <xsd:complexType name="rootType">
        <xsd:annotation>
            <xsd:documentation>DIDL-Lite is the root element</xsd:documentation>
        </xsd:annotation>
```

```
        <xsd:group ref="didl-lite:allowed-under-DIDL-Lite"/>
        <xsd:attribute ref="xml:lang"/>
    </xsd:complexType>
    <!--============================================================

    A 'container' element may contain any number of 1. Dublin Core, 2. upnp, 3. res,
    4. ref, 5 item,  6. container and 7. desc elements.  In all cases, the first element in
    each container child element sequence is required to be "dc:title".  The 'upnp:class'
    element must also appear under container.  Each container is required to specify
    a value for the 'id' and 'parentID' attributes. Each container is also required to specify a value for
    the 'restricted' attribute (true, false, 1, 0).   When restricted="true", the ability to
    change or delete the container is restricted. Other optional container element
    attributes are 'parentID', 'childCount', and 'searchable'. Other optional attributes
    are 'parentID' and 'childCount'.

     Attributes:
     id          type:   string        use: required
     parentID    type:       string        use: required
     childCount type:    int           use: optional
     restricted  type:   boolean           use: required

     The equivalent MPEG21 DIDL element is 'CONTAINER'

    ============================================================-->
    <xsd:group name="allowed-under-container">
        <xsd:annotation>
            <xsd:documentation>This group defines the elements allowed under the 'container'
element</xsd:documentation>
        </xsd:annotation>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:any namespace="urn:schemas-upnp-org:metadata-1-0/upnp/" processContents="strict"/>
            <xsd:any namespace="http://purl.org/dc/elements/1.1/" processContents="strict"/>
            <xsd:element ref="didl-lite:desc"/>
            <xsd:element ref="didl-lite:item"/>
            <xsd:element ref="didl-lite:container"/>
            <xsd:element ref="didl-lite:res"/>
        </xsd:choice>
    </xsd:group>
    <xsd:element name="container" type="didl-lite:containerType"/>
    <xsd:complexType name="containerType">
        <xsd:annotation>
            <xsd:documentation>A 'container' element may contain any number of 1. Dublin Core, 2. upnp, 3. res,
4. ref, 5 item,  6. container and 7. desc elements.  In all cases, the first element in each container child element
sequence is required to be "dc:title".  The 'upnp:class' element must also appear under container.  Each container
is required to specify a value for the 'id' and 'parentID' attributes. Each container is also required to specify a value
for the 'restricted' attribute (true, false, 1, 0).   When restricted="true", the ability to change or delete the container
is restricted. Other optional container element attributes are 'childCount', and 'searchable'. </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="dc:title"/>
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:group ref="didl-lite:allowed-under-container"/>
            </xsd:choice>
            <xsd:element ref="upnp:class"/>
            <xsd:group ref="didl-lite:allowed-under-container"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
        <xsd:attribute name="restricted" type="xsd:boolean" use="required"/>
        <xsd:attribute name="parentID" type="xsd:string" use="required"/>
        <xsd:attribute name="searchable" type="xsd:boolean"/>
        <xsd:attribute name="childCount" type="xsd:int"/>
    </xsd:complexType>
    <!--============================================================

    An 'item' element contains any number of 1. Dublin Core, 2. upnp, 3. res,  and
    4. desc elements.   In all cases, the first element in each item child element
```

```
        sequence is required to be "dc:title".  The 'upnp:class' element must also appear
        under item.   Each item is additionally required to specify a value for the 'id'
        attribute.  If the item is actually a reference to another item, a value for 'refID'
        is specified.  Each item is also required to specify a value for the 'parentID
        attribute, as well as the 'restricted' attribute (true, false, 1, 0).  When
        restricted="true", the ability to change or delete the item is restricted.

        Attributes:
        id          type:   string      use: required
        parentID    type:   string      use: required
        refID       type:   string      use: optional
        restricted  type:   boolean     use: required

        The equivalent MPEG21 DIDL element is 'ITEM'.

    ============================================================-->
    <xsd:group name="allowed-under-item">
        <xsd:annotation>
            <xsd:documentation>This group defines the elements allowed under the 'item'
element</xsd:documentation>
        </xsd:annotation>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:any namespace="urn:schemas-upnp-org:metadata-1-0/upnp/" processContents="strict"/>
            <xsd:any namespace="http://purl.org/dc/elements/1.1/" processContents="strict"/>
            <xsd:element ref="didl-lite:desc"/>
            <xsd:element ref="didl-lite:res"/>
        </xsd:choice>
    </xsd:group>
    <xsd:element name="item" type="didl-lite:itemType"/>
    <xsd:complexType name="itemType">
        <xsd:annotation>
            <xsd:documentation>An 'item' element contains any number of 1. Dublin Core, 2. upnp, 3. res,  and 4.
desc elements.   In all cases, the first element in each item child element sequence is required to be "dc:title".  The
'upnp:class' element must also appear under item.   Each item is additionally required to specify a value for the 'id'
attribute.  If the item is actually a reference to another item, a value for 'refID' is specified.  Each item is also
required to specify a value for the 'parentID attribute, as well as the 'restricted' attribute (true, false, 1, 0).  When
restricted="true", the ability to change or delete the item is restricted.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element ref="dc:title"/>
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:group ref="didl-lite:allowed-under-item"/>
            </xsd:choice>
            <xsd:element ref="upnp:class"/>
            <xsd:group ref="didl-lite:allowed-under-item"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
        <xsd:attribute name="refID" type="xsd:string"/>
        <xsd:attribute name="parentID" type="xsd:string" use="required"/>
        <xsd:attribute name="restricted" type="xsd:boolean" use="required"/>
    </xsd:complexType>
    <!--============================================================

        A 'res' element indentifies a resource.  A resource is typically some type of a
        binary asset, such as photo, song, video, etc.  A 'res' element contains a uri
        that identifies the resource.

        Attributes:

        'importUri'         type:   anyURI      use:    optional
        The 'importUri' attribute is the optional uri locator for resource update.

        'protocolInfo'      type:   string      use:    required
        The 'protocolInfo' attribute is a string that identifies the streaming or transport protocol for transmitting
        the resource.  If not present then the content has not yet been fully imported by the ContentDirectory and is
        not yet  accesible for playback purposes.
```

'size' type: unsignedLong use: optional
The size, in bytes, of the resource.

'duration' type: string use: optional
The 'duration' attribute identifies the duration of the playback of the resource, at normal speed.
The form of the duration string is:
H*:MM:SS.F*, or H*:MM:SS.F0/F1
where :
H* means any number of digits (including no digits) to indicate elapsed hours
MM means exactly 2 digits to indicate minutes (00 to 59)
SS means exactly 2 digits to indicate seconds (00 to 59)
F* means any number of digits (including no digits) to indicate fractions of seconds
F0/F1 means a fraction, with F0 and F1 at least one digit long, and F0 < F1
The string may be preceded by an optional + or – sign, and the decimal point itself may be omitted if there are no fractional second digits.

'bitrate' type: unsignedInt use: optional
The bitrate in bytes/second of the resource.

'sampleFrequency' type: unsignedInt use: optional
The sample frequency of the resource in Hz

'bitsPerSample' type: unsignedInt use: optional
The bits per sample of the resource.

'nrAudioChannels' type: unsignedInt use: optional
Number of audio channels of the resource, e.g. 1 for mono, 2 for stereo, 6 for Dolby surround, etc.

'resolution' type: pattern-string use: optional
X*Y resolution of the resource (image or video). The string pattern is restricted to strings of the form:
[0-9]+x[0-9]+ (one or more digits,'x', followed by one or more digits).

'colorDepth' type: unsignedInt use: optional
The color depth in bits of the resource (image or video).

'protection' type: string use: optional
Some statement of the protection type of the resource (not standardized).

The equivalent MPEG21 DIDL element is 'RESOURCE'.

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

```
<xsd:element name="res" type="didl-lite:resType"/>
<xsd:complexType name="resType" mixed="true">
    <xsd:annotation>
        <xsd:documentation>A 'res' element indentifies a resource. A resource is typically some type of a binary asset, such as photo, song, video, etc. A 'res' element contains a uri that identifies the resource.</xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:restriction base="xsd:string">
            <xsd:attribute name="importUri" type="xsd:anyURI"/>
            <xsd:attribute name="protocolInfo" type="xsd:string" use="required"/>
            <xsd:attribute name="size" type="xsd:unsignedLong"/>
            <xsd:attribute name="duration" type="xsd:string"/>
            <xsd:attribute name="bitrate" type="xsd:unsignedInt"/>
            <xsd:attribute name="sampleFrequency" type="xsd:unsignedInt"/>
            <xsd:attribute name="bitsPerSample" type="xsd:unsignedInt"/>
            <xsd:attribute name="nrAudioChannels" type="xsd:unsignedInt"/>
            <xsd:attribute name="resolution">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:pattern value="[0-9]+x[0-9]+"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
```

```
                <xsd:attribute name="colorDepth" type="xsd:unsignedInt"/>
                <xsd:attribute name="protection" type="xsd:string"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <!--============================================================

     A 'desc' element identifies a descriptor.  A descriptor is intended to contain a block of metadata.  A bio
     of a music artist is an example use of 'desc'.  A 'desc' element may possess child elements from any
     namespace except the DIDL-Lite namespace. Values for 'id' and 'nameSpace' is required.  An optional
     'type' attribute allows designation of the metadata type, e.g. 'ratings', 'rights', etc.

     Attributes:
    'id'              type:   string      use:  optional
    'name'            type:   string      use:  optional
    'nameSpace'       type:   uri         use: optional

     The equivalent MPEG21 DIDL element is 'DESCRIPTOR'.

     ============================================================-->
    <xsd:element name="desc" type="didl-lite:descType"/>
    <xsd:complexType name="descType">
        <xsd:annotation>
            <xsd:documentation> A 'desc' element identifies a descriptor.  A descriptor is intended to contain a
block of metadata.  A bio of a music artist is an example use of 'desc'.  A 'desc' element may possess child
elements from any namespace except the DIDL-Lite namespace.  A value for 'id' is required.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:any namespace="##other"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
        <xsd:attribute name="type" type="xsd:string"/>
        <xsd:attribute name="nameSpace" type="xsd:anyURI" use="required"/>
    </xsd:complexType>
</xsd:schema>
```

----------(cut here) ----------

6. Appendix B - AV Working Committee Extended Properties

The table below lists all properties defined by the AV Working Committee . In addition, a number of Dublin Core properties are listed (see Dublin Core Metadata Elements, Version 1.1 (http://dublincore.org/documents/dces), where applicable.

Property Name	Namespace	Property Type	Multiple Values	Property Description
Base properties				
(object)@id	DIDL-Lite	string	no	An identifier for the object. . The value of each object id property should be unique with respect to the Content Directory.
title	Dublin Core	string	no	Name of the object
creator	Dublin Core	string	no	Primary content creator or owner of the object
res	DIDL-Lite	URI	yes	Resource, typically a media file, associated with the object. Values must be properly escaped URIs as described in [RFC 2396].
class	upnp	string	no	Class of the object.
class@name	upnp	string	no	Friendly name for the class of the object. This should not be used for class based searches as it is not guaranteed to be unique or consistent across content items of the same class
container@searchable	DIDL-Lite	boolean	no	When *true*, the ability to perform a Search action under a **container** is enabled, otherwise a Search under that container will return no results. The default value of this attribute when if it is absent on a container is false.

Property Name	Namespace	Property Type	Multiple Values	Property Description
searchClass	upnp	string	yes	Search class of the associated **container** object. **If *(object)*@searchable = true, then** • If no searchClass elements are specified, then Search can return any match • If searchClass elements are specified then, the Search can return only matches from the classes specified in the searchClass tags • searchClass is optional • searchClass is always CDS determined • searchClass semantics are per container, there is no parent-child relationship, they only apply to searches started from that **container**.
searchClass@includeDerived	upnp	boolean	no	This is a required attribute of searchClass and indicates that the class specified also includes derived classes
searchClass@name	upnp	string	no	This is an optional attribute of searchClass and indicates a friendly name for the class

Property Name	Namespace	Property Type	Multiple Values	Property Description
createClass	upnp	string	yes	Create class of the associated **container** object. **If restricted = false, then** • If no createClass elements are specified, then CreateObject can create any class of object under the container • If createClass elements are specified then, CreateObject can only create classes of objects specified in the createClass tags • createClass is optional • createClass semantics are per **container**, there is no parent-child relationship, they only apply to CreateObject actions in that **container**
createClass@includeDerived	upnp	boolean	no	This is a required attribute of createClass, and indicates that the class specified also includes derived classes
createClass@name	upnp	string	no	This is an optional attribute of createClass and indicates a friendly name for the class
(object)@parentID	DIDL-Lite	string	no	id property of object's parent. The parentID of the Content Directory 'root' container must be set to the reserved value of "-1". No other parentID attribute of any other Content Directory object may take this value.
item@refID	DIDL-Lite	string	no	id property of the item being referred to.
object@restricted	DIDL-Lite	boolean	no	When *true*, ability to modify a given object is confined to the Content Directory. Control point metadata write access is disabled.

Property Name	Namespace	Property Type	Multiple Values	Property Description
writeStatus	UPnP	string	no	When present, controls the modifiability of the resources of a given object. Ability of a Control Point to change 'writeStatus' of a given resource(s) is implementation dependent. Allowed values are: *WRITABLE*, *PROTECTED*, *NOT_WRITABLE*, *UNKNOWN*, *MIXED*.
container@childCount	DIDL-Lite	integer	no	Child count for the object. Applies to containers only.
People involved				
artist	upnp	string	yes	Name of an artist
artist@role	upnp	string	no	Role of the artist in the work
actor	upnp	string	yes	Name of an actor appearing in a video item
actor@role	upnp	string	no	Role of the actor in the work
author	upnp	string	yes	Name of an author of a text item
author@role	upnp	string	no	Role of the author in the work
producer	upnp	string	yes	Name of producer of e.g., a movie or CD
director	upnp	string	yes	Name of the director of the video content item (e.g., the movie)
publisher	Dublin Core	string	yes	http://dublincore.org/documents/dces
contributor	Dublin Core	string	yes	http://dublincore.org/documents/dces It is recommended that **contributor** includes the name of the primary content creator or owner (DublinCore 'creator' property)
Links to containers, by container 'title'				
genre	upnp	string, not standardized by the Content Directory Service	yes	Name of the genre to which an object belongs
album	upnp	string	yes	Title of the album to which the item elongs.

Property Name	Namespace	Property Type	Multiple Values	Property Description
playlist	upnp	string	yes	Name of a playlist to which the item belongs
Resource encoding characteristics				
res@size	DIDL-Lite	unsigned long	no	Size in bytes of the resource
res@duration	DIDL-Lite	The form of the duration string is: H+:MM:SS[.F+] , or H+:MM:SS[.F0/F1] **where :** **H+** : number of digits (including no digits) to indicate elapsed hours, **MM** : exactly 2 digits to indicate minutes (00 to 59), **SS** : exactly 2 digits to indicate seconds (00 to 59), **F+** : any number of digits (including no digits) to indicate fractions of seconds, **F0/F1** : a fraction, with F0 and F1 at least one digit long, and F0 < F1. The string may be preceded by an optional + or – sign, and the decimal point itself may be omitted if there are no fractional second digits.	no	Time duration of the playback of the resource, at normal speed

Property Name	Namespace	Property Type	Multiple Values	Property Description
res@bitrate	DIDL-Lite	unsigned integer	no	Bitrate in bytes/seconds of the encoding of the resource
res@sampleFrequency	DIDL-Lite	unsigned integer	no	Sample frequency of the audio in HZ.
res@bitsPerSample	DIDL-Lite	unsigned integer	no	Encoding characteristic of the resource
res@nrAudioChannels	DIDL-Lite	unsigned integer	no	Number of audio channels, e.g., 1 for mono, 2 for stereo, 6 for Dolby Surround
res@resolution	DIDL-Lite	pattern string	no	XxY resolution of the resource in pixels (typically image item or video item). String pattern is of the form: [0-9]+x[0-9]+ (one or more digits,'x', followed by one or more digits).
res@colorDepth	DIDL-Lite	unsigned integer	no	Encoding characteristic of the resource.
res@protocolInfo	DIDL-Lite	string	no	Astring that identifies the recommended HTTP protocol for transmitting the resource (see also UPnP A/V Conection Manager Service template, section 2.5.2). If not present, then the content has not yet been fully imported by CDS and is not yet accesible for playback purposes.
res@protection	DIDL-Lite	string, not standardized by the CDS	no	Some identification of a protection system used for the resource .
res@importUri	DIDL-Lite	URI	no	URI via which the resource can be imported to the CDS via **ImportResource()** or HTTP POST.
Associated resources				
albumArtURI	upnp	URI	yes	Reference to album art. Values must be properly escaped URIs as described in [RFC 2396].
artistDiscographyURI	upnp	URI	no	Reference to artist discography. Values must be properly escaped URIs as described in [RFC 2396].

Property Name	Namespace	Property Type	Multiple Values	Property Description
lyricsURI	upnp	URI	no	Reference to lyrics of the song or of the whole album. Values must be properly escaped URIs as described in [RFC 2396].
relation	Dublin Core	URI	yes	http://dublincore.org/documents/dces. Values must be properly escaped URIs as described in [RFC 2396].
Storage-related				
storageTotal	upnp	signed long	no	Total capacity, in bytes, of the storage represented by the container. Value –1 is reserved to indicate that the capacity is 'unknown'
storageUsed	upnp	signed long	no	Combined space, in bytes, used by all the objects held in the storage represented by the container Value –1 is reserved to indicate that the space is 'unknown'.
storageFree	upnp	signed long	no	Total free capacity, in bytes, of the storage represented by the container Value –1 is reserved to indicate that the capacity is 'unknown'.
storageMaxPartition	upnp	signed long	no	Largest amount of space, in bytes, available for storing a single resource in the container. Value –1 is reserved to indicate that the amount of space is 'unknown'.

Property Name	Namespace	Property Type	Multiple Values	Property Description
storageMedium	upnp	string	no	Indicates the type of storage medium used for the content. Potentially useful for user-interface purposes. Allowed values are: *"UNKNOWN"*, *"DV"*, *"MINI-DV"*, *"VHS"*, *"W-VHS"*, *"S-VHS"*, *"D-VHS"*, *"VHSC"*, *"VIDEO8"*, *"HI8"*, *"CD-ROM"*, *"CD-DA"*, *"CD-R"*, *"CD-RW"*, *"VIDEO-CD"*, *"SACD"*, *"MD-AUDIO"*, *"MD-PICTURE"*, *"DVD-ROM"*, *"DVD-VIDEO"*, *"DVD-R"*, *"DVD+RW"*, *"DVD-RW"*, *"DVD-RAM"*, *"DVD-AUDIO"*. *"DAT"*, *"LD"*, *"HDD"*
General description, mainly for UI purposes				
description	Dublin Core	string	no	http://dublincore.org/documents/dces
longDescription	upnp	string	no	A few lines of description of the content item (longer than DublinCore's description element

Property Name	Namespace	Property Type	Multiple Values	Property Description
icon	upnp	URI	no	Some icon that a control point can use in its UI to display the content, e.g. a CNN logo for a Tuner channel. Recommend same format as the icon element in the UPnP device description document schema. (PNG). Values must be properly escaped URIs as described in [RFC 2396].
region	upnp	string, not standardized by the Content Directory Service.	no	Some identification of the region, associated with the 'source' of the object, e.g. "US", "Latin America", "Seattlle".
rating	upnp	string, not standardized by the Content Directory Service.	no	Rating of the object's resource, for 'parental control' filtering purposes, such as "R", "PG-13", "X", etc.,.
rights	Dublin Core	string, not standardized by the Content Directory Service.	yes	http://dublincore.org/documents/dces
date	Dublin Core	string	no	ISO 8601, of the form *"YYYY-MM-DD"*,
language	Dublin Core	string	no	RFC 1766, e.g. of the form *"en-US"*.
Radio broadcast				
radioCallSign	upnp	string, not standardized by the Content Directory Service	no	Radio station call sign, e.g. "KSJO"
radioStationID	upnp	string, not standardized by the Content Directory Service	no	Some identification, e.g. "107.7", broadcast frequency of the radio station
radioBand	upnp	string	no	Radio station frequency band. Recommended values are *"AM", "FM", "Shortwave", "Internet", "Satellite"*. Vendor's may extend this list.
Video broadcast				

Property Name	Namespace	Property Type	Multiple Values	Property Description
channelNr	upnp	integer	no	Used for identification of tuner channels themselves, or information associated with a piece of recorded content
channelName	upnp	string, not standardized by the Content Directory Service.	no	Used for identification of channels themselves, or information associated with a piece of recorded content
scheduledStartTime	upnp	string	yes	ISO 8601, of the form *"yyyy-mm-ddThh:mm:ss"*. Used to indicate the start time of a schedule program, indented for use by tuners
scheduledEndTime	upnp	string	yes	ISO 8601, of the form *"yyyy-mm-ddThh:mm:ss"*. Used to indicate the end time of a scheduled program, indented for use by tuners
Miscellaneous				
DVDRegionCode	upnp	integer	no	Region code of the DVD disc
originalTrackNumber	upnp	integer	no	Original track number on an audio CD or other medium
toc	upnp	string	no	Identifier of an audio CD.
userAnnotation	upnp	string	yes	General-purpose tag where a user can annotate an object with some user-specific information

Note: The following DublinCore elements will **not** be used as AV Working Committee Extended Properties:

- type
- subject
- format
- identifier
- source
- coverage
-

7. Appendix C - AV Working Committee Class Definitions

In addition to classes **object**, **item** and **container** (described in Section 2.4), the AV Working Committee has defined a number of class descriptions that are derived from either the **item** or **container** classes. Figure 1and Figure 2 below shows the hierarchy of these AV Working Committee-defined class definitions.

For each class in these figures, the required and most relevant optional properties that apply to instances of the class are listed. Any device that adds a property whose description matches one of our property descriptions must use our property name. In addition, any device that uses a property name from the Content Directory Service specification must use it with the same semantics as our description of that property. Content Directory Service providers are free to add other properties than those defined in Appendix B to instances of one of the classes below, from any kind of XML namespace

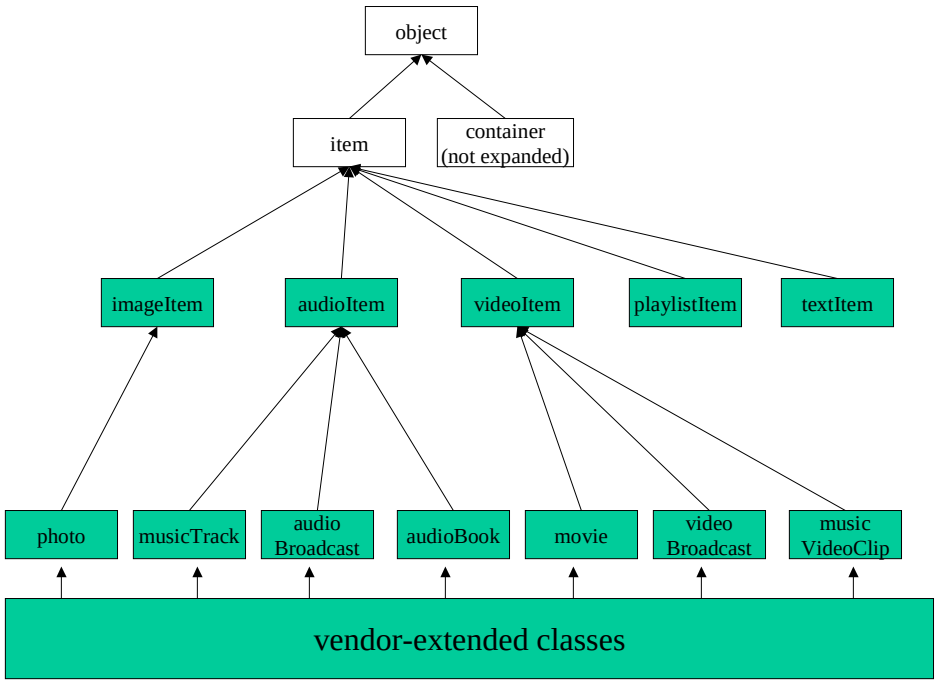

Figure 1: Class hierarchy for 'item' base class.

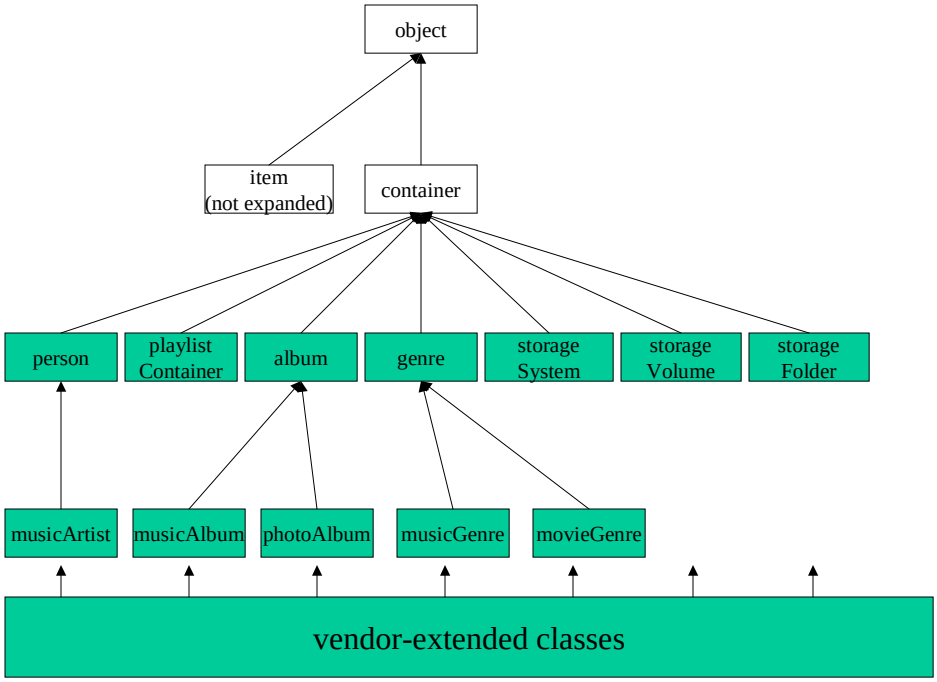

Figure 2: Class hierarchy for 'container' base class.

7.1. audioItem : item

An ‘audioItem’ instance represents a piece of content that, when rendered, generates some audio[2]. It is atomic in the sense that it does not contain other objects in the ContentDirectory. It typically has at least 1 <**res**> element. The class is derived from ‘item’ and adds the following properties:

Property Name	NameSpace	Required	Remarks
genre	upnp	No	
description	dc	No	
longDescription	upnp	No	
publisher	dc	No	
language	dc	No	
relation	dc	No	
rights	dc	No	

[2] Movies, TV broadcasts, etc., that also contain an audio track are excluded from this definition; those objects should be classified under ‘videoItem’.

7.1.1. musicTrack : audioItem

A ‘musicTrack’ instance is a discrete piece of audio that should be interpreted as music (as opposed to, for example, a news broadcast or an audio book). It typically has at least 1 <**res**> element. The class is derived from ‘audioItem’ and adds the following properties:

Property Name	NameSpace	Required	Remarks
artist	upnp	No	
album	upnp	No	
originalTrackNumber	upnp	No	
playlist	upnp	No	
storageMedium	upnp	No	
contributor	dc	No	
date	dc	No	

7.1.2. audioBroadcast : audioItem

An ‘audioBroadcast’ instance is a continuus stream of audio that should be interpreted as an audio broadcast (as opposed to, for example, a song or an audio book). It typically has at least 1 <**res**> element. The class is derived from ‘audioItem’ and adds the following properties:

Property Name	NameSpace	Required	Remarks
region	upnp	No	
radioCallSign	upnp	No	
radioStationID	upnp	No	
radioBand	upnp	No	
channelNr	upnp	No	

7.1.3. audioBook : audioItem

An ‘audioBook’ instance is a discrete piece of audio that should be interpreted as a book (as opposed to, for example, a news broadcast or a song). It typically has at least 1 <**res**> element. The class is derived from ‘audioItem’ and adds the following properties:

Property Name	NameSpace	Required	Remarks
storageMedium	upnp	No	
producer	upnp	No	
contributor	dc	No	
date	dc	No	

7.2. videoItem : item

A 'videoItem' instance represents a piece of content that, when rendered, generates some video. It is atomic in the sense that it does not contain other objects in the ContentDirectory. It typically has at least 1 <**res**> element. The class is derived from 'item' and adds the following properties:

Property Name	NameSpace	Required	Remarks
genre	upnp	No	
longDescription	upnp	No	
producer	upnp	No	
rating	upnp	No	
actor	upnp	No	
director	upnp	No	
description	dc	No	
publisher	dc	No	
language	dc	No	
relation	dc	No	

7.2.1. movie : videoItem

A 'movie' instance is a discrete piece of video that should be interpreted as a movie (as opposed to, for example, a continuus TV broadcast or a music video clip). It typically has at least 1 <**res**> element. The class is derived from 'videoItem' and adds the following properties:

Property Name	NameSpace	Required	Remarks
storageMedium	upnp	No	
DVDRegionCode	upnp	No	
channelName	upnp	No	
scheduledStartTime	upnp	No	
scheduledEndTime	upnp	No	

7.2.2. videoBroadcast: videoItem

A tvStation represents an (Internet or conventional) tv station, and is derived from the cdsItemContainer base class. A tv channel can contain other items representing the broadcast schedule of the channel, or it can be present as an atomatic item, for example when no schedule information is known. In the latter case, the 'childCount' attribute of the <container> tag will simply be '0'. The tvStation class identifies the following properties:

A 'videoBroadcast' instance is a continuus stream of video that should be interpreted as a broadcast (e.g., a convential TV channel or a Webcast). It typically has at least 1 <**res**> element. The class is derived from 'videoItem' and adds the following properties:

Property Name	NameSpace	Required	Remarks
icon	upnp	No	

Property Name	NameSpace	Required	Remarks
region	upnp	No	
channelNr	upnp	No	

7.2.3. musicVideoClip: videoItem

A 'musicVideoClip' instance is a discrete piece of video that should be interpreted as a clip supporting a song (as opposed to, for example, a continuus TV broadcast or a movie). It typically has at least 1 <**res**> element. The class is derived from 'videoItem' and adds the following properties:

Property Name	NameSpace	Required	Remarks
artist	upnp	No	
storageMedium	upnp	No	
album	upnp	No	
scheduledStartTime	upnp	No	
scheduledStopTime	upnp	No	
director	upnp	No	
contributor	dc	No	
date	dc	No	

7.3. imageItem : item

An 'imageItem' instance represents a piece of content that, when rendered, generates some still image. It is atomic in the sense that it does not contain other objects in the ContentDirectory. It typically has at least 1 <**res**> element. The class is derived from 'item' and adds the following properties:

Property Name	NameSpace	Required	Remarks
longDescription	upnp	No	
storageMedium	upnp	No	
rating	upnp	No	
description	dc	No	
publisher	dc	No	
date	dc	No	
rights	dc	No	

7.3.1. photo: imageItem

A 'photo' instance is an image that should be interpreted as a photo (as opposed to, for example, an icon). It typically has at least 1 <**res**> element. The class is derived from 'imageItem' and adds the following properties:

Property Name	NameSpace	Required	Remarks
album	upnp	No	

7.4. playlistItem : item

A 'playlistItem' instance represents a playable sequence of resources. It is different from 'musicAlbum' in the sense that a 'playlistItem' may contain a mix of audio, video and images and is typically created by a user, while an 'album' is typically a fixed published sequence of songs (e.g., an audio CD). A 'playlistItem' item is required to have a <**res**> element for playback of the whole sequence. This <**res**> element is a reference to a playlist file authored outside of the ContentDirectory service (e.g., an external M3U file). Rendering the 'playlistItem' has the semantics defined by the playlist's resource (e.g., ordering, transition effects, etc.). The class is derived from 'item' and adds the following properties:

Property Name	NameSpace	Required	Remarks
artist	upnp	No	Applies to the resources inside the playlist. May be multiple-valued to express multiple artists.
genre	upnp	No	Applies to the playlist as a whole, not any individual resources that it might reference.
longDescription	upnp	No	
storageMedium	upnp	No	Applies to the storageMedium of the playlist file itself, not the resources that the playlist file might reference.
description	dc	No	
date	dc	No	Applies to the creation date of the playlist file itself, not the resources that it might reference.
language	dc	No	Applies to the resources inside the playlist. May be multiple-valued to express multiple languages.

7.5. textItem : item

A 'textItem' instance represents a piece of content that, when rendered, is readable as text. It is atomic in the sense that it does not contain other objects in the ContentDirectory. It typically has at least 1 <**res**> element. The class is derived from 'item' and adds the following properties:

Property Name	NameSpace	Required	Remarks
author	upnp	No	
protection	upnp	No	
longDescription	upnp	No	
storageMedium	upnp	No	
rating	upnp	No	
description	dc	No	
publisher	dc	No	
contributor	dc	No	
date	dc	No	
relation	dc	No	
language	dc	No	
rights	dc	No	

7.6. album : container

An 'album' instance represents an ordered collection of 'objects'. It may have a <**res**> element for playback of the whole album, or not. In the first case, rendering the album has the semantics of rendering each object in sequence. In the latter case, a control point needs to separately initiate rendering for each child object. The class is derived from 'container' and adds the following properties:

Property Name	NameSpace	Required	Remarks
storageMedium	upnp	No	
longDescription	dc	No	
description	dc	No	
publisher	dc	No	
contributor	dc	No	
date	dc	No	
relation	dc	No	
rights	dc	No	

7.6.1. musicAlbum : album

A 'musicAlbum' instance is an 'album' which contains items of class 'musicTrack' (see Section 7.1.1) or 'sub'-albums of class 'musicAlbum'. It can be used to model, for example, an audio-CD. The class is derived from 'album' and adds the following properties:

Property Name	NameSpace	Required	Remarks
artist	upnp	No	
genre	upnp	No	
producer	upnp	No	
albumArtURI	upnp	No	
toc	upnp	No	

7.6.2. photoAlbum : album

A ‘photoAlbum’ instance is an ‘album’ which contains items of class ‘photo (see Section 7.3.1) or ‘sub’-albums of class ‘photoAlbum’. The class is derived from ‘album’ and adds the following properties:

7.7. genre : container

A ‘genre’ instance represents an unordered collection of ‘objects’ that “belong” to the genre, in a loose sense. It may have a <**res**> element for playback of all elements of the genre, or not. In the first case, rendering the genre has the semantics of rendering each object in the collection, in some order. In the latter case, a control point needs to separately initiate rendering for each child object. A ‘genre’ container can contain objects of class ‘person’, ‘album’, ‘audioItem’, ‘videoItem’ or “sub”-genres of the same class (e.g. ‘Rock’ contains ‘Alternative Rock’). Which classes of objects a ‘genre’ contains in a ContentDirectory implementation is device-dependent. The class is derived from ‘container’ and adds the following properties:

Property Name	NameSpace	Required	Remarks
longDescription	upnp	No	
description	dc	No	

7.7.1. musicGenre : genre

A ‘musicGenre’ instance is a ‘genre’ which should be interpreted as a “style of music”. A ‘musicGenre’ container can contain objects of class ‘musicArtist, ‘musicAlbum’, ‘audioItem’ or “sub”-music genres of the same class (e.g. ‘Rock’ contains ‘Alternative Rock’). Which classes of objects a ‘musicGenre’ contains in a ContentDirectory implementation is device-dependent. The class is derived from ‘genre’ and currently does not add any properties.

7.7.2. movieGenre : genre

A ‘movieGenre’ instance is a ‘genre’ which should be interpreted as a “style of movies”. A ‘movieGenre’ container can contain objects of class ‘people’, ‘videoItem’ or “sub”-movie genres of the same class (e.g. ‘Western’ contains ‘Spaghetti Western’). Which classes of objects a ‘movieGenre’ contains in a ContentDirectory implementation is device-dependent. The class is derived from ‘genre’ and currently does not add any properties.

7.8. playlistContainer : container

A ‘playlistContainer’ instance represents a collection of ‘objects’. It is different from ‘musicAlbum’ in the sense that a ‘playlistContainer’ may contain a mix of audio, video and images and is typically created by a user, while an ‘album’ is typically a fixed published sequence of songs (e.g., an audio CD). A ‘playlistContainer’ may have a <**res**> element for playback of the whole playlist or not. This <**res**> element may be a dynamically created playlist resource, as described in Section 2.8.9.2, or a reference to a playlist file authored outside of the ContentDirectory service (e.g., an external M3U file); this is device-dependent. In any case, rendering the playlist has the semantics defined by the playlist resource (e.g., ordering, transition effects, etc.). If the ‘playlistContainer’ has no <**res**> element, a control point needs to separately initiate rendering for each child object, typically in the order the children are received from a ‘Browse’ action. The class is derived from ‘container’ and adds the following properties:

Property Name	NameSpace	Required	Remarks
artist	upnp	No	
genre	upnp	No	
longDescription	upnp	No	
producer	upnp	No	
storageMedium	upnp	No	
description	dc	No	
contributor	dc	No	
date	dc	No	
language	dc	No	
rights	dc	No	

7.9. person : container

A ‘person’ instance represents an unordered collection of ‘objects’ that “belong” to the people, in a loose sense. It may have a <**res**> element for playback of all elements belongin to the person, or not. In the first case, rendering the ‘person’ has the semantics of rendering each object in the collection, in some order. In the latter case, a control point needs to separately initiate rendering for each child object. A ‘person’ container can contain objects of classes ‘album’, ‘item’, or ‘playlist’. Which classes of objects a ‘person’ contains in a ContentDirectory implementation is device-dependent. The class is derived from ‘container’ and adds the following properties:

Property Name	NameSpace	Required	Remarks
language	dc	No	May be multiple valued.

7.9.1. musicArtist : person

A ‘musicArtist’ instance is a ‘person’ which should be interpreted as a music artist. A ‘musicArtist’ container can contain objects of class ‘musicAlbum’, ‘musicTrack’ or ‘musicVideoClip’. Which classes

of objects a ‘musicArtist’ contains in a ContentDirectory implementation is device-dependent. The class is derived from ‘person’ and adds the following properties:

Property Name	NameSpace	Required	Remarks
genre	upnp	No	
artistDiscographyURI	upnp	No	

7.10. storageSystem : container

A ‘storageSystem’ instance represents a potentially heterogeneous collection of storage media. A ‘storageSystem’ may contain other objects, including all types of storage containers. A ‘storageSystem’ may only be a child of the root container or another ‘storageSystem’ container. Examples of ‘storageSystem’ instances are

- a CD Jukebox
- a Hard Disk Drive plus a CD in a combo device
- a single CD

The ‘storageSystem’ class identifies the following properties:

Property Name	Name Space	Required	Remarks
storageTotal	upnp	Yes	
storageUsed	upnp	Yes	
storageFree	upnp	Yes	
storageMaxPartition	upnp	Yes	
storageMedium	upnp	Yes	

Regarding the writeStatus property of a StorageSystem (see ‘object’ class definition), if there are content items/containers in a storageSystem that are not contained within any storageVolume, consider all of these “free” items to be contained in a single virtual storageVolume. For purposes of establishing the writeStatus of a storageSystem, this virtual volume is treated like all the other “real” storageVolumes in the storageSystem.

If every storageVolume in a storageSystem has the same writeStatus, the value of writeStatus for the storageSystem must also be set to that value.

If any two storageVolumes in a storageSystem have different values for writeStatus, the value of writeStatus for the storageSystem must be set to “MIXED”.

7.11. storageVolume : container

A ‘storageVolume’ instance represents all, or a partition of, some physical storage unit of a single type (as indicated by the ‘storageMedium’ property). The storage volume may be writable or not, indicating whether new items can be created as children of the volume. A storageVolume may contain other object, except a ‘storageSystem’ or another ‘storageVolume’. A ‘storageVolume’ may only be a child of the root container or a ‘storageSystem’ container. Examples of ‘storageVolume’ instances are

- a Hard Disk Drive

- a partition on a Hard Disk Drive
- a CD-Audio disc
- a Flash memory card

The ‘storageVolume’ class identifies the following properties:

Property Name	NameSpace	Required	Remarks
storageTotal	upnp	Yes	
storageUsed	upnp	Yes	
storageFree	upnp	Yes	
storageMedium	upnp	Yes	

7.12. storageFolder : container

A ‘storageFolder’ instance represents a collection of objects stored on some storage medium. The storage folder may be writable or not, indicating whether new items can be created as children of the folder or whether existing child items can be removed. If the parent storage container is not writable, then the ‘storageFolder’ itself cannot be writable. A ‘storageFolder’ may contain other objects, except a ‘storageSystem’ or a ‘storageVolume’. A ‘storageFolder’ may only be a child of the root container or another storage container. Examples of ‘storageFolder’ instances are

- a directory on a Hard Disk Drive
- a directory on CD-Rom, etc.

The ‘storageFolder’ class identifies the following properties:

Property Name	NameSpace	Required	Remarks
storageUsed	upnp	Yes	