ISO/IEC 29341-14-12

Edition 1.0 2011-09

INTERNATIONAL STANDARD

colour inside

Information technology – UPnP device architecture –
Part 14-12: Audio, Video Device Control Protocol – Level 3 – Audio Video Content Directory Service

ISO/IEC 29341-14-12:2011(E)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

ISO/IEC 29341-14-12

Edition 1.0 2011-09

INTERNATIONAL STANDARD

colour inside

Information technology – UPnP device architecture – Part 14-12: Audio, Video Device Control Protocol – Level 3 – Audio Video Content Directory Service

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE XC

ICS 35.200

ISBN 978-2-88912-658-3

CONTENTS

INFORMATION TECHNOLOGY – UPNP DEVICE ARCHITECTURE –

Part 14-12: Audio, Video Device Control Protocol – Level 3 – Audio Video Content Directory Service

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.

2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.

4) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IEC publication and the corresponding national or regional publication should be clearly indicated in the latter.

6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC publication.

7) All users should ensure that they have the latest edition of this publication.

8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.

9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

10) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 29341-14-12 was prepared by UPnP Forum Steering committee[1], was adopted, under the fast track procedure, by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

The list of all currently available parts of the ISO/IEC 29341 series, under the general title *Information technology – UPnP device architecture*, can be found on the IEC web site.

This International Standard has been approved by vote of the member bodies, and the voting results may be obtained from the address given on the second title page.

---

1 UPnP Forum Steering committee, UPnP Forum, 3855 SW 153rd Drive, Beaverton, Oregon 97006 USA. See also "Introduction".

IMPORTANT – The "colour inside" logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this publication using a colour printer.

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 ContentDirectory service.

1.1 Introduction

Many devices within the home network contain various types of content that other devices would like to access (for example, music, videos, still images, etc). As an example, a MediaServer 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 MediaServer, select a specific one, and cause it to be played on an appropriate rendering device (for example, 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 (that is: the homeowner won't have to walk over to the server device). In order to enable this capability, the server 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 ContentDirectory service.

The ContentDirectory service additionally provides a lookup/storage service that allows clients (for example, UI devices) to locate (and possibly store) individual objects (for example, 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 ContentDirectory service. For devices that contain multiple types of content (for example, MP3, MPEG2, JPEG, etc.), a single instance of the ContentDirectory service can be used to enumerate all objects, regardless of their type.

1.2 Notation

- In this document, features are described as Required, Recommended, or Optional as follows:

  The keywords "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this specification are to be interpreted as described in [RFC 2119].

  In addition, the following keywords are used in this specification:

  PROHIBITED – The definition or behavior is prohibited by this specification. Opposite of REQUIRED.

  CONDITIONALLY REQUIRED – The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is REQUIRED, otherwise it is PROHIBITED.

  CONDITIONALLY OPTIONAL – The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is OPTIONAL, otherwise it is PROHIBITED.

  These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

- Strings that are to be taken literally are enclosed in "double quotes".

- Words that are emphasized are printed in *italic*.
- Keywords that are defined by the UPnP AV Working Committee are printed using the *forum* character style.
- Keywords that are defined by the UPnP Device Architecture specification are printed using the **arch** character style [DEVICE].
- A double colon delimiter, "::", signifies a hierarchical parent-child (parent::child) relationship between the two objects separated by the double colon. This delimiter is used in multiple contexts, for example: Service::Action(), Action()::Argument, parentProperty::childProperty.

1.2.1 Data Types

This specification uses data type definitions from two different sources. The UPnP Device Architecture defined data types are used to define state variable and action argument data types [DEVICE]. The XML Schema namespace is used to define property data types [XML SCHEMA-2].

For UPnP Device Architecture defined **boolean** data types, it is strongly RECOMMENDED to use the value "**0**" for false, and the value "**1**" for true. However, when used as input arguments, the values "**false**", "**no**", "**true**", "**yes**" may also be encountered and MUST be accepted. Nevertheless, it is strongly RECOMMENDED that all **boolean** state variables and output arguments be represented as "**0**" and "**1**".

For XML Schema defined Boolean data types, it is strongly RECOMMENDED to use the value "*0*" for false, and the value "*1*" for true. However, when used as input properties, the values "*false*", "*true*" may also be encountered and MUST be accepted. Nevertheless, it is strongly RECOMMENDED that all Boolean properties be represented as "*0*" and "*1*".

1.2.2 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 subclause 1.3.1, "Comma Separated Value (CSV) Lists") and property values in search criteria strings. Escaping conventions use the backslash character, "\" (character code U+005C), 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
   3) not escaped in CSV lists
   4) not escaped in search strings when it appears as the start or end delimiter of a property value
   5) represented as "\"" in search strings when it appears as a character that is part of the property value

1.2.3 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].

1.2.3.1 Typographic conventions for EBNF

`Non-terminal` symbols are unquoted sequences of characters from the set of English upper and lower case letters, the digits "`0`" through "`9`", and the hyphen ("`-`"). Character sequences between `'single quotes'` are terminal strings and MUST appear literally in valid strings. 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 valid strings. White space usage in valid strings is described explicitly in the EBNF. Finally, the EBNF uses the following operators:

Table 1-1 — EBNF 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.
[ ]	**optional –** the expression between the brackets is optional.
( )	**grouping –** groups the expressions between the parentheses.
-	**character range** – represents all characters between the left and right character operands inclusively.

1.3 Derived Data Types

This subclause defines a derived data type that is represented as a string data type with special syntax. This specification uses string data type definitions that originate from two different sources. The UPnP Device Architecture defined **string** data type is used to define state variable and action argument **string** data types. The XML Schema namespace is used to define property xsd:string data types. The following definition applies to both string data types.

1.3.1 Comma Separated Value (CSV) Lists

The UPnP AV services use state variables, action arguments and properties that represent lists – or one-dimensional arrays – of values. 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** or xsd:string and denoted by CSV (*x*), where *x* is the type of the individual values. The data type of a heterogeneous list is also **string** or xsd:string and denoted by CSV (*x, y, z*), where *x*, *y* and *z* are the types of the 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 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** or xsd:string and denoted by 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 **string** type (for state variables and action arguments) or xsd:string type (for properties).
- Commas separate values within a list.

- Integer values are represented in CSVs with the same syntax as the integer data type specified in [DEVICE] (that is: optional leading sign, optional leading zeroes, numeric US-ASCII)
- Boolean values are represented in state variable and action argument 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**.
- Boolean values are represented in property CSVs as either "*0*" for false or "*1*" for true. These values are a subset of the defined Boolean data type values specified in [XML SCHEMA-2]: 0, false, 1, true.
- Escaping conventions for the comma and backslash characters are defined in subclause 1.2.2, "Strings Embedded in Other Strings".
- 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 1-2 — CSV Examples

Type refinement of string	Value	Comments
CSV (**string**) or CSV (xsd:string)	"+artist,-date"	List of 2 property sort criteria.
CSV (**int**) or CSV (xsd:integer)	"1,-5,006,0,+7"	List of 5 integers.
CSV (**boolean**) or CSV (xsd:Boolean)	"0,1,1,0"	List of 4 booleans
CSV (**string**) or CSV (xsd:string)	"Smith\, Fred,Jones\, Davey"	List of 2 names, "Smith, Fred" and "Jones, Davey"
CSV (**i4**,**string**,**ui2**) or CSV (xsd:int, xsd:string, xsd:unsignedShort)	"-29837, string with leading blanks,0"	Note that the second value is " string with leading blanks"
CSV (**i4**) or CSV (xsd:int)	"3, 4"	Illegal CSV. White space is not allowed as part of an integer value.
CSV (**string**) or CSV (xsd:string)	",,"	List of 3 empty string values
CSV (heterogeneous)	"Alice,Marketing,5,Sue,R&D,21,Dave,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** or a name xsd:string, a department xsd:string and years-of-service xsd:unsignedShort.

1.4 Management of XML Namespaces in Standardized DCPs

UPnP specifications make extensive use of XML namespaces. This allows separate DCPs, and even separate components of an individual DCP, to be designed independently and still avoid name collisions when they share XML documents. Every name in an XML document belongs to exactly one namespace. In documents, XML names appear in one of two forms: qualified or unqualified. An unqualified name (or no-colon-name) contains no colon (":") characters. An unqualified name belongs to the document's default namespace. A qualified name is two no-colon-names separated by one colon character. The no-colon-name before the colon is the qualified name's namespace prefix, the no-colon-name after the colon is the qualified name's "local" name (meaning local to the namespace identified by the namespace prefix). Similarly, the unqualified name is a local name in the default namespace.

The formal name of a namespace is a URI. The namespace prefix used in an XML document is *not* the name of the namespace. The namespace name is, or should be, globally unique. It has a single definition that is accessible to anyone who uses the namespace. It has the same meaning anywhere that it is used, both inside and outside XML documents. The namespace prefix, however, in formal XML usage, is defined only in an XML document. It must be locally unique to the document. Any valid XML no-colon-name may be used. And, in formal XML usage, no two XML documents are ever required to use the same namespace prefix to refer to the same namespace. The creation and use of the namespace prefix was standardized by the W3C XML Committee in [XML-NMSP] strictly as a convenient local shorthand replacement for the full URI name of a namespace in individual documents.

All AV object properties are represented in XML by element and attribute names, therefore, all property names belong to an XML namespace.

For the same reason that namespace prefixes are convenient in XML documents, it is convenient in specification text to refer to namespaces using a namespace prefix. Therefore, this specification declares a “standard” prefix for all XML namespaces used herein. In addition, this specification expands the scope where these prefixes have meaning, beyond a single XML document, to all of its text, XML examples, and certain string-valued properties. This expansion of scope *does not* supercede XML rules for usage in documents, it only augments and complements them in important contexts that are out-of-scope for the XML specifications. For example, action arguments which refer to CDS properties, such as the *SearchCriteria* argument of the *Search()* action or the *Filter* argument of the *Browse()* action, MUST use the predefined namespace prefixes when referring to CDS properties (“upnp:”, “dc:”, etc).

All of the namespaces used in this specification are listed in the Tables “Namespace Definitions” and “Schema-related Information”. For each such namespace, Table 1-3, “Namespace Definitions” gives a brief description of it, its name (a URI) and its defined “standard” prefix name. Some namespaces included in these tables are not directly used or referenced in this document. They are included for completeness to accommodate those situations where this specification is used in conjunction with other UPnP specifications to construct a complete system of devices and services. For example, since the Scheduled Recording Service depends on and refers to the Content Directory Service, the predefined “srs:” namespace prefix is included. The individual specifications in such collections all use the same standard prefix. The standard prefixes are also used in Table 1-4, “Schema-related Information”, to cross-reference additional namespace information. This second table includes each namespace’s valid XML document root element(s) (if any), its schema file name, versioning information (to be discussed in more detail below), and a link to the entry in subclause 1.6, ”References” for its associated schema.

The normative definitions for these namespaces are the documents referenced in Table 1-3. The schemas are designed to support these definitions for both human understanding and as test tools. However, limitations of the XML Schema language itself make it difficult for the UPnP-defined schemas to accurately represent all details of the namespace definitions. As a result, the schemas will validate many XML documents that are not valid according to the specifications.

The Working Committee expects to continue refining these schemas after specification release to reduce the number of documents that are validated by the schemas while violating the specifications, but the schemas will still be informative, supporting documents. Some schemas might become normative in future versions of the specifications.

Table 1-3 — Namespace Definitions

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
AV Working Committee defined namespaces			

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
av	urn:schemas-upnp-org:av:av	Common data types for use in AV schemas	[AV-XSD]
avs	urn:schemas-upnp-org:av:avs	Common structures for use in AV schemas	[AVS-XSD]
avdt	urn:schemas-upnp-org:av:avdt	Datastructure Template	[AVDT]
avt-event	urn:schemas-upnp-org:metadata-1-0/AVT/	Evented *LastChange* state variable for AVTransport	[AVT]
didl-lite	urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/	Structure and metadata for ContentDirectory	[CDS]
cds-event	urn:schemas-upnp-org:av:cds-event	Evented *LastChange* state variable for ContentDirectory	[CDS]
rcs-event	urn:schemas-upnp-org:metadata-1-0/RCS/	Evented *LastChange* state variable for RenderingControl	[RCS]
srs	urn:schemas-upnp-org:av:srs	Metadata and structure for ScheduledRecording	[SRS]
srs-event	urn:schemas-upnp-org:av:srs-event	Evented *LastChange* state variable for ScheduledRecording	[SRS]
upnp	urn:schemas-upnp-org:metadata-1-0/upnp/	Metadata for ContentDirectory	[CDS]
Externally defined namespaces			
dc	http://purl.org/dc/elements/1.1/	Dublin Core	[DC-TERMS]
xsd	http://www.w3.org/2001/XMLSchema	XML Schema Language 1.0	[XML SCHEMA-1] [XML SCHEMA-2]
xsi	http://www.w3.org/2001/XMLSchema-instance	XML Schema Instance Document schema	clauses 2.6 & 3.2.7 of [XML SCHEMA-1]
xml	http://www.w3.org/XML/1998/namespace	The “xml” Namespace	[XML-NS]

Table 1-4 — Schema-related Information

Standard Name-space Prefix	Relative URI and File Name [1] • Form 1, Form 2, Form3	Valid Root Element(s)	Schema Reference
AV Working Committee Defined Namespaces			
av	av-v*n*-*yyyymmdd*.xsd av-v*n*.xsd av.xsd	*n/a*	[AV-XSD]

avs	avs-v*n*-*yyyymmdd*.xsd
	avs-vn.xsd
	avs.xsd
		`<Capabilities>`
		`<Features>`
		`<stateVariableValuePairs>`
			[AVS-XSD]

avdt	avdt-v*n*-*yyyymmdd*.xsd avdt-vn.xsd avdt.xsd	`<AVDT>`	[AVDT]
avt-event	avt-event-v*n*-*yyyymmdd*.xsd avt-event-vn.xsd avt-event.xsd	`<Event>`	[AVT-EVENT-XSD]
cds-event	cds-event-v*n*-*yyyymmdd*.xsd cds-event-vn.xsd cds-event.xsd	`<StateEvent>`	[CDS-EVENT-XSD]
didl-lite	didl-lite-v*n*-*yyyymmdd*.xsd didl-lite-vn.xsd didl-lite.xsd	`<DIDL-Lite>`	[DIDL-LITE-XSD]

Standard Name-space Prefix	Relative URI and File Name [1] • Form 1, Form 2, Form3	Valid Root Element(s)	Schema Reference
rcs-event	rcs-event-v*n-yyyymmdd*.xsd rcs-event-vn.xsd rcs-event.xsd	<Event>	[RCS-EVENT-XSD]
srs	srs-v*n-yyyymmdd*.xsd srs-vn.xsd srs.xsd	<srs>	[SRS-XSD]
srs-event	srs-event-v*n-yyyymmdd*.xsd srs-event-vn.xsd srs-event.xsd	<StateEvent>	[SRS-EVENT-XSD]
upnp	upnp-v*n-yyyymmdd*.xsd upnp-vn.xsd upnp.xsd	*n/a*	[UPNP-XSD]
Externally Defined Namespaces			
dc	*Absolute URL*: http://dublincore.org/schemas/xmls/simpledc20021212.xsd		[DC-XSD]
xsd	*n/a*	<schema>	[XMLSCHEMA-XSD]
xsi	*n/a*		*n/a*
xml	*n/a*		[XML-XSD]

[1] Absolute URIs are generated by prefixing the relative URIs with "http://www.upnp.org/schemas/av/"

1.4.1 Namespace Prefix Requirements

There are many occurrences in this specification of string data types that contain XML names (property names). These XML names in strings will not be processed under namespace-aware conditions. Therefore, all occurrences in instance documents of XML names in strings MUST use the standard namespace prefixes as declared in Table 1-3. In order to properly process the XML documents described herein, control points and devices MUST use namespace-aware XML processors [XML-NMSP] for both reading and writing. As allowed by [XML-NMSP], the namespace prefixes used in an instance document are at the sole discretion of the document creator. Therefore, the declared prefix for a namespace in a document MAY be different from the standard prefix. All devices MUST be able to correctly process any valid XML instance document even when it uses a non-standard prefix for ordinary XML names. However, it is strongly RECOMMENDED that all devices use these standard prefixes for all instance documents to avoid confusion on the part of both human and machine readers. These standard prefixes are used in all descriptive text and all XML examples in this and related UPnP specifications. Also, each individual specification may assume a default namespace for its descriptive text. In that case, names from that namespace may appear with no prefix.

The assumed default namespace, if any, for each UPnP AV specification is given in Table 1-5, “Default Namespaces for the AV Specifications”.

Note: all UPnP AV schemas declare attributes to be “unqualified”, so namespace prefixes are never used with AV Working Committee defined attribute names.

Table 1-5 — Default Namespaces for the AV Specifications

AV Specification Name	Default Namespace Prefix
AVTransport	avt-event
ConnectionManager	*n/a*
ContentDirectory	didl-lite
MediaRenderer	*n/a*

AV Specification Name	Default Namespace Prefix
MediaServer	*n/a*
RenderingControl	rcs-event
ScheduledRecording	srs

1.4.2 Namespace Names, Namespace Versioning and Schema Versioning

The UPnP AV service specifications define several data structures (such as state variables and action arguments) whose format is an XML instance document that must comply with one or more specific XML namespaces. Each namespace is uniquely identified by an assigned namespace name. The namespaces that are defined by the AV Working Committee MUST be named by a URN. See Table 1-3, “Namespace Definitions” for a current list of namespace names. Additionally, each namespace corresponds to an XML schema document that provides a machine-readable representation of the associated namespace to enable automated validation of the XML (state variable or action parameter) instance documents.

Within an XML schema and XML instance document, the name of each corresponding namespace appears as the value of an `xmlns` attribute within the root element. Each `xmlns` attribute also includes a namespace prefix that is associated with that namespace in order to disambiguate (a.k.a. qualify) element and attribute names that are defined within different namespaces. The schemas that correspond to the listed namespaces are identified by URI values that are listed in the `schemaLocation` attribute also within the root element. (See subclause 1.4.3, “Namespace Usage Examples”)

In order to enable both forward and backward compatibility, namespace names are permanently assigned and MUST NOT change even when a new version of a specification changes the definition of a namespace. However, all changes to a namespace definition MUST be backward-compatible. In other words, the updated definition of a namespace MUST NOT invalidate any XML documents that comply with an earlier definition of that same namespace. This means, for example, that a namespace MUST NOT be changed so that a new element or attribute is required. Although namespace names MUST NOT change, namespaces still have version numbers that reflect a specific set of definitional changes. Each time the definition of a namespace is changed, the namespace’s version number is incremented by one.

Each time a new namespace version is created, a new XML schema document (.xsd) is created and published so that the new namespace definition is represented in a machine-readable form. Since a XML schema document is just a representation of a namespace definition, translation errors can occur. Therefore, it is sometime necessary to re-release a published schema in order to correct typos or other namespace representation errors. In order to easily identify the potential multiplicity of schema releases for the same namespace, the URI of each released schema MUST conform to the following format (called Form 1):

Form 1: "http://www.upnp.org/schemas/av/" ***schema-root-name*** "-v" ***ver*** "-" ***yyyymmdd***

where

- ***schema-root-name*** is the name of the root element of the namespace that this schema represents.
- ***ver*** corresponds to the version number of the namespace that is represented by the schema.
- ***yyyymmdd*** is the year, month and day (in the Gregorian calendar) that this schema was released.

Table 1-4, “Schema-related Information” identifies the URI formats for each of the namespaces that are currently defined by the UPnP AV Working Committee.

As an example, the original schema URI for the “rcs-event” namespace (that was released with the original publication of the UPnP AV service specifications in the year 2002) was

"http://www.upnp.org/schemas/av/rcs-event-v1-20020625.xsd". When the UPnP AV service specifications were subsequently updated in the year 2006, the URI for the updated version of the "rcs-event" namespace was "http://www.upnp.org/schemas/av/rcs-event-v2-20060531.xsd". However, in 2006, the schema URI for the newly created "srs-event" namespace was "http://www.upnp.org/schemas/av/srs-event-v1-20060531.xsd". Note the version field for the "srs-event" schema is "v1" since it was first version of that namespace whereas the version field for the "rcs-event" schema is "v2" since it was the second version of that namespace.

In addition to the dated schema URIs that are associated with each namespace, each namepace also has a set of undated schema URIs. These undated schema URIs have two distinct formats with slightly different meanings:

Form 2: "http://www.upnp.org/schemas/av/" *schema-root-name* "-v" ***ver***

where ***ver*** is described above.

Form 3: "http://www.upnp.org/schemas/av/" *schema-root-name*

Form 2 of the undated schema URI is always linked to the most recent release of the schema that represents the version of the namespace indicated by ***ver***. For example, the undated URI ".../av/rcs-event-v2.xsd" is linked to the most recent schema release of version 2 of the "rcs-event" namespace. Therefore, on May 31, 2006 (20060531), the undated schema URI was linked to the schema that is otherwise known as ".../av/rcs-event-v2-20060531.xsd". Furthermore, if the schema for version 2 of the "rcs-event" namespace was ever re-released, for example to fix a typo in the 20060531 schema, then the same undated schema URI (".../av/rcs-event-v2.xsd") would automatically be updated to link to the updated version 2 schema for the "rcs-event" namespace.

Form 3 of the undated schema URI is always linked to the most recent release of the schema that represents the highest version of the namespace that has been published. For example, on June 25, 2002 (20020625), the undated schema URI ".../av/rcs-event.xsd" was linked to the schema that is otherwise known as ".../av/rcs-event-v1-20020625.xsd". However, on May 31, 2006 (20060531), that same undated schema URI was linked to the schema that is otherwise known as ".../av/rcs-event-v2-20060531.xsd".

When referencing a schema URI within an XML instance document or a referencing XML schema document, the following usage rules apply:

- All instance documents, whether generated by a service or a control point, MUST use Form 3.
- All UPnP AV published schemas that reference other UPnP AV schemas MUST also use Form 3.

Within an XML instance document, the definition for the `schemaLocation` attribute comes from the XML Schema namespace "http://www.w3.org/2002/XMLSchema-instance". A single occurrence of the attribute can declare the location of one or more schemas. The `schemaLocation` attribute value consists of a whitespace separated list of values that is interpreted as a namespace name followed by its schema location URL. This pair-sequence is repeated as necessary for the schemas that need to be located for this instance document.

In addition to the schema URI naming and usage rules described above, each released schema MUST contain a `version` attribute in the `<schema>` root element. Its value MUST correspond to the format:

ver "-" ***yyyymmdd*** where ***ver*** and ***yyyymmdd*** are described above.

The `version` attribute provides self-identification of the namespace version and release date of the schema itself. For example, within the original schema released for the "rcs-event" namespace (.../rcs-event-v2-20020625.xsd), the `<schema>` root element contains the following attribute: `version="2-20020625"`.

1.4.3 Namespace Usage Examples

The `schemaLocation` attribute for XML instance documents comes from the XML Schema instance namespace “http::://www.w3.org/2002/XMLSchema-instance”. A single occurrence of the attribute can declare the location of one or more schemas. The `schemaLocation` attribute value consists of a whitespace separated list of values: namespace name followed by its schema location URL. This pair-sequence is repeated as necessary for the schemas that need to be located for this instance document.

Example 1:

Sample *DIDL-Lite XML Instance Document*. Note that the references to the UPnP AV schemas do not contain any version or release date information. In other words, the references follow Form 3 from above. Consequently, this example is valid for all releases of the UPnP AV service specifications.

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="18" parentID="13" restricted="0">
      ...
   </item>
</DIDL-Lite>
```

1.5 Vendor-defined Extensions

Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation MUST follow the naming conventions and XML rules as specified below.

1.5.1 Vendor-defined Action Names

Vendor-defined action names MUST begin with “X_”. Additionally, it SHOULD be followed by an ICANN assigned domain name owned by the vendor followed by the underscore character (“_”). It MUST then be followed by the vendor-assigned action name. The vendor-assigned action name MUST NOT contain a hyphen character (“-”, 2D Hex in UTF-8) nor a hash character (“#”, 23 Hex in UTF-8). Vendor-assigned action names are case sensitive. The first character of the name MUST be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), or a non-experimental Unicode letter or digit greater than U+007F. Succeeding characters MUST be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), a period (“.”), a Unicode combiningchar, an extender, or a non-experimental Unicode letter or digit greater than U+007F. The first three letters MUST NOT be “XML” in any combination of case.

1.5.2 Vendor-defined State Variable Names

Vendor-defined state variable names MUST begin with “X_”. Additionally, it SHOULD be followed by an ICANN assigned domain name owned by the vendor, followed by the underscore character (“_”). It MUST then be followed by the vendor-assigned state variable name. The vendor-assigned state variable name MUST NOT contain a hyphen character (“-”, 2D Hex in UTF-8). Vendor-assigned action names are case sensitive. The first character of the name MUST be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), or a non-experimental Unicode letter or digit greater than U+007F. Succeeding characters MUST be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an

underscore (“_”), a period (“.”), a Unicode combiningchar, an extender, or a non-experimental Unicode letter or digit greater than U+007F. The first three letters MUST NOT be “XML” in any combination of case.

1.5.3 Vendor-defined XML Elements and attributes

UPnP vendors MAY add non-standard elements and attributes to a UPnP standard XML document, such as a device or service description. Each addition MUST be scoped by a vendor-owned XML namespace. Arbitrary XML MUST be enclosed in an element that begins with “**X_**,” and this element MUST be a sub element of a standard complex type. Non-standard attributes MAY be added to standard elements provided these attributes are scoped by a vendor-owned XML namespace and begin with “**X_**”.

1.5.4 Vendor-defined Property Names

UPnP vendors MAY add non-standard properties to the ContentDirectory service. Each property addition MUST be scoped by a vendor-owned namespace. The vendor-assigned property name MUST NOT contain a hyphen character (“-”, 2D Hex in UTF-8). Vendor-assigned property names are case sensitive. The first character of the name MUST be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), or a non-experimental Unicode letter or digit greater than U+007F. Succeeding characters MUST be a US-ASCII letter (“A”-“Z”, “a”-“z”), US-ASCII digit (“0”-“9”), an underscore (“_”), a period (“.”), a Unicode combiningchar, an extender, or a non-experimental Unicode letter or digit greater than U+007F. The first three letters MUST NOT be “XML” in any combination of case.

1.6 References

This subclause lists the normative references used in the UPnP AV specifications and includes the tag inside square brackets that is used for each such reference:

[AVARCH] – *AVArchitecture:1*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v1-20080930.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v1.pdf.

[AVDT] – *AV DataStructure Template:1*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/specs/av/UPnP-av-AVDataStructure-v1-20080930.pdf. Latest version available at: http://www.upnp.org/specs/av/UPnP-av-AVDataStructure-v1.pdf.

[AVDT-XSD] – *XML Schema for UPnP AV Datastructure Template:1*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/schemas/av/avdt-v1-20080930.xsd. Latest version available at: http://www.upnp.org/schemas/av/avdt-v1.xsd.

[AV-XSD] – *XML Schema for UPnP AV Common XML Data Types*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/schemas/av/av-v2-20080930.xsd. Latest version available at: http://www.upnp.org/schemas/av/av-v2.xsd.

[AVS-XSD] – *XML Schema for UPnP AV Common XML Structures*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/schemas/av/avs-v2-20071231.xsd. Latest version available at: http://www.upnp.org/schemas/av/avs-v2.xsd.

[AVT] – *AVTransport:2*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/specs/av/UPnP-av-AVTransport-v2-Service-20080930.pdf. Latest version available at: http://www.upnp.org/specs/av/UPnP-av-AVTransport-v2-Service.pdf.

[AVT-EVENT-XSD] – *XML Schema for AVTransport:2 LastChange Eventing*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/schemas/av/avt-event-v2-20080930.xsd. Latest version available at: http://www.upnp.org/schemas/av/avt-event-v2.xsd.

[CDS] – *ContentDirectory:3*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v3-Service-20080930.pdf. Latest version available at: http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v3-Service.pdf.

[CDS-EVENT-XSD] – *XML Schema for ContentDirectory:3 LastChange Eventing*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/schemas/av/cds-event-v1-20080930.xsd. Latest version available at: http://www.upnp.org/schemas/av/cds-event-v1.xsd.

[CM] – *ConnectionManager:2*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v2-Service-20080930.pdf. Latest version available at: http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v2-Service.pdf.

[DC-XSD] – *XML Schema for UPnP AV Dublin Core*. Available at: http://www.dublincore.org/schemas/xmls/simpledc20020312.xsd.

[DC-TERMS] – *DCMI term declarations represented in XML schema language*. Available at: http://www.dublincore.org/schemas/xmls.

[DEVICE] – *UPnP Device Architecture, version 1.0*, UPnP Forum, July 20, 2006. Available at: http://www.upnp.org/specs/architecture/UPnP-DeviceArchitecture-v1.0-20060720.htm. Latest version available at: http://www.upnp.org/specs/architecture/UPnP-DeviceArchitecture-v1.0.htm.

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

[DIDL-LITE-XSD] – *XML Schema for ContentDirectory:3 Structure and Metadata (DIDL-Lite)*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/schemas/av/didl-lite-v2-20080930.xsd. Latest version available at: http://www.upnp.org/schemas/av/didl-lite-v2.xsd.

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

[HTTP/1.1] – *HyperText Transport Protocol – HTTP/1.1*, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, June 1999. Available at: http://www.ietf.org/rfc/rfc2616.txt.

IEC 61883] – *IEC 61883 Consumer Audio/Video Equipment – Digital Interface - Part 1 to 5*. Available at: http://www.iec.ch.

[IEC-PAS 61883] – *IEC-PAS 61883 Consumer Audio/Video Equipment – Digital Interface - Part 6*. Available at: http://www.iec.ch.

[ISO 8601] – *Data elements and interchange formats – Information interchange -- Representation of dates and times*, International Standards Organization, December 21, 2000. Available at: ISO 8601:2000.

[MIME] – *IETF RFC 1341, MIME (Multipurpose Internet Mail Extensions)*, N. Borenstein, N. Freed, June 1992. Available at: http://www.ietf.org/rfc/rfc1341.txt.

[MR] – *MediaRenderer:2*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/specs/av/UPnP-av-MediaRenderer-v2-Device-20080930.pdf. Latest version available at: http://www.upnp.org/specs/av/UPnP-AV-MediaRenderer-v2-Device.pdf.

[MS] – *MediaServer:3*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/specs/av/UPnP-av-MediaServer-v3-Device-20080930.pdf. Latest version available at: http://www.upnp.org/specs/av/UPnP-AV-MediaServer-v3-Device.pdf.

[RCS] – *RenderingControl:2*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v2-Service-20080930.pdf. Latest version available at: http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v2-Service.pdf.

[RCS-EVENT-XSD] –*XML Schema for RenderingControl:2 LastChange Eventing*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/schemas/av/rcs-event-v1-20080930.xsd. Latest version available at: http://www.upnp.org/schemas/av/rcs-event.xsd.

[RFC 1738] – *IETF RFC 1738*, *Uniform Resource Locators (URL)*, Tim Berners-Lee, et. Al., December 1994. Available at: http://www.ietf.org/rfc/rfc1738.txt.

[RFC 2045] – *IETF RFC 2045*, *Multipurpose Internet Mail Extensions (MIME) Part 1:Format of Internet Message Bodies*, N. Freed, N. Borenstein, November 1996. Available at: http://www.ietf.org/rfc/rfc2045.txt.

[RFC 2119] – *IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels*, S. Bradner, 1997. Available at: http://www.faqs.org/rfcs/rfc2119.html.

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

[RFC 3339] – *IETF RFC 3339, Date and Time on the Internet: Timestamps*, G. Klyne, Clearswift Corporation, C. Newman, Sun Microsystems, July 2002. Available at: http://www.ietf.org/rfc/rfc3339.txt.

[RTP] – *IETF RFC 1889, Realtime Transport Protocol (RTP)*, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, January 1996. Available at: http://www.ietf.org/rfc/rfc1889.txt.

[RTSP] – *IETF RFC 2326, Real Time Streaming Protocol (RTSP)*, H. Schulzrinne, A. Rao, R. Lanphier, April 1998. Available at: http://www.ietf.org/rfc/rfc2326.txt.

[SRS] – *ScheduledRecording:2*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v2-Service-20080930.pdf. Latest version available at: http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v2-Service.pdf.

[SRS-XSD] – *XML Schema for ScheduledRecording:2 Metadata and Structure*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/schemas/av/srs-v2-20080930.xsd. Latest version available at: http://www.upnp.org/schemas/av/srs-v2.xsd.

[SRS-EVENT-XSD] – *XML Schema for ScheduledRecording:2 LastChange Eventing*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/schemas/av/srs-event-v1-20080930.xsd. Latest version available at: http://www.upnp.org/schemas/av/srs-event-v1.xsd.

[UAX 15] – *Unicode Standard Annex #15, Unicode Normalization Forms, version 4.1.0, revision 25*, M. Davis, M. Dürst, March 25, 2005. Available at: http://www.unicode.org/reports/tr15/tr15-25.html.

[UNICODE COLLATION] – *Unicode Technical Standard #10, Unicode Collation Algorithm version 4.1.0*, M. Davis, K. Whistler, May 5, 2005. Available at: http://www.unicode.org/reports/tr10/tr10-14.html.

[UPNP-XSD] – *XML Schema for ContentDirectory:3 Metadata*, UPnP Forum, September 30, 2008. Available at: http://www.upnp.org/schemas/av/upnp-v3-20080930.xsd. Latest version available at: http://www.upnp.org/schemas/av/upnp-v3.xsd.

[UTS 10] – *Unicode Technical Standard #10, Unicode Collation Algorithm, version 4.1.0, revision 14*, M. Davis, K. Whistler, May 5, 2005. Available at: http://www.unicode.org/reports/tr10/tr10-14.html.

[UTS 35] – *Unicode Technical Standard #35, Locale Data Markup Language, version 1.3R1, revision 5*,.M. Davis, June 2, 2005. Available at: http://www.unicode.org/reports/tr35/tr35-5.html.

[XML] – *Extensible Markup Language (XML) 1.0 (Third Edition)*, François Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, eds., W3C Recommendation, February 4, 2004. Available at: http://www.w3.org/TR/2004/REC-xml-20040204.

[XML-NS] – *The "xml:" Namespace*, November 3, 2004. Available at: http://www.w3.org/XML/1998/namespace.

[XML-XSD] – *XML Schema for the "xml:" Namespace*. Available at: http://www.w3.org/2001/xml.xsd.

[XML-NMSP] – *Namespaces in XML*, Tim Bray, Dave Hollander, Andrew Layman, eds., W3C Recommendation, January 14, 1999. Available at: http://www.w3.org/TR/1999/REC-xml-names-19990114.

[XML SCHEMA-1] – *XML Schema Part 1: Structures, Second Edition*, Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, W3C Recommendation, 28 October 2004. Available at: http://www.w3.org/TR/2004/REC-xmlschema-1-20041028.

[XML SCHEMA-2] – *XML Schema Part 2: Data Types, Second Edition*, Paul V. Biron, Ashok Malhotra, W3C Recommendation, 28 October 2004. Available at: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028.

[XMLSCHEMA-XSD] – *XML Schema for XML Schema*. Available at: http://www.w3.org/2001/XMLSchema.xsd.

[XPATH20] – *XML Path Language (XPath) 2.0*. Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernandez, Michael Kay, Jonathan Robie, Jerome Simeon. W3C Recommendation, 21 November 2006. Available at: http://www.w3.org/TR/xpath20.

[XQUERY10] – *XQuery 1.0 An XML Query Language*. W3C Recommendation, 23 January 2007. Available at: http://www.w3.org/TR/2007/REC-xquery-20070123.

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:3*

ContentDirectory service is used herein to refer to this service type.

2.2 Key Concepts

2.2.1 *On-line* and *Off-line* Network States

In the context of the ContentDirectory service, a device is considered *attached* to the network (a.k.a. *on-line* or *connected*, or *re-connected*) when the device is physically attached to the network and has sent a UPnP **ssdp:alive** message that has not yet expired as defined in subclause 1.2.2, "Discovery:Advertisement:Device available - NOTIFY with ssdp:alive" of the UPnP Device Architecture specification [DEVICE] for details. A device is considered to be *unattached* to the network (a.k.a. *off-line* or *disconnected*) when it sends a UPnP **ssdp:byebye** message or when all of the device's **ssdp:alive** messages have expired. See subclause 1.2.2, "Discovery:Advertisement:Device unavailable - NOTIFY with ssdp:byebye" of the UPnP Device Architecture specification [DEVICE] for details.

A UPnP control point is considered *connected* to a network when it is physically attached to that network, is actively monitoring the UPnP multicast discovery address, and is capable of receiving **ssdp:alive** messages from connected devices as described in subclause 1.2.2, "Discovery:Advertisement:Device available - NOTIFY with ssdp:alive" of the UPnP Device Architecture specification [DEVICE].

2.2.2 object

A ContentDirectory object is a structured set of metadata properties representing entertainment content that may be playable on a device connected to a network. For example, an object can represent:

- Static content such as a stored song, photo, video, etc.
- Transient content (such as broadcast program) that will be temporarily accessible in real-time (for example, a live broadcast).
- A long-lived access portal for dynamic content such as a "Most recently played" placeholder.
- A collection of other objects (called a "container").

As illustrated above, an object can represent individual content (for example, a song or photo) or a collection of content (for example, a photo album or the contents of an audio CD). Objects are typically obtained from the ContentDirectory service via a DIDL-Lite compliant XML document usually by way of an action parameter whose data type is *A_ARG_TYPE_Result*. See subclause 2.5.6, "*Browse()*" and subclause 2.5.8 "*Search()*" for examples. See subclause 2.3.12, "*A_ARG_TYPE_Result*" for details.

Each ContentDirectory object includes a set of metadata properties that provide various information about the object and the content that the object represents. Examples include a unique ID, a class, a title, one or more artists, the time created, the access method for the content, etc. See subclause 2.2.20 "property" for details.

For identification purposes, the ContentDirectory service MUST assign a unique ID (called an *object ID*) to each object. The object ID is the one and only reliable method for identifying a specific object. Although multiple objects can represent the same piece of content (for example, the same song exposed by both a genre container and an artist container), each object has its own unique object ID. See subclause 2.2.3, "Object Identity" for details.

The ContentDirectory service also defines an object class hierarchy that corresponds to the different types of objects that are managed by the ContentDirectory service. The root (base) class of the class hierarchy, from which all other classes are derived, is named *object*. Although the *object* class itself cannot be instantiated, all classes derived from the *object* class can be instantiated. See subclause 2.2.6, "class" for details.

2.2.3 Object Identity

Each object that is managed by a ContentDirectory service is assigned a unique ID by the implementation. This object ID, which is exposed via the object's *@id* property, provides a unique identity for the object. The *@id* property is essential for reliably identifying a specific object among those hosted by a particular ContentDirectory service because no other object metadata (or combinations of metadata) provides a distinct identity for the object. For example, two distinct objects, each with a unique *@id* property value, can have identical metadata (except for the *@id* property) such as two objects representing the same content. Without the *@id* property, there is no way to definitively distinguish between the two objects. Even when there are differences in the metadata of two objects, those differences may be temporary and do not provide a reliable way to distinguish the two objects.

Since the ContentDirectory implementation assigns each object's *@id* property value, the implementation fully controls the lifetime of each object; that is: it controls whether or not an object retains its identity. Specifically, when an implementation returns an object with an *@id* property value that has been returned in the past, the implementation is indicating that this object is the same object that was returned previously. Even when the object's metadata has changed, an identical *@id* property value indicates that this is the same object as before. Similarly, when an implementation assigns a new value to an object's *@id* property, the implementation is creating a new object and declaring that this object is different from any other previously known object even if the object's metadata is identical to a previously known object.

If the *@id* property values are preserved, control points are able to correlate objects across time, even across periods when the control point or the ContentDirectory service is *off-line*. See subclause 2.2.1, "*On-line* and *Off-line* Network States". For example, a control point can maintain a *Favorites* or a *Most Recently Used* list of objects so that an end-user can quickly locate specific content. Without persistent *@id* property values (that is: with objects that have only a temporary identity), a control point would not be able to deterministically locate the same object that it used earlier.

Additionally, if a ContentDirectory service implementation uses an *@id* property value for one object and when that object is deleted reuses that same *@id* property value for a new and different object, a control point can mistake the new object for the original, deleted object. This is known as object *@id* reuse. Note that if the ContentDirectory service implementation does reuse the *@id* property value of a deleted object, it is making the statement that the new object is exactly the same object as was deleted previously (although its metadata may have changed). A ContentDirectory service implementation MUST enforce the above rule for object *@id* reuse during periods when the *ServiceResetToken* state variable is constant. See subclause 2.3.7, "*ServiceResetToken*" for details.

Preserving the *@id* property value across periods when the server is *off-line* is RECOMMENDED for all objects. For those objects that support tracking changes, through the exposure of the *upnp:objectUpdateID* and *upnp:containerUpdateID* properties, preservation of the *@id* property value across periods when the ContentDirectory service implementation is *off-line* is REQUIRED. See subclause 2.6.2, "Generating Object ID Values" for examples of mechanisms to generate persistent and unique *@id* property values.

2.2.4 Object Lifetime

The term *object lifetime* refers to the period of time that an object exists. By definition, a ContentDirectory service object exists as long as it is accessible by a control point via the ContentDirectory service *Browse()* action. Since an object's identity is defined by the value of the object's *@id* property, an object's lifetime is directly related to the period of time that the object's *@id* property value can be used to locate the object.

Although the duration of an object's lifetime (short vs. long) can be influenced by many factors, the two most common situations that truncate an object's lifetime (that is: cause the object to cease to exist) are:

- Deleting the object from the ContentDirectory service.
- Changing the value of the object's *@id* property thereby creating a brand new object identity.

Some objects tend to be inherently short-lived because they are deleted shortly (for example, within a few hours) after they are created because the object's usefulness fades. Such objects might include EPG objects which represent broadcast programs or objects that are stored on removable media.

Other objects are inherently long-lived, for example, objects that represent files that reside in storage controlled by the ContentDirectory service. As discussed in the previous subclause (2.2.3, "Object Identity"), preserving the longevity of inherently long-lived objects provide control points with certain advantages. ContentDirectory service implementations are RECOMMENDED to maintain the lifetime of inherently long-lived objects, for example, by routinely preserving their identity. See subclause 2.6.2, "Generating Object ID Values" for details.

2.2.5 Object Modification

Except as noted below, an *Object Modification* occurs when the value of one or more of an object's properties is modified, added, or deleted. This includes any vendor-defined properties. Adding or deleting a child object of a container object constitutes an *Object Modification* on the container only when one or more exposed properties of the container (with the exceptions noted below) change as a result of that add or delete (such as the *@childCount* or *upnp:totalDeletedChildCount* properties). However, a change to any property belonging to any of the container's child objects MUST NOT be treated as an *Object Modification* of the container. See subclause 2.2.9, "Container " for details.

Exceptions:

The following properties are excluded from the definition of an *Object Modification*:

- *upnp:objectUpdateID*
- *upnp:containerUpdateID*

Consequently, a modification to any of the above properties MUST NOT be treated as an *Object Modification*. For example, the *SystemUpdateID* state variable and all *upnp:containerUpdateID* properties (defined below) MUST NOT be incremented when either of these properties are modified.

2.2.6 class

A class is used to assign a type to an object. It also identifies the minimum REQUIRED set of properties that MUST be included in the object's metadata and the OPTIONAL properties that MAY be included. 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 C.1.1, "Class name syntax" for a definition of the format of the class specification for an object.

2.2.7 *item*

An *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 is derived from the *item* class is represented in XML using the DIDL-Lite element `<item>…</item>`.

Note: The term item is used in this specification to indicate an object whose class is either *item* or any of the defined *item*-derived classes.

2.2.8 *container*

A *container* is a first-level class derived directly from *object*. The term container is used in this specification to indicate an object whose class is either *container* or any of the defined *container*-derived classes. A container instance 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 represented in XML using the DIDL-Lite element `<container>…</container>`.

Note: A ContentDirectory service implementation is REQUIRED to maintain a *ContainerUpdateIDValue* indicator for each of its containers. See subclause 2.2.11, "*ContainerUpdateIDValue* Indicator" for details.

2.2.9 Container Modification

Since a container (that is: any object whose class is derived from the *container* class) is derived from the *object* class, the semantics of an *Object Modification* also apply to all container objects. However, since a container contains other objects (see subclause 2.2.8, "*container*") the concept of a *Container Modification* is introduced. It is used to indicate that some change has occurred within a container (for example, one or more of the container's properties has changed, this is also defined as an *Object Modification* of that container) or within any of its child item(s) (that is: a child object whose class IS NOT derived from the *container* class). This includes any vendor-defined properties within the container or any child item. Child container objects (that is: children derived from the *container* class) do not generate a *Container Modification* for their parent container because they have their own notion of a *Container Modification*. Each change in the ContentDirectory service's metadata results in one and only one *Container Modification* which is associated with one and only one container.

In specific terms, except as noted below, a container experiences a *Container Modification* when any of the following conditions occur:

- The container experiences an *Object Modification*; that is: one or more properties of the container (including any vendor-defined property) are added, removed or changed. See subclause 2.2.5, "Object Modification" for details.
- A child object (either a child item or child container - including vendor-defined object classes) is added to or removed from the container.
- A child item (i.e. an object whose class IS NOT derived from the *container* class) has any of its properties added, removed or changed, except those explicitly listed as exceptions in the *Object Modification* definition in subclause 2.2.3 (Object Modification).

Note: The exceptions listed under the definition of object modification also apply to a container modification. Refer to the exceptions listed in subclause 2.2.5, "Object Modification"

2.2.10 ContentDirectory Tracking Changes Option

A ContentDirectory service implementation MAY choose to implement the *Tracking Changes Option*. This means that the ContentDirectory service implementation supports all necessary state variables, properties, and eventing mechanisms to expose to control points the changes to individual objects within the ContentDirectory hierarchy. If the ContentDirectory service implementation supports the *Tracking Changes Option*, there can be objects within the ContentDirectory hierarchy for which the service does not support tracking changes; for example, objects that frequently change, such as EPG data. At any point in time, a

ContentDirectory service implementation can support the *Tracking Changes Option* but have no objects for which it is currently tracking changes.

If the ContentDirectory service implementation supports the *Tracking Changes Option*, then it MUST:

- Support the *LastChange* state variable as defined in subclause 2.3.8, "*LastChange*".
- Implement the *Search()* action.
- Include the *upnp:objectUpdateID* and *upnp:containerUpdateID* properties in the *SearchCapabilities* state variable.
- Support the following operators for the *upnp:objectUpdateID* and *upnp:containerUpdateID* properties: <, <=, >=, >, = , !=, exists.
- Support the = operator for the *@id* and *@parentID* properties.
- Support the = and derivedFrom operators for the *upnp:class* property.

The following metadata properties MUST be implemented for all items for which the ContentDirectory service implementation is tracking changes:

- *upnp:objectUpdateID*
- *res@updateCount*

The following metadata properties MUST be implemented for all containers for which the ContentDirectory service implementation is tracking changes:

- *upnp:containerUpdateID*
- *upnp:objectUpdateID*
- *@childCount*
- *upnp:totalDeletedChildCount*

If the ContentDirectory service implementation does not implement the *Tracking Changes Option*, then the ContentDirectory service implementation MUST NOT support the *LastChange* state variable and the following metadata properties MUST NOT be used on any object within the ContentDirectory hierarchy:

- *upnp:objectUpdateID*
- *res@updateCount*
- *upnp:containerUpdateID*
- *upnp:totalDeletedChildCount*

A control point can determine if the ContentDirectory service implementation supports the *Tracking Changes Option* by checking for the existence of the *LastChange* state variable in the SCPD.

2.2.11 *ContainerUpdateIDValue* Indicator

The *ContainerUpdateIDValue* indicator is an internal, unsigned integer that MUST be maintained for each instance of class *container* and any of its derived classes. In previous versions of the specification, this was simply known as the *ContainerUpdateID*. However with this version of the specification it is known as the *ContainerUpdateIDValue* indicator in order to differentiate it from the exposed optional *upnp:containerUpdateID* property, which, if present, carries this same value.

If the ContentDirectory service implementation supports the *Tracking Changes Option*, then the *ContainerUpdateIDValue* indicator (whether or not it is exposed in a corresponding *upnp:containerUpdateID* property of the container) MUST be the same as the property value defined in B.15.1, "*upnp:containerUpdateID*".

If the ContentDirectory service implementation does not support the *Tracking Changes Option*, then the *ContainerUpdateIDValue* indicator is incremented each time the container is modified (see subclause 2.2.9, "Container Modification" for the precise definition of *Container Modification*. Upon reaching the value of 2^{32}-1, the next update rolls the value back to 0, and the implementation MUST invoke the *Service Reset Procedure* as defined in 2.3.7.1, "*Service Reset Procedure*". The initial *ContainerUpdateIDValue* indicator value for any newly created container is unspecified, but RECOMMENDED to be 0. Implementers SHOULD maintain the same value for each container's *ContainerUpdateIDValue* indicator through power cycles and any other disappearance/appearance on the network. The *ContainerUpdateIDValue* indicator is not a formal property of a container object, so a modification to a child container that affects that child's *ContainerUpdateIDValue* indicator does not propagate upward to the parent container.

2.2.12 ContentDirectory Service Object Organization

From a logical viewpoint, objects are organized in a ContentDirectory service according to a tree hierarchy. This tree hierarchy is called the ContentDirectory service content hierarchy. At the origin (top) of the ContentDirectory service content hierarchy, there is the single root container. This root container contains all other objects—items and containers, in a hierarchical tree fashion—that make up the entire ContentDirectory service content. Containers can contain both sub-containers and items. Items cannot contain other objects. Items are therefore always leaf nodes on the tree. The following figure illustrates the concepts. (The figure represents a hypothetical ContentDirectory service content structure.)

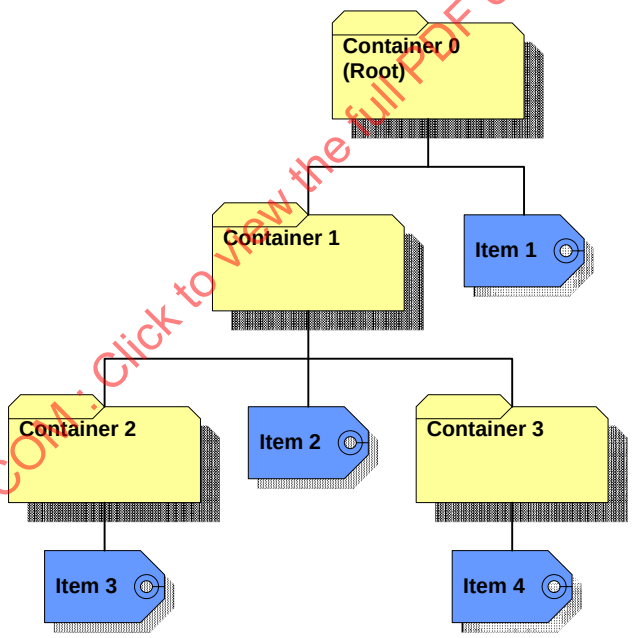


Figure 1 — ContentDirectory Service Object Organization.

Container 0 is the root container of the ContentDirectory service tree hierarchy.

2.2.13 Hierarchical location

Within the ContentDirectory service tree hierarchy, an object resides directly below a container if that object's *@parentID* property value equals the *@id* property value of that container. That container is called the *parent (container)* of the object (one level up) and the container is said to have a parent relationship with the object. The object is called a *child (object)* of the container (one level down) and the object is said to have a child relationship with that container.

A child object can be either an item object or a container object. In Figure 1, Item 1 is a child of Container 0 (the Root container). Likewise, Item 3 is a child of Container 2. Any object can

only have one parent container, except for the root container, which has none (indicated by setting its *@parentID* property value to "-1"). A parent container can have multiple child objects. In Figure 1, Container 1 is the parent of Container 2 , Item 2, and Container 3.

An object is called a *descendant object* of a container if it is a child of that container or if it is connected to that container through one or more intermediate child relationships (any level down). A descendant object can be either an item object or a container object. All objects (except the root container) are descendants of the root container. In Figure 1, Item 3 is a descendant of Container 1.

A container is called an *ancestor container* of an object if that container is a parent of that object or if it is connected to the object through one or more intermediate parent relationships (any level up). The root container is an ancestor of all objects within the ContentDirectory service. In Figure 1, Container 1 is an ancestor of Item 4.

See B.1.1, "*@id*" and B.1.2, "*@parentID*" for additional information.

2.2.14 Subtree

A subtree for a given container is defined as the container itself plus the collection of all objects that have a descendant relationship to that container. That container is called the root container of the subtree. Each container within the ContentDirectory service tree hierarchy is the root container of a single subtree. In Figure 1, Container 1 is the subtree root container of the subtree that consists of Container1, Container 2, Item 2, Container 3, Item 3, and Item 4.

2.2.15 Subtree Updates

In some situations, a ContentDirectory Service needs to manipulate (that is: add, modify, or delete) a relatively large number of objects within a single ContentDirectory subtree (that is: all of the objects to be updated are descendants of a single container). Typically, these situations involve an internal operation such as searching a newly found storage medium for new content or updating a large set of EPG objects with fresh data. While performing these types of operations, numerous objects may be created, modified, and/or deleted in a short period of time.

As described in subclause 2.3.8, "*LastChange*", each object update triggers an event that may need to be processed by a control point. However, the large number of rapidly occurring events can overwhelm the capabilities of some control points. Consequently, the events contained in the *LastChange* state variable that correspond to these large subtree updates, may be tagged with the `stUpdate` attribute so that they can be distinguished from other update events which occur far less rapidly and affect far fewer objects (for example, those events triggered by the *CreateObject()* or *UpdateObject()* actions). The `stUpdate` attribute allows a control point to apply special processing algorithms that are specifically designed to accommodate a large number of object updates.

In addition to the `stUpdate` attribute, the *LastChange* state variable also defines a special-purpose event, called `<stDone>`, which is used by the device to indicate that a sub-tree update operation has finished. The `<stDone>` event identifies the container object that represents the root of the updated subtree. This information can be used by a control point to process the subtree updates more efficiently for example, container-by-container rather than object-by-object.

Although the mechanisms above allow a control point to more efficiently handle subtree updates, there is no precise definition of what constitutes a subtree update. Therefore, each ContentDirectory service implementation can designate various update operations as a subtree update as it deems appropriate. A given update operation MAY be represented as one or more subtree updates. The following guidelines identify when the subtree update mechanism should be used.

When to use the subtree update mechanism:

- When an entire sub-tree is added to or deleted from the ContentDirectory service.
- When the set of updates underneath a container object has a high “update density”. In other words, when the percentage of descendant objects that are modified vs. the total number of descendant objects within the subtree is relatively high.

When NOT to use the subtree update mechanism:

- When the set of objects that need to be updated are scattered throughout the ContentDirectory service; that is: there are no high update density subtree roots. If the update density is low, marking every container update as a subtree update could cause the control point to do more work than just processing each object update individually.

When an implementation chooses to use the subtree update mechanism, the following criteria MUST be obeyed:

- Each subtree update MUST have one and only one container object designated as the root of the subtree. In some extreme cases, the root of the sub-tree update may be the ContentDirectory service root container (*@id*=“0”).
- All objects represented by an object modification event with the `stUpdate` attribute set to one (“1”) MUST be a descendant of one and only one designated subtree root. In other words, the root of an active subtree update operation MUST NOT be a descendant of the root of another active subtree update operation.

2.2.16 XML Document

An XML document is a string that represents a valid XML 1.0 document according to a specific schema. Every occurrence of the phrase “*XML Document*” is italicized and preceded by the document’s root element name (also italicized), as listed in column 3, “Valid Root Element(s)” of Table 1-4, “Schema-related Information”.

For example, the phrase *DIDL-Lite XML Document* refers to a valid XML 1.0 document according to the DIDL-Lite schema [DIDL-LITE-XSD]. Such a document comprises a single `<DIDL-Lite …>` root element, optionally preceded by the XML declaration `<?xml version="1.0" …?>`.

This string will therefore be of one of the following two forms:

“`<DIDL-Lite …>…</DIDL-Lite>`”

or

“`<?xml …?><DIDL-Lite …>…</DIDL-Lite>`”

2.2.17 XML Fragment

An XML fragment is a sequence of XML elements that are valid direct or indirect child elements of the root element according to a specific schema. Every occurrence of the phrase “*XML Fragment*” is italicized and preceded by the document’s root element name (also italicized), as listed in column 3, “Valid Root Element(s)” of Table 1-4, “Schema-related Information”.

The following are examples of *DIDL-Lite XML Fragments*:

“`<item id="…" …>…</item>`”

or

“<res protocolInfo="…" …>…</res>”

or

“<dc:title>Sunrise</dc:title>”

2.2.18 *DIDL-Lite XML Document*

Whenever there is a need for action arguments to contain a description of (part of) the ContentDirectory service tree hierarchy (for example, the result of a *Browse()* or *Search()* action), a valid *DIDL-Lite XML Document* is used. The phrase *DIDL-Lite XML Document* refers to a valid XML 1.0 document according to the DIDL-Lite schema as defined in [DIDL-LITE-XSD].

Such a document comprises a single <DIDL-Lite …> root element, optionally preceded by the XML declaration <?xml version="1.0" …?>.

This string will therefore be of one of the following two forms:

“<DIDL-Lite …>…</DIDL-Lite>”

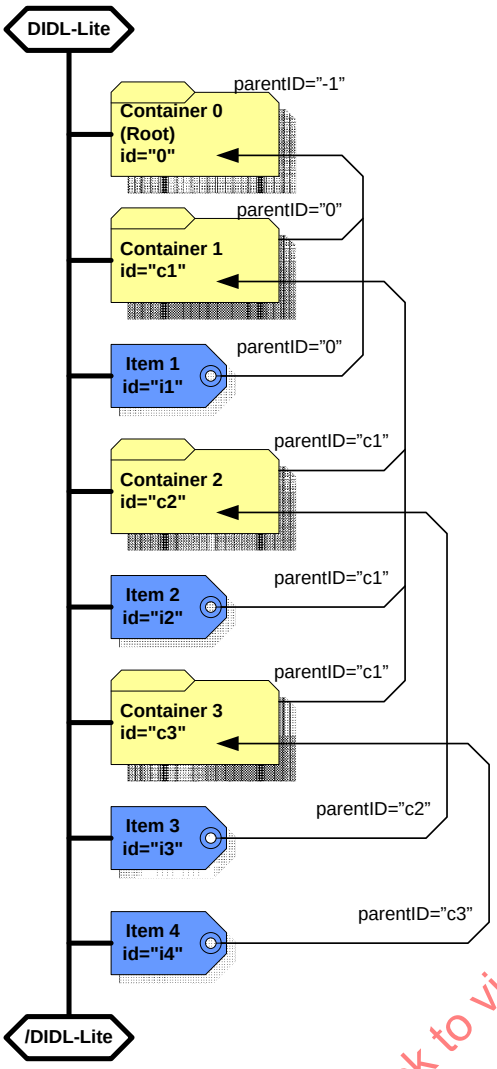
or

“<?xml …?><DIDL-Lite …>…</DIDL-Lite>”

The *DIDL-Lite XML Document* presents a flattened view of (part of) the ContentDirectory service tree hierarchy. It is important to make a clear distinction between the ContentDirectory tree hierarchy (a logical concept) and the *DIDL-Lite XML Document* (with its intrinsic document hierarchy), which is a syntax used to express (part of) the ContentDirectory tree hierarchy. Although it is perfectly possible to accurately express hierarchical structure in an XML Document (XML is intrinsicly hierarchical), this specification does not use XML hierarchy to express ContentDirectory service tree hierarchy. Instead, information about the hierarchical location of an object within the ContentDirectory service tree hierarchy is maintained by including the object ID of the parent container in which the object resides into the metadata of the object (*@parentID* property).

Within the context of the *DIDL-Lite XML Document*, *all* ContentDirectory service objects are represented as either <container> or <item> XML elements. They all reside at the same XML hierarchical level. All <container> or <item> XML elements are sub-elements of the XML root element <DIDL-Lite>. No <container> element can contain another <container> or <item> element. In other words, <container> and <item> elements MUST NOT be embedded in <container> elements.

The following figure illustrates this flattened view. To the right, the corresponding (incomplete, simplified) *DIDL-Lite XML Document* is also included.




```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite ...>
   <container id="0" parentID="-1" ...>
      ...
   </container>
   <container id="c1" parentID="0" ...>
      ...
   </container>
   <item id="i1" parentID="0" ...>
      ...
   </item>
   <container id="c2" parentID="c1" ...>
      ...
   </container>
   <item id="i2" parentID="c1" ...>
      ...
   </item>
   <container id="c3" parentID="c1" ...>
      ...
   </container>
   <item id="i3" parentID="c2" ...>
      ...
   </item>
   <item id="i4" parentID="c3" ...>
      ...
   </item>
</DIDL-Lite>
```

Figure 2 — Flattened DIDL-Lite hierarchical structure.

It must be noted that a *DIDL-Lite XML Document* does not necessarily contain enough information to determine the hierarchical location of an object, contained in the *DIDL-Lite XML Document*. Indeed, to determine the correct hierarchical location of an object in the ContentDirectory tree hierarchy, a control point needs to know all of that object's ancestor containers up to the root container. However, in many cases, based on the arguments to the *Browse()* or *Search()* action, is it possible that the resulting *DIDL-Lite XML Document* does not contain sufficient hierarchical information to reconstruct the exact location within the entire ContentDirectory service tree hierarchy. For example, a *Browse()* action on direct children, performed on Container 3 in the example from Figure 1 above will only return the following (incomplete, simplified) *DIDL-Lite XML Document*:

```
<?xml version="1.0" encoding="UTF-8">
<DIDL-Lite …>
   <item id="i4" parentID="c3" ...>
   </item>
</DIDL-Lite>
```

This *DIDL-Lite XML Document* only indicates that Item 4 resides below Container 3 but it does not convey that in turn, Container 3 resides below Container 1, which in turn resides below the root Container 0.

2.2.19 *CDS View*

CDS View is an XML representation of ContentDirectory objects that is used as input to an XQuery processor. A *CDS View* contains all descendant objects available underneath a given container in the ContentDirectory service at the time it is processed. The predominant *CDS View* that is used in the ContentDirectory service is the *DIDL-Lite XML Document* (a.k.a. *DIDL-Lite View*).

2.2.19.1 *DIDL-Lite View*

The *DIDL-Lite View* is represented by a flat-structured valid *DIDL-Lite XML Document* of an entire subtree of the ContentDirectory service. The container at which the subtree starts is called the subtree root container. The `<DIDL-Lite>` root element contains a single level sequence of `<container>` and `<item>` elements that represent the objects (containers and content items) that comprise the subtree. The *DIDL-Lite View* contains the subtree root container and all its descendant objects.

2.2.20 property

A property in the ContentDirectory service represents a characteristic of an object. Properties are distinguished by their names. The ContentDirectory service defines two kinds of properties – independent and dependent. Each independent property has zero or more dependent properties associated with it. Independent property names contain no "@" symbol; they may contain an XML namespace prefix (see below for an explanation of the relationship between properties and XML). Each dependent property is associated either with exactly one independent property or directly with a ContentDirectory service class such as an object.

The name of a dependent property that is associated with an independent property is the concatenation of three parts: its associated independent property name, the "@" symbol, and a name which conveys the relationship between the the dependent property and the associated independent property. For example, a dependent property named "rating@type" identifies the type of information that is stored within the associated independent property named "rating". The name of a dependent property that is associated directly with a ContentDirectory class is just the "@" symbol followed by a name which conveys its relationship with the class. For example, a dependent property named "@id" contains an identification value for a given instance of the class containing the "*@id*" property.

A small number of independent properties have independent child (nested) properties, which in turn, may have independent child properties of their own. To fully qualify a nested property, the fully qualified parent name is used, followed by "::", followed by the child property name, recursively. For example, xxx::yyy::zzz where zzz is the child of yyy and yyy is the child of xxx. The data types and meanings for all properties are defined in Annex B, "**(normative)**

AV Working Committee Properties".

Even though ContentDirectory service properties are not XML objects, XML is used to express them in all exchanges between a control point and a ContentDirectory service implementation. This creates an unavoidable relationship between XML syntax and property names and values. In XML, an independent property is represented as an element. The property name is used as the element name. The property value is the element content. A child property is represented as an element within the content of the element that represents the child's parent property. A dependent property is represented as an attribute in XML. The dependent property's relationship name (see above) is used as the attribute name. The dependent property's value is the attribute value. For dependent properties that are associated with an independent property, the attribute appears in the (opening tag of the) element that represents its associated independent property. For dependent properties that are associated directly with a class, the attribute appears in the (opening tag of the) top-level element that represent the object of that class. For some examples, see Table 2-1 "Properties in XML".

Table 2-1 — Properties in XML

Property Name	XML Representation (didl-lite declared as default namespace)
dc:title	`<dc:title>…</dc:title>`
res	`<res>…</res>`
res@size	`<res size="…">…</res>`
@id	`<item id="…">…</item>`

2.2.20.1 Multi-valued property

Some independent properties are multi-valued. This means that the property MAY occur more than once in an object.

2.2.20.2 Single-valued property

Most independent properties are single-valued. This means that the property MUST occur at most once in an object. Some single-valued properties can contain a CSV list of values. A dependent property is always considered single-valued, because it can occur at most once with each occurrence of its associated independent property, even though the independent property may be multi-valued.

2.2.21 *reference*, *reference item*, *referenced item*

A reference is a link from one ContentDirectory service item (that is: any object whose class is derived from the *item* class) to another item. It allows one item (the *reference item*) to expose the same metadata as the other item (the *referenced item*) without having to store duplicate copies of the metadata. In addition to eliminating duplicate physical copies of the *referenced item*'s metadata, a reference enables a *reference item* to automatically *track* metadata changes in the *referenced item*. For example, if there are three playlist containers that all contain child items representing the same song, the ContentDirectory service implementation may store one item that contains all of the song's metadata and store two (smaller) *reference items* that simply point to the one *referenced item*.

When a *reference item* is browsed (via *Browse()* or *Search()* actions), it MUST be returned as a valid DIDL-Lite object (for example, *@id*, *dc:title*, etc. properties are REQUIRED). The metadata of the returned object MUST be an exact copy of the metadata from the *referenced item* except for any of the following:

- The *reference item* MUST NOT inherit the following property values from the *referenced item*:
  - *@id*
  - *@parentID* (but may happen to have the same value as the *@parentID* property of the referenced item if they reside in the same container.)
  - *upnp:objectUpdateID* (The ContentDirectory service implementation determines whether it is tracking changes on the *reference item*, the *reference item* may or may not expose an *upnp:objectUpdateID* property independent of whether the *referenced item* exposes an *upnp:objectUpdateID* property.)
- The *reference item* MUST contain a *@refID* property, whose value MUST be equal to the value of the *@id* property of the *referenced item*. Note: Control points may use the existence of the *@refID* property to distinguish between a *referenced item* and all of the *reference items* that point to it.
- The *reference item* MAY (as described below) override (for example, change a property value or remove an existing property) any of the original *referenced item*'s properties except as listed above.

Additionally, a reference item MAY override any of the original referenced item's properties in one of the following ways:

- A *reference item* MAY be updated so that its metadata includes one or more additional properties not present in the *referenced item*.
- A *reference item* MAY be updated so that its metadata does not contain one or more of the existing properties of the *referenced item*.
- A *reference item* MAY be updated so that its metadata overrides the value of one or more existing properties of the *referenced item*.

All of the modifications listed above are bound only to the *reference item* and MUST NOT propagate back to the *referenced item*; that is: the original *referenced item* MUST NOT be affected by any modifications of the *reference item*. All resulting changes specified by the *reference item* MUST result in a valid DIDL-Lite object when subsequently browsed and/or searched.

Since a modification to a reference item (for example, via the *UpdateObject()* action), only affects that object (and not the underlying *referenced item*), each *reference item* modification results in a single event. However, when the underlying *referenced item* is modified, those property changes propagate to all of the *reference items* that refer to the modified *referenced item* but not to the *reference items* that override those modified properties. Such modifications constitue an *object modification* on each of the modified items. Additionally, the parent objects of each of the modified items also experience a *container modification*. For details, see subclause 2.2.5 “Object Modification” and subclause 2.2.9 “Container Modification”.

When the ContentDirectory service implementation contains multiple objects that refer to the same content, it is RECOMMENDED that the implementation use *reference items* for all but one of those objects. In other words, there should be a single *master* object for that content which is referenced by all of the other objects for that content. Additionally, it is RECOMMENDED that all *reference items* refer directly to the *master* object (that is: the object that has no *@refID* property) rather than referencing another reference item. In other words, a ContentDirectory service SHOULD NOT daisy-chain *reference items*.

2.2.22 CDS feature

The *CDS feature* exposes extended functionality of a ContentDirectory service implementation. Each *CDS feature* has a normative name, such as “*EPG*”, and a set of requirements to realize the feature. These requirements are defined in Annex E, “ **(normative)**

CDS features.”

2.2.23 Metadata vs. Foreign Metadata

For each ContentDirectory object there is a set of metadata that describes various characteristics of the object. This metadata is represented as a set of individual *properties* bundled together in an XML data structure that represents the object. (See subclause 2.2.20, “property” for more details.) The ContentDirectory service defines a standardized set of properties that a ContentDirectory service can use to expose the various characteristics of an object in a predictable way.

Some ContentDirectory service implementations may have access to additional metadata that can not be exposed via the standard set of metadata properties because there are no corresponding properties predefined within the ContentDirectory service. However, the ContentDirectory service does provide a mechanism for an implementation to expose this *foreign metadata* so that control points can extract and process it. This mechanism is called the *FOREIGN_METADATA Feature*. It is described in detail in subclause 2.6.14, “Foreign Metadata”, clause B.16, “Foreign Metadata-related Properties” and clause E.4, “Requirements for the *FOREIGN_METADATA feature*, Version 1”.

2.2.24 Embedded XML Documents

Object properties are represented as sub-elements of the `<container>` or `<item>` element. Some of these properties MAY contain entire XML documents or XML fragments. In that case, the *DIDL-Lite Document* will have XML Documents embedded in it. A complication with embedded XML Documents is that they possibly have a different XML version and/or encoding than the document in which they are embedded. However, to ensure that the *DIDL-Lite XML Document* used to describe (part of) the ContentDirectory service content is valid in its entirety, the embedded XML Documents MUST use the same encoding and XML version as the main *XML 1.0 DIDL-Lite XML Document* and any `<?xml …>` header that may be present in the original XML Document MUST be discarded before embedding.

2.3 State Variables

Unlike most other services, the ContentDirectory service is primarily action-based. The service state variables exist primarily to support argument passing in the service actions. Information is not exposed directly through explicit state variables. Rather, a client retrieves ContentDirectory service information via the return arguments 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 a first-time reader, it may be more helpful to read the action definitions before reading the state variable definitions.

2.3.1 State Variable Overview

Table 2-2 — State variables

Variable Name	R/O 1)	Data Type	Allowed Value	Default Value	Eng. Units
SearchCapabilities	*R*	string	CSV (string) See subclause 2.3.2		
SortCapabilities	*R*	string	CSV (string) See subclause 2.3.3		
SortExtensionCapabilities	*O* 2)	string	CSV (string) See subclause 2.3.4		
SystemUpdateID	*R*	ui4	See subclause 2.3.5		
ContainerUpdateIDs	*O*	string	CSV ({string,ui4}) See subclause 2.3.6		
ServiceResetToken	*R*	string	See subclause 2.3.7		
LastChange	*O*	string	*LastChange XML Document* See subclause 2.3.8		
TransferIDs	*O*	string	CSV (ui4) See subclause 2.3.9		
FeatureList	*R*	string	*Features XML Document* See subclause 2.3.10		
A_ARG_TYPE_ObjectID	*R*	string	See subclause 2.3.11		
A_ARG_TYPE_Result	*R*	string	See subclause 2.3.12		
A_ARG_TYPE_SearchCriteria	*O*	string	See subclause 2.3.13		
A_ARG_TYPE_BrowseFlag	*R*	string	BrowseMetadata, BrowseDirectChildren		
A_ARG_TYPE_Filter	*R*	string	CSV (string) See subclause 2.3.15		
A_ARG_TYPE_SortCriteria	*R*	string	CSV (string) See subclause 2.3.16		
A_ARG_TYPE_Index	*R*	ui4	See subclause 2.3.17		

Variable Name	R/O 1)	Data Type	Allowed Value	Default Value	Eng. Units
A_ARG_TYPE_Count	*R*	**ui4**	See subclause 2.3.18		
A_ARG_TYPE_UpdateID	*R*	**ui4**	See subclause 2.3.19		
A_ARG_Type_TransferID	*O*	**ui4**	See subclause 2.3.20		
A_ARG_Type_TransferStatus	*O*	**string**	See subclause 2.3.21		
A_ARG_Type_TransferLength	*O*	**string**	See subclause 2.3.22		
A_ARG_Type_TransferTotal	*O*	**string**	See subclause 2.3.23		
A_ARG_TYPE_TagValueList	*O*	**string**	CSV (**string**) See subclause 2.3.24		
A_ARG_TYPE_URI	*O*	**uri**	See subclause 2.3.25		
A_ARG_TYPE_CDSView	*O*	**ui4**	See subclause 2.3.26		
A_ARG_TYPE_QueryRequest	*O*	**string**	See subclause 2.3.27		
A_ARG_TYPE_QueryResult	*O*	**string**	See subclause 2.3.28		
A_ARG_TYPE_FFQCapabilities	*O*	**string**	See subclause 2.3.29		
Non-standard state variables implemented by a UPnP vendor go here	*X*	*TBD*	*TBD*	*TBD*	*TBD*

1) *R* = REQUIRED, *O* = OPTIONAL, *X* = Non-standard.

2) REQUIRED if implementation supports sort modifiers other than "+" and "-".

2.3.2 *SearchCapabilities*

This state variable contains a CSV list of property names that can be used in search queries. Each property name MUST include the standard namespace prefix for that property, except for the DIDL-Lite namespace. Properties in the DIDL-Lite namespace MUST always be returned without the prefix. See Table 1-3, "Namespace Definitions" in subclause 1.4, "Management of XML Namespaces in Standardized DCPs" and Annex B, "**(normative)**

AV Working Committee Properties" for details.

If a ContentDirectory service does not implement the *Search()* action, then the *SearchCapabilities* state variable MUST be the empty string (""). If a ContentDirectory service implements the *Tracking Changes Option* then the *Search()* action is required and Table 2-3 identifies the minimum set of properties and operators on those properties that MUST be supported for searching.

All property names MUST be fully qualified using the double colon ("::") syntax as defined in subclause 2.2.20, "property". For example, "`upnp:foreignMetadata::fmBody::fmURI`"

A wildcard ("*") indicates that the device supports search queries using any property name(s) supported by this ContentDirectory service implementation.

Note: It is RECOMMENDED that implementations explicitly enumerate all of the properties that are supported for the *Search()* action and not use the wildcard ("*") indicator.

When the *Tracking Changes Option* is supported, the ContentDirectory service is required to provide certain search capabilities. The following table identifies the required search capabilities values that MUST be supported when the *Track Changes Option* is supported.

Table 2-3 — *SearchCapabilities* requirements for supporting *Tracking Changes Option*

Value	R/O	Required Operators

"@id"	R	=,
"@parentID"	R	=,
"upnp:class"	R	=, derivedFrom
"upnp:objectUpdateID"	R	<, <=, >=, >, =, !=, exists
"upnp:containerUpdateID"	R	<, <=, >=, >, =, !=, exists

2.3.3 SortCapabilities

This state variable is a CSV list of property names that the ContentDirectory service can use to sort Search() or Browse() action 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 property names supported by the ContentDirectory service. The property names returned MUST include the appropriate namespace prefixes, except for the DIDL-Lite namespace. Properties in the DIDL-Lite namespace MUST always be returned without the prefix. All property names MUST be fully qualified using the double colon ("::") syntax as defined in subclause 2.2.20, "property". For example, "`upnp:foreignMetadata::fmBody::fmURI`"

2.3.4 SortExtensionCapabilities

This state variable is a CSV list of sort modifiers that the ContentDirectory service can use to sort Search() or Browse() results. Table 2-4, "Sort Modifiers" defines the standard sort modifiers. Other standard sort modifiers MAY be defined in future versions of this specification. Vendors MAY define vendor-specific sort modifiers.

Modifiers MUST be treated as case-sensitive.

Omitting this state variable is identical to listing only "+" and "-" modifiers. If modifiers other than "+" and "-" are supported, this state variable is REQUIRED.

Table 2-4 — Sort Modifiers

Sort Modifiers	Descriptions
+, -	The "+" and "-" modifiers indicate that the sort is in ascending or descending order, respectively, with regard to the value of its associated property. The modifiers "+" and "-"MUST be supported by any service that supports sorting. Sorting support is indicated by a non-empty value for the SortCapabilities state variable. When a ContentDirectory service implements the SortExtensionCapabilities state variable, the values "+" and "-" MUST be included.
TIME+, TIME-	The "TIME+" and "TIME-" modifiers indicate the sort is in ascending or descending order, respectively, with regard to only the time part value of the date format property. For example, sorting on "`TIME+dc:date`" results in the following response. Either both of these modifiers MUST be supported or neither of them. If a time zone offset is included in the property's value, it MUST be accounted for in the sort results. If no time zone offset is included, the time value is assumed to be local time. 1. Object D that has "2005-02-10" in dc:date. 2. Object A that has "2004-05-08T10:00:00" in dc:date. 3. Object C that has "2004-05-11T12:00:00" in dc:date. 4. Object B that has "2003-02-12T18:30:00" in dc:date. As shown above, some objects may not have a value for the time part in the specified property. In that case, such objects MUST appear before the other sorted results in ascending order or after in descending order. If no time value is present the implementation MUST assume that nothing is known about it. This is *not* equivalent to a time value of "00:00:00".
Vendor defined	Vendors MAY add sort modifiers.

2.3.5 SystemUpdateID

The SystemUpdateID state variable is REQUIRED and is modified whenever a change occurs within the ContentDirectory service hierarchy. A change could be an added or deleted object,

or a change in the metadata of an object. This does not include changes to state variables of the service. This variable is evented and the event is moderated at a maximum rate of 5 Hz (once every 0.2 seconds).

Changing the *SystemUpdateID* state variable MUST occur atomically with the action that triggered the object modification(s). In other words, all of the necessary *Object Modification*(s) and their corresponding change(s) of the *SystemUpdateID* state variable MUST be completed before the triggering action returns (for example, the *CreateObject*() or *UpdateObject*() actions).

2.3.5.1 *SystemUpdateID* when Supporting the Tracking Changes Option

If the ContentDirectory service implementation supports the *Tracking Changes Option*, (even if it does not currently have any objects which expose the *upnp:objectUpdateID* or *upnp:containerUpdateID* properties), the *SystemUpdateID* state variable contains a numeric value that is incremented whenever information within the ContentDirectory service hierarchy that is visible to a control point changes.

The *SystemUpdateID* state variable MUST be incremented by 1 whenever any of the following occurs:

- An object experiences an *Object Modification*. See subclause 2.2.5, “Object Modification” for details.
- A new object is created.
- An existing object is deleted.

The *SystemUpdateID* state variable MUST NOT be incremented for any reason other than those listed above.

Additionally, the *SystemUpdateID* state variable MUST be preserved and incremented according to the above rules during periods while the ContentDirectory service is *off-line*. See subclause 2.2.1, “*On-line* and *Off-line* Network States”. Although its value will continually increase (due to persistence), its maximum value can accommodate highly dynamic objects. For example, the *SystemUpdateID* state variable can accommodate 10 updates per second, 24 hours a day, 365 days a year for over 13 years or one million (1,000,000) updates every day for nearly 11 years. In the unlikely situation where the value of *SystemUpdateID* reaches its maximum (**ui4**) value of 4294967295 (2^{32}-1), the ContentDirectory service implementation MUST invoke the *Service Reset Procedure*. See subclause 2.3.7, “*ServiceResetToken*” and subclause 2.3.7.1, “*Service Reset Procedure*” for details.

If the ContentDirectory service cannot meet the above requirements for any reason (such as a corrupted internal state), then the service MUST invoke the *Service Reset Procedure*. See subclause 2.3.7.1, “*Service Reset Procedure*” for details.

In many cases, multiple properties of the same object can be modified by a single operation such as the *UpdateObject()* action. In these situations, an implementation SHOULD represent all of these property changes (within the same object) by a single increment (by 1) of the *SystemUpdateID* state variable. However, when multiple objects are modified, the *SystemUpdateID* state variable MUST be incremented at least once for each object that is modified.

Since part of the *LastChange* state variable is based on the *SystemUpdateID* state variable (that is: the `updateID` attribute of each event), each increment of the *SystemUpdateID* state variable while the ContentDirectory service is *on-line* will correspond to a specific *LastChange* event. See subclause 2.3.8, “*LastChange*”. Also, since the *upnp:objectUpdateID* property values are based on the *SystemUpdateID* state variable, each object that exposes the *upnp:objectUpdateID* property will have a unique *SystemUpdateID* value stored in its *upnp:objectUpdateID* property. See B.15.2, “*upnp:objectUpdateID*”. Additionally, each container object that exposes the *upnp:containerUpdateID* property will have a unique

SystemUpdateID value stored in its *upnp:containerUpdateID* property. However, within a container object, its *upnp:objectUpdateID* and *upnp:containerUpdateID* properties MAY have the same value.

Due to the relationship between the *SystemUpdateID* state variable and the *upnp:objectUpdateID* property, the initial value of the *SystemUpdateID* state variable MUST be set to the highest value of all *upnp:objectUpdateID* properties within the ContentDirectory service implementation.(See B.15.2, "*upnp:objectUpdateID* for details) If a ContentDirectory service implementation supports tracking changes but does not currently support tracking on any objects within its content hierarchy, then the initial value of the *SystemUpdateID* state variable MUST be zero ("0").

SystemUpdateID Increment Rules:

In some cases, a single action can trigger changes to multiple objects which will result in multiple increments of the *SystemUpdateID* state variable (one for each modified object). To simplify the processing of these changes, the following increment ordering rules are defined. Specifically, certain changes must affect the *SystemUpdateID* state variable before other changes:

- The creation of a container MUST increment the *SystemUpdateID* state variable prior to increments generated by the creation of any of that container's descendants.
- The creation of a *referenced item* MUST increment the *SystemUpdateID* state variable prior to an increment generated by the creation of any reference item(s) that refer to that specific r*eferenced item*. See subclause 2.2.21, "*reference*, *reference item*, *referenced item*" for details.
- The deletion of a container MUST increment the *SystemUpdateID* state variable only after the increment(s) generated by the deletion of all of its descendants.

2.3.5.2 *SystemUpdateID* when not Supporting the Tracking Changes Option

If the ContentDirectory service implementation does not support the *Tracking Changes Option*, then the actual value of *SystemUpdateID* state variable is unspecified. However, implementers SHOULD maintain the same value for the *SystemUpdateID* state variable through power cycles and any other disappearance/reappearance of the service on the network. Control points can use a change in the value of this variable to determine if there has been a change in the ContentDirectory service.

Note that the (OPTIONAL) *ContainerUpdateIDs* state variable provides more information about the scope of the change, since it takes advantage of the *ContainerUpdateIDValue* indicator maintained for each container.

2.3.6 *ContainerUpdateIDs*

This OPTIONAL state variable is an unordered CSV list of ordered pairs. Each pair consists of a container's *@id* property value and the value of its *ContainerUpdateIDValue* indicator, in that order, separated by a comma (","). *ContainerUpdateIDs* is a moderated evented state variable and is *only* used for eventing. There is no action that returns the value of *ContainerUpdateIDs*. The initial value of *ContainerUpdateIDs* is the empty string.

Each time a container is modified (see *Container Modification* in subclause 2.2.9, "Container Modification"), its *ContainerUpdateIDValue* indicator changes according to the rules in subclause 2.2.11, "*ContainerUpdateIDValue* Indicator" and the ordered pair of that container's *@id* property value and the value of its *ContainerUpdateIDValue* indicator is concatenated to the list, maintained in the *ContainerUpdateIDs* state variable. If that container's *@id* property value already appears in the *ContainerUpdateIDs* state variable, the new ordered pair is *not* added to the list. Instead, the value of the corresponding *ContainerUpdateIDValue* indicator that is already in the *ContainerUpdateIDs* state variable is replaced by the new value of the *ContainerUpdateIDValue* indicator. Consequently, there can be at most one occurrence in the

ContainerUpdateIDs state variable of an ordered pair with any given *@id* value. In other words, the evented value of the *ContainerUpdateIDs state variable* will never contain multiple ordered pairs with the same *@id* value. The *ContainerUpdateIDs* state variable is not a history list of container changes. Rather, the evented value will only reflect updates to the *ContainerUpdateIDs* state variable that occurred after the last event notification for this state variable.

The *ContainerUpdateIDs* state variable MUST NOT be cleared immediately after it has been evented. The *ContainerUpdateIDs* state variable MUST be cleared immediately *before* the first new (*@id* value, *ContainerUpdateIDValue* indicator value) pair is added to the *ContainerUpdateIDs* state variable following a *ContainerUpdateIDs* event message. The reason for this behavior is that if the *ContainerUpdateIDs* state variable were to be cleared immediately after eventing, then when the current moderation period ends, the empty list would be evented (because the *ContainerUpdateIDs* state variable changed since the last event message). This would falsely indicate a state change in the ContentDirectory service that did not actually occur.

Example 1: The following table shows a time-ordered sequence of actions on a ContentDirectory service implementation that does not support the *Tracking Changes Option* for a sequence of container modifications.

Table 2-5 — *ContainerUpdateIDs* Example

Action	*@id*	New value of *ContainerUpdateIDValue* ⇓	New value of *ContainerUpdateIDs*
Initialization	—	—	“” (empty)
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 does not change.
New control point signs up for events	—	—	Value does not change. The special event value unicast to the new control point includes the full set of 3 pairs
container modified	musicAlbum01	97	Value is first cleared, then set to “musicAlbum01,97”

Example 2: The following table shows a time-ordered sequence of actions on a ContentDirectory service implementation that supports the *Tracking Changes Option* for a sequence of container modifications. Note that the values of the *ContainerUpdateIDValue* indicator now store a sequence of *SystemUpdateID* state variable values and are not independently incremented. This example assumes that the only changes that are happening are to the containers of the example so that their *ContainerUpdateIDValue* indicators are monotonically increasing.

Table 2-6 — *ContainerUpdateIDs* Example

Action	*@id*	New value of *ContainerUpdateIDValue* ⇓	New value of *ContainerUpdateIDs*
Initialization	—	—	“” (empty)
container modified	musicAlbum15	53	“musicAlbum15,53”
container modified	photoAlbum28	54	“musicAlbum15,53,photoAlbum28,54”
container modified	musicAlbum15	55	“musicAlbum15,55,photoAlbum28,54”
container modified	musicAlbum11	56	“musicAlbum15,55,photoAlbum28,54,

			musicAlbum11,56”
ContainerUpdateIDs is evented	—	—	Value does not change.
New control point signs up for events	—	—	Value does not change. The special event value unicast to the new control point includes the full set of 3 pairs
container modified	musicAlbum01	57	Value is first cleared, then set to “musicAlbum01,57”

2.3.7 *ServiceResetToken*

This REQUIRED state variable contains a non-empty value that MUST be unique over the lifetime of this ContentDirectory service implementation and MUST be persisted over periods when the ContentDirectory service is *off-line*. For example, a *ServiceResetToken* value can be used that is initialized to 0 and subsequently incremented whenever the value must be changed. Alternatively, a sequence of GUIDs can be used.

The specific value of the *ServiceResetToken* state variable is not important. Rather, it is a change in the value that is significant. A change in this state variable indicates that the ContentDirectory service implementation can no longer maintain a consistent progression of internal state. When this occurs the implementation MUST invoke the *Service Reset Procedure* and assign a different permanently unique token to the *ServiceResetToken* state variable. See subclause 2.3.7.1, “*Service Reset Procedure*” for details.

When a change in the value of the *ServiceResetToken* state variable occurs, control points can no longer rely on any values that they have cached from the ContentDirectory service.

The value of the *ServiceResetToken* state variable MUST only be changed upon invocation of the *Service Reset Procedure*. This involves removing the ContentDirectory service from the network. Therefore, control points are recommended to check for a change in value of the *ServiceResetToken* state variable when either the control point or the device (re)connects to the network.

2.3.7.1 *Service Reset Procedure*

The *Service Reset Procedure* consists of the following steps in sequence:

- The device MUST immediately disconnect from the network by sending a “bye-bye” message as described in subclause 1.1.3 “Discovery:Advertisement:Device Unavailable – NOTIFY with ssdp:byebye” of the UPnP Device Architecture Specification [DEVICE].
- The *ServiceResetToken* state variable MUST be assigned a new never-been-seen-before permanently unique token.
- All *upnp:objectUpdateID* properties MUST be reset according to the initialization requirements defined in B.15.2, “*upnp:objectUpdateID*”.
- All *upnp:containerUpdateID* properties MUST be reset according to the initialization requirements defined in B.15.1, “*upnp:containerUpdateID*”.
- All *upnp:totalDeletedChildCount* properties MUST be reset according to the initialization requirements defined in B.15.3, ”*upnp:totalDeletedChildCount*”.
- All *res@updateCount* properties MUST be reset according to the initialization requirements defined in B.15.4, “*res@updateCount*”.
- The *SystemUpdateID* state variable MUST be set to the highest value of all *upnp:objectUpdateID* properties within the ContentDirectory service.
- The ContentDirectory service implementation MAY create new or re-assign *@id* property values to some or all of the objects within the ContentDirectory hierarchy.
- The device may then reconnect to the network.

2.3.8 *LastChange*

The *LastChange* state variable is REQUIRED when the ContentDirectory service implementation supports the *Tracking Changes Option*. Otherwise, it is PROHIBITED. It contains a *LastChange XML Document* identifying *all* changes that have occurred since the last time the *LastChange* state variable was evented. It is used to event changes that are not directly related to one of the state variables of the ContentDirectory service; that is: changes made to the properties of an object. See subclause 2.4, "Eventing and Moderation" for details. For every type of change that is defined in the XML schema for the *LastChange* state variable, an implementation MUST generate an event whenever that type of change occurs. Additionally, individual events MUST be buffered and delivered in the order that they occurred with the most recent event corresponding to the last XML element within the *LastChange XML Document* that is stored in the *LastChange* state variable. Refer to subclause 2.3.8.1, "*LastChange* Data Format" and the ContentDirectory service Event Schema document [CDS-EVENT-XSD] for more details.

The *LastChange* state variable is evented and moderated according to the GENA eventing mechanism as defined by the UPnP Device Architecture Specification [DEVICE]. When multiple object modifications occur within the same moderation period (as determined by the implementation), each change MUST be accumulated in the *LastChange* state variable and MUST be evented as a single event notification message after the current moderation period expires. After the event notification message has been sent to all subscribed control points, the value of the *LastChange* state variable is reset when an update to the *LastChange* state variable becomes necessary; that is: when the next event occurs. The resulting value is a fresh *LastChange XML Document* that contains a single element that represents the update (that is: it contains the first update event following the distribution of the previous event message to all subscribers). Subsequently, additional update elements are added to the *LastChange XML Document* until the current moderation period ends and the current value of the *LastChange* state variable (i.e. the current event message) is propagated to all event subscribers.

The *LastChange* state variable is NOT REQUIRED to accumulate changes when the ContentDirectory service is *off-line* nor when the ContentDirectory service has no subscribers for events. When the ContentDirectory service comes *on-line*, the *LastChange* state variable MAY be empty. It is NOT REQUIRED to event changes that had been accumulated but not evented when the ContentDirectory service last went *off-line*.

Note: the *LastChange* state variable contains event information about all object changes within the ContentDirectory hierarchy regardless of whether the objects contain the *upnp:objectUpdateID* or *upnp:containerUpdateID* properties.

2.3.8.1 *LastChange* Data Format

The optional XML header `<?xml version="1.0" ?>` is allowed. The (one and only) root element, `<StateEvent>`, MUST contain zero or more elements, each of which represents a change to a specific object. As shown below, three types of elements are defined to indicate the type of change that occurred on that object: an object creation, modification, or deletion.

The following example shows a generalized "template" for the format of the *LastChange* state variable. Additional elements and/or attributes MAY be added to future versions of this specification. Furthermore, a 3rd-party vendor MAY add vendor-defined elements or attributes. However, by definition, this specification does not define the format or the values for these 3rd-party elements. In order to eliminate element or attribute naming conflicts, the name of any vendor-defined element or attribute MUST follow the rules set forth in subclause 1.5, "Vendor-defined Extensions". All control points should gracefully ignore any element or attribute that it does not understand.

Note: The content of this state variable (that is: the *LastChange XML Document*) MUST be properly escaped before it is sent to an event subscriber via GENA. See subclause 2.4 - Character Data and Markup in [XML] for more details.

The following example shows fields that need to be filled out by individual implementations in the *vendor* character style.

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:cds-event
      http://www.upnp.org/schemas/av/cds-events.xsd">
   <objAdd
    objID="object ID (@id property) of the added object"
    updateID="Resulting value of the SystemUpdateID state variable"
    objParentID="object ID (@id property) of the new object's parent"
    objClass="class of the object(upnp:class property)"
    stUpdate="subtree update flag"/>

   <objMod
    objID="object ID (@id property) of the modified object"
    updateID="Resulting value of the SystemUpdateID state variable"
    stUpdate="subtree update flag"/>

   <objDel
    objID="object ID (@id property) of the deleted object"
    updateID="Resulting value of the SystemUpdateID state variable"
    stUpdate="subtree update flag"/>

   <stDone
    objID="object ID (@id property) of the subtree root container"
    updateID="Resulting value of the SystemUpdateID state variable"/>
</StateEvent>
```

`<xml>`
OPTIONAL. Case sensitive.

`<StateEvent>`
REQUIRED. MUST include a namespace declaration for the ContentDirectory service Event Schema ("urn:schemas-upnp-org:av:cds-event"). MUST include zero or more of the following elements. This namespace defines the following elements and attributes:

`<objAdd>`
OPTIONAL. Indicates that an object was added to the ContentDirectory service within the most recent event moderation period. See subclause 2.4, "Eventing and Moderation" for details on the event moderation period. MUST appear once for each object added. The contents of this element MUST be the empty string. However, future versions of this specification may define specific values for this element. Consequently, control points must be prepared to gracefully ignore any element contents or element attributes that it does not understand. Contains all of the following attributes:

`objID`
REQUIRED. xsd:string, Contains the *@id* property of the object that was added.

`updateID`
REQUIRED. xsd:unsignedInt, Contains the value of the *SystemUpdateID* state variable that resulted when the object was added.

`stUpdate`
REQUIRED. xsd:boolean, Indicates whether or not the object was added as part of a subtree update operation. A value of "1" (one) indicates that the object was added as part of a subtree update operation. A value of "0" (zero) indicates that the object WAS NOT added as part of a subtree update operation but rather it was added as an individual object addition. See subclause 2.2.15, "Subtree Updates" for details.

`objParentID`
REQUIRED. xsd:string, Contains the *@id* property of the parent container to which this object was added. This information may be useful for control points to determine if this new object is of interest.

`objClass`
REQUIRED. xsd:string, Contains the value of the upnp:class property of the object was added. This information may be useful for control points to determine if this new object is of interest.

`<objMod>`
OPTIONAL. Indicates that an existing object was modified within the most recent event moderation period. See subclause 2.4, “Eventing and Moderation” for details on the event moderation period. MUST appear once for each object that was modified. The contents of this element MUST be the empty string. However, future versions of this specification may define specific values for this element. Consequently, control points must be prepared to gracefully ignore any element contents or element attributes that it does not understand. Contains all of the following attributes:

`objID`
REQUIRED. xsd:string, Contains the *@id* property of the object that was modified.

`updateID`
REQUIRED. xsd:unsignedInt, Contains the value of the *SystemUpdateID* state variable that resulted when the object was modified

`stUpdate`
REQUIRED. xsd:boolean, Indicates whether or not the object was modified as part of a subtree update operation. A value of “1” (one) indicates that the object was modified as part of a subtree update operation. A value of “0” (zero) indicates that the object WAS NOT modified as part of a subtree update operation but rather it was modified as an individual object modification. See subclause 2.2.15, “Subtree Updates” for details.

`<objDel>`
OPTIONAL. Indicates that an object was deleted from the ContentDirectory service within the most recent event moderation period. See subclause 2.4, “Eventing and Moderation” for details on the event moderation period. MUST appear once for each object deleted. The contents of this element MUST be the empty string. However, future versions of this specification may define specific values for this element. Consequently, control points must be prepared to gracefully ignore any element contents or element attributes that it does not understand. Contains all of the following attributes:

`objID`
REQUIRED. xsd:string, Contains the *@id* property of the object that was deleted.

`updateID`
REQUIRED. xsd:unsignedInt, Contains the value of the *SystemUpdateID* state variable that resulted when the object was deleted.

`stUpdate`
REQUIRED. xsd:boolean, Indicates whether or not the object was deleted as part of a subtree update operation. A value of “1” (one) indicates that the object was deleted as part of a subtree update operation. A value of “0” (zero) indicates that the object WAS NOT deleted as part of a subtree update operation but rather it was deleted as an individual object deletion. See subclause 2.2.15, “Subtree Updates” for details.

`<stDone>`
OPTIONAL. Indicates that a sub-tree update operation has completed within the most recent event moderation period. See subclause 2.4, “Eventing and Moderation” and subclause 2.2.15, “Subtree Updates” for details on the event moderation period and subtree update operations. MUST appear once for each completed subtree update operation. The contents of this element MUST be the empty string. However, future versions of this specification may define specific values for this element. Consequently, control points must be prepared to gracefully ignore any element contents or element attributes that it does not understand. Contains all of the following attributes:

`objID`
REQUIRED. xsd:string, Contains the value of the *@id* property of the container object that represents the root of the updated subtree.

`updateID`
REQUIRED. xsd:unsignedInt, Contains the value of the *SystemUpdateID* state variable when the subtree update operation completed.

Note: Additional elements or attributes may also be present, for example, defined by individual vendors or future versions of the ContentDirectory service specification. Consequently, a control point must gracefully ignore any additional elements or attributes that it does not understand.

2.3.8.2 Event Ordering Rules

Events in the *LastChange XML Document* MUST be ordered according to increasing numeric values of their `updateID` attributes.

In some cases, a single action can trigger changes to multiple objects which will result in multiple events (one for each modified object. To simplify the processing of those events, the following event ordering rules are defined. Specifically, certain events MUST be added to the event buffer (while waiting for the moderation period to expire - See subclause 2.4, “Eventing and Moderation”) before other related events.

- An `<objAdd>` event corresponding to the creation of a container MUST precede all `<objAdd>` event(s) corresponding to the creation of any of that container’s descendants.
- An `<objAdd>` event corresponding to the creation of a *referenced item* MUST precede all `<objAdd>` event(s) corresponding to the creation of any *reference item(s)* that refer to that specific *referenced item*. See subclause 2.2.21, “*reference*, *reference item*, *referenced item*” for details.
- An `<objDel>` event corresponding to the deletion of a container MUST NOT precede any `<objDel>` event(s) corresponding to the deletion of any of its descendants.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
   <objAdd objID="s002" updateID="213" objParentID="s001"
    objClass=”object.container.album” stUpdate=”1”/>
   <objMod objID="s001" updateID="214" stUpdate=”1”/>
   <objAdd objID="s003" updateID="215" objParentID="s001"
    objClass=”object.item.audioItem” stUpdate=”0”/>
   <objAdd objID="s004" updateID="216" objParentID="s002"
    objClass=”object.item.audioItem” stUpdate=”1”/>
   <objDel objID="s003" updateID="217" stUpdate=”0”/>
   <objMod objID="s001" updateID="218" stUpdate=”0”/>
   <objMod objID="s004" updateID="219" stUpdate=”1”/>
   <stDone objID="s001" updateID="219"/>
</StateEvent>
```

2.3.9 *TransferIDs*

This state variable 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 the *TransferIDs* list.

This state variable is used for eventing only.

2.3.10 *FeatureList*

This state variable enumerates the *CDS features* supported by this ContentDirectory service. The value is a valid *Features XML Document*, according to [AVS-XSD]:

- The root element of the document is `<Features>`. It contains zero or more child `Feature` elements, each of which represents one ContentDirectory service feature that is supported in this implementation.
- A `<Feature>` element MUST have a `version` attribute.

- A <Feature> element MAY have other attributes defined per each feature.
- See the schema in [AVS-XSD] for more details on the structure.

Example (this string must be escaped when transmitted in a SOAP response message):

```
<?xml version="1.0" encoding="UTF-8"?>
<Features
 xmlns="urn:schemas-upnp-org:av:avs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:avs
    http://www.upnp.org/schemas/av/avs.xsd">
   <Feature name="BOOKMARK" version="1">
      <objectIDs>bookmark1</objectIDs>
   </Feature>
   <Feature name="EPG" version="1">
      <objectIDs>epg1,epg2</objectIDs>
   </Feature>
</Features>
```

2.3.11 *A_ARG_TYPE_ObjectID*

This state variable is introduced to provide type information for the *ObjectID* argument in various actions. The *ObjectID* argument uniquely identifies individual objects within the ContentDirectory service.

2.3.12 *A_ARG_TYPE_Result*

This state variable is introduced to provide type information for the *Result* argument in various actions. The structure of the *Result* argument is a *DIDL-Lite XML Document*:

- Optional XML declaration <?xml version="1.0" ?>
- <DIDL-Lite> is the root element.
- <container> is the element representing objects of class *container* and all its derived classes.
- <item> is the element representing objects of class *item* and all its derived classes.
- Elements in the Dublin Core (dc) and UPnP (upnp) namespaces represent object metadata.
- See the DIDL-Lite schema [DIDL-LITE-XSD] for more details on the structure. The available properties and their names are described in Annex B, “ (normative)

  AV Working Committee Properties”.

Note that since the value of *Result* is XML, it needs to be escaped (using the normal XML rules: [XML] subclause 2.4 Character Data and Markup) before embedding in a SOAP response message. In addition, when a value of type *A_ARG_TYPE_Result* is employed in a CSV list, commas (“,”) that appear within XML CDATA MUST be escaped as “\,”. See subclause 1.2.2, “Strings Embedded in Other Strings”

2.3.13 *A_ARG_TYPE_SearchCriteria*

This state variable is introduced to provide type information for the *SearchCriteria* argument in the *Search()* action. The *SearchCriteria* argument provides one or more search criteria to be used for querying the ContentDirectory service. All property names MUST be fully qualified using the double colon (“::”) syntax as defined in subclause 2.2.20, “property”. For example, “upnp:foreignMetadata::fmBody::fmURI”.

Each property name MUST include the standard namespace prefix for that property, except for the DIDL-Lite namespace. Properties in the DIDL-Lite namespace MUST always be returned without the prefix.

2.3.13.1 *SearchCriteria* String Syntax

SearchCriteria string syntax is described here formally using EBNF as described in subclause 1.2.3, "Extended Backus-Naur Form".

```
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 subclause 2.2.20 *)
escapedQuote        ::=  (* double-quote escaped string as defined in subclause
                         1.2.2 *)
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 *)
```

2.3.13.2 *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)`"

  Likewise,

  "`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 (that is: 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 ContentDirectory service 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.

2.3.14 *A_ARG_TYPE_BrowseFlag*

This state variable is introduced to provide type information for the *BrowseFlag* argument in the *Browse()* action. A *BrowseFlag* argument specifies a browse option to be used for browsing the ContentDirectory service. Valid values are:

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

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

2.3.15 *A_ARG_TYPE_Filter*

This state variable is introduced to provide type information for the *Filter* argument in the *Browse()* and *Search()* actions. The comma-separated list of property specifiers indicates which metadata properties are to be returned in the *Result* of the *Browse()* or *Search()* actions. Each property name MUST include the standard namespace prefix for that property, except for the DIDL-Lite namespace. Properties in the DIDL-Lite namespace MUST always be returned without the prefix. All property names MUST be fully qualified using the double colon ("::") syntax as defined in subclause 2.2.20, "property". For example, "`upnp:foreignMetadata::fmBody::fmURI`".

The *Filter* argument allows control points to control the complexity of the object metadata properties that are returned within the DIDL-Lite *Result* argument of the *Browse()* and *Search()* actions. Properties REQUIRED by the DIDL-Lite schema are always returned in the *Result* output argument. The *Filter* argument allows a control point to specify additional properties, not REQUIRED by the DIDL-Lite schema to be returned in *Result*. Compliant ContentDirectory service implementations do not return optional properties unless they are explicitly requested in the *Filter* input argument.

Both independent and dependent properties MAY be included in the comma-separated *Filter* argument. If the *Filter* argument is equal to "*", all supported properties, both REQUIRED and OPTIONAL, from all namespaces are returned.

A compliant ContentDirectory service implementation MUST also ignore optional properties requested in the *Filter* input argument, which are not actually present in the matching objects. For example, a *Browse()* *Filter* input argument of the form "`dc:creator`" is successful and returns a DIDL-Lite *Result* value that complies with the other *Browse()* input arguments, even in the case that the objects represented in *Result* do not have a *dc:creator* property defined.

In all cases, a compliant ContentDirectory service implementation MUST always respond to *Search()* and *Browse()* requests with the smallest, valid *DIDL-Lite XML Document* in the *Result* argument that satisfies the *Filter* input argument. In some cases, a ContentDirectory service MUST add properties that are not specified in the *Filter* argument so that the resulting *DIDL-Lite XML Document* is valid. If the XML document can not be made valid by adding other properties, the offending properties in the *Filter* argument MUST be ignored by the ContentDirectory service.

Example 1: The *Filter* argument in a *Search()* action is specified as "`res@size`", indicating that the optional *res@size* property, if present, MUST be returned in the results of the *Search()*.

```
Request:
Search("0", "dc:title contains "tenderness"", "res@size", 0, 1, "")
```

The ContentDirectory service responds with the smallest, valid *DIDL-Lite XML Document* in the *Result* argument that satisfies the *Filter* argument, as follows:

```
Response:
Search("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="18" parentID="13" restricted="0">
      <dc:title>Try a little tenderness</dc:title>
      <upnp:class>object.item.audioItem.musicTrack</upnp:class>
      <res protocolInfo="http-get:*:audio/mpeg:*" size="3558000">
         http://168.192.1.1/audio197.mp3
      </res>
   </item>
</DIDL-Lite>", 1, 1, 2345)
```

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. In Example 1, since *dc:title* and *upnp:class* are REQUIRED properties for both item and container objects, the *dc:title* and *upnp:class* elements are automatically included in all item and container objects in the *Result*. The REQUIRED *res@protocolInfo* property is also automatically included in the *Result*. (Note that the REQUIRED inclusion of the dependent *res@protocolInfo* property forces the inclusion of its associated independent *res* property.)

Example 2: The *Filter* argument in a *Search()* action is specified as "`upnp:longDescription,dc:creator`", indicating that the optional *upnp:longDescription* and *dc:creator* properties MUST be included in the DIDL-Lite *Result* returned for each object.

```
Request:
Search("0", "dc:title contains "tenderness"",
   "upnp:longDescription,dc:creator", 0, 1, "")
```

The ContentDirectory service responds with the smallest, valid *DIDL-Lite XML Document* that satisfies the other *Search()* arguments and the specified *Filter* argument, as follows:

```
Response:
Search("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="18" parentID="13" restricted="0">
      <dc:title>Try a little tenderness</dc:title>
      <upnp:class>object.item.audioItem.musicTrack</upnp:class>
      <upnp:longDescription>
         This song is considered to be the finest R&amp;B tune ever
      </upnp:longDescription>
      <dc:creator>Otis Redding</dc:creator>
   </item>
</DIDL-Lite>", 1, 1, 2345)
```

2.3.16 *A_ARG_TYPE_SortCriteria*

This state variable is introduced to provide type information for the *SortCriteria* argument in the *Browse()* and *Search()* actions. *A_ARG_TYPE_SortCriteria* is a possibly empty CSV list of property names, each of which MUST be prefixed by a sort modifier. Each property name MUST include the standard namespace prefix for that property, except for the DIDL-Lite namespace. Properties in the DIDL-Lite namespace MUST always be returned without the prefix. All property names MUST be fully qualified using the double colon ("::") syntax as defined in subclause 2.2.20, "property". For example, "`upnp:foreignMetadata::fmBody::fmURI`".

Sort modifiers indicate whether the prefixed property is to be sorted in ascending or descending order. They may also indicate that the sort process should use some special interpretation of the property's value. See subclause 2.3.4 "*SortExtensionCapabilities*" for detailed information about sort modifiers. 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 by artist in ascending order, then within each artist by date in descending order (most recent first) and finally by title in ascending order.

When a device receives a *SortCriteria* argument using unsupported sort modifiers, it MUST return with error code 709, "Unsupported or invalid sort criteria".

When a *SortCriteria* argument contains property names of optional and/or multi-valued or CSV list properties, the following rules apply:

If the property is prefixed by "+" then:

- Objects that do not have a value for the property are returned first in their group.

- Objects that have at least one value for the property are returned next in their group. Objects that have multiple values for the property (either multi-valued or CSV list) are sorted based on the property value that would cause the object to appear earliest in the list.

If the property is prefixed by "-" then:

- Objects that have at least one value for the property are returned first in their group. Objects that have multiple values (either multi-valued or CSV list) for the property are sorted based on the property value that would cause the object to appear earliest in the list.
- Objects that do not have a value for the property are returned last in their group.

Depending on the property, the sort operation uses the semantics of the property, rather than the alphabetical order of the values of that property. Note that alphabetical sorting should not be based on Unicode character values but rather based on localized lexical conventions. For example, the "ö" character in German sorts between "n" and "p" characters whereas in Swedish, it sorts after "z". See [UNICODE COLLATION].

When an empty string is specified, then the order is device dependent. Additionally, this device dependent ordering MUST remain constant unless the *SystemUpdateID* value has changed since the last action invocation. In other words, any two objects that appear in a *Result* argument MUST always appear in the same relative order as long as the *SystemUpdateID* value did not change.

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

2.3.17 *A_ARG_TYPE_Index*

This state variable is introduced to provide type information for the *Index* argument in various actions. *Index* arguments specify an offset into an arbitrary list of objects. A value of 0 represents the first object in the list.

2.3.18 *A_ARG_TYPE_Count*

This state variable is introduced to provide type information for the *Count* argument in various actions. *Count* arguments specify an ordinal number of arbitrary objects.

2.3.19 *A_ARG_TYPE_UpdateID*

This state variable is introduced to provide type information for the *UpdateID* output argument in various actions, such as the *Browse()* and *Search()* actions. The returned value will always be a *SystemUpdateID* state variable value and therefore the *A_ARG_TYPE_UpdateID* type definition is identical to the *SystemUpdateID* type. (**ui4**). (see subclause 2.3.5, "*SystemUpdateID*")

2.3.20 *A_ARG_TYPE_TransferID*

This state variable is introduced to provide type information for the *TransferID* argument in various actions. The *TransferID* argument uniquely identifies individual file transfers initiated by the *ImportResource()* or the *ExportResource()* action of the ContentDirectory service. The *TransferID* is a unique value assigned by the device.

2.3.21 *A_ARG_TYPE_TransferStatus*

This state variable is introduced to provide type information for the *TransferStatus* argument in various actions. This variable MAY assume one of the enumerated values: "*IN_PROGRESS*", "*STOPPED*", "*ERROR*", or "*COMPLETED*", indicating the status of a file transfer.

2.3.22 *A_ARG_TYPE_TransferLength*

This state variable is introduced to provide type information for the *TransferLength* argument in various actions. Its data type is **string**, representing a numerical value that MAY exceed 32 bits in size.

2.3.23 *A_ARG_TYPE_TransferTotal*

This state variable is introduced to provide type information for the *TransferTotal* argument in various actions. Its data type is **string**, representing a numerical value that MAY exceed 32 bits in size.

2.3.24 *A_ARG_TYPE_TagValueList*

This state variable is introduced to provide type information for the *CurrentTagValue* and *NewTagValue* arguments in the *UpdateObject()* action. It is a CSV list of *DIDL-Lite 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 subclause 1.3.1, “Comma Separated Value (CSV) Lists”.

2.3.25 *A_ARG_TYPE_URI*

This state variable is introduced to provide type information for the *URI* argument in various actions. *URI* IN or OUT arguments in ContentDirectory service actions MUST be properly escaped URIs as described in [RFC 2396]. In addition, *URI* arguments MUST be escaped according to the requirements of RFC1738 (http://www.ietf.org/rfc/rfc1738.txt).

2.3.26 *A_ARG_TYPE_CDSView*

The state variable is introduced to provide type information for the *CDSView* argument in various actions, such as the *FreeFormQuery()* action. The data type is **ui4** and allowed values are:

- “0”: DIDL-Lite View.
- All other values are reserved for future extensions.

2.3.27 *A_ARG_TYPE_QueryRequest*

The state variable is introduced to provide type information for the *QueryRequest* argument in various actions, such as the *FreeFormQuery()* action. The data type is **string** and contains an XML-formatted document that MUST comply with the W3C XQuery 1.0 language recommendation. See [XQUERY10]. In addition, the remainder of this subclause describes additional rules that MUST be followed when constructing XQuery requests.

The namespaces used in the *QueryRequest* argument MUST be part of the ones supported by the implementation, as is indicated by the *FFQCapabilities* argument of the *GetFreeFormQueryCapabilities()* action (see subclause 2.5.20, “*GetFreeFormQueryCapabilities()*”). All other namespaces (even when present in the CDS view), MUST NOT be used in this argument.

The properties used in the *QueryRequest* argument MUST be the ones supported by the implementation, as is indicated by the *FFQCapabilities* argument of the *GetFreeFormQueryCapabilities()* action (see subclause 2.5.20, “*GetFreeFormQueryCapabilities()*”).All other property names MUST NOT be used in this argument. The content of each `<propertyName>` element (see subclause 2.3.29, “*A_ARG_TYPE_FFQCapabilities*”), when used, MUST be used in the *QueryRequest* argument as follows:

- Property names that do not contain the “@” symbol (i.e. do not appear as attributes in the DIDL-Lite view) can be used as is (i.e. including the entire path of the property

name, for example: “didl-lite:item/dc:title”, “didl-lite:item/upnp:foreignMetadata/upnp:fmBody/upnp:fmURI”). Alternatively, some components of the path may be left out and replaced by double slashes (“//”), for example: “//upnp:fmURI”. The double slashes (“//”) symbol is defined by XQuery and indicates that the specified property may appear anywhere in the DIDL-Lite document. The property name itself (the last component of the path) MUST be present as this is the property supported by the implementation.

- Property names containing the “@” symbol can be used with or without their path prefix, as long as the context in which this property is used is properly defined in the XQuery request. For example, assume that the *GetFreeFormQueryCapabilities()* action returned “`didl-lite:item/@id`” in one of the `<propertyName>` elements but not “`didl-lite:container/@id`”. In this case, a valid use of the @id property is: `//didl-lite:item[@id = "Some item ID"]`

  On the other hand, an invalid use of the @id property is: `//didl-lite:container[@id = "Some container ID"]` since @id is incorrectly used in the context of a container, which was not supported by the implementation, as indicated by the absence of the “`didl-lite:container/@id`” in a `<propertyName>` element.

Example: This request example creates a *DIDL-Lite XML Document* that contains all items whose class is “object.item.audioItem” but only those within a container named “Album 1”.

```
<DIDL-Lite
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:didl-lite="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/">
{
   for $object in //didl-lite:item[upnp:class = "object.item.audioItem"]
   let $containerId := $object/@parentID
   where
      //didl-lite:container[@id=$containerId and dc:title="Album 1"]
   return $object
}
</DIDL-Lite>
```

Note that since the *QueryRequest* argument contains an XML document it MUST be properly escaped (using the normal XML rules: subclause 2.4 Character Data and Markup) when transmitted in a SOAP message.

2.3.28 *A_ARG_TYPE_QueryResult*

This state variable is introduced to provide type information for the *QueryResult* argument in various actions, such as the *FreeFormQuery()* action. Contrary to the structure of the *Result* output argument of the *Browse()* and *Search()* actions, the structure of the *QueryResult* argument is defined by the XQuery request. Depending on the *XQuery XML Document*, specified in the *QueryRequest* argument, it may contain a valid *XML Document*, or any other text output.

Note that since the value of the *QueryResult* argument may contain XML nodes, it MUST be properly escaped (using the normal XML rules: subclause 2.4 Character Data and Markup) when transmitted in a SOAP message.

2.3.29 *A_ARG_TYPE_FFQCapabilities*

This state variable is introduced to provide type information for the *FFQCapabilities* argument in various actions, such as the *GetFreeFormQueryCapabilities()* action. The structure of the *FFQCapabilities* argument is a *FFQCapabilities XML Document*. The optional XML header `<?xml version="1.0" ?>` is allowed. The (one and only) root element is `<Capabilities>`, which MUST contain exactly one `<namespaceList>` element and exactly

one <propertyList> element. <namespaceList> contains a flat list of <namespaceName> elements. <propertyList> contains a flat list of <propertyName> elements. There are no specific ordering requirements on the occurrence of the <namespaceList> and <propertyList> elements. See the *FFQCapabilities* schema [AVS-XSD] for details.

The following example shows a generalized “template” for the format of the *FFQCapabilities* XML Document. The example shows fields that need to be filled out by individual implementations in the *vendor* character style.

```
<?xml version="1.0" encoding="UTF-8"?>
<Capabilities
 xmlns="urn:schemas-upnp-org:av:avs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:avs
       http://www.upnp.org/schemas/av/avs.xsd">
   <namespaceList>
       <namespaceName>
          Name of a namespace supported by this implementation
       </namespaceName>
       <namespaceName>
          Name of a namespace supported by this implementation
       </namespaceName>
   </namespaceList>
   <propertyList>
       <propertyName>
          Name of a property supported by this implementation
       </propertyName>
       <propertyName>
          Name of a property supported by this implementation
       </propertyName>
   </propertyList>
</Capabilities>
```


<xml>
OPTIONAL. Case sensitive.

<Capabilities>
REQUIRED. MUST include a namespace declaration for the ContentDirectory service Common Datastructures Schema (“urn:schemas-upnp-org:av:avs”). MUST include exactly one instance of each of the following elements (in no specific order):

<namespaceList>
REQUIRED. MUST appear exactly once. The contents of this element MUST contain one or more of the following elements:

<namespaceName>
REQUIRED. xsd:string. Identifies the name of a particular namespace (including its corresponding prefix) that may be used within the *QueryRequest* argument of the *FreeFormQuery()* action. The format of this element is:

<prefix> “=” <namespace name>

where

<prefix> MUST be either one of the namespace prefixes defined in Table 1-4, “Schema-related Information”, or a vendor-defined namespace prefix.

<namespace name> is the name of the namespace without the double quotes (“”) that normally appear in an “xlmns” declaration.

Example: upnp=urn:schemas-upnp-org:metadata-1-0/upnp/

<propertyList>
REQUIRED. MUST appear exactly once. The contents of his element MUST contain one or more of the following elements:

<propertyName>
REQUIRED. xsd:string. Identifies the name of a particular property that may be used within the *QueryRequest* argument of the *FreeFormQuery()* action. The property name MUST be fully qualified using the XQuery path expressions syntax [XQUERY10], using the slash ("/") symbol to separate different components in nested properties and dependent properties (see subclause 2.2.20, "property"). Each component in the path may consist of the name of an XML element (in case of independent properties) or attribute (in case of dependent properties) in the DIDL-Lite view of the ContentDirectory service (subclause 2.2.19.1, "*DIDL-Lite View*"). Element names MUST include their appropriate namespace prefixes. Attribute names MUST be preceded by the "@" symbol.

Examples:

"didl-lite:item/dc:title", "didl-lite:item/upnp:artist".

"didl-lite:item/upnp:foreignMetadata/upnp:fmBody/upnp:fmURI", "didl-lite:item/upnp:foreignMetadata/upnp:fmBody/upnp:fmEmbeddedXML/tva:TVAMain/tva:ProgramDescription/tva:ProgramLocationTable/tva:ScheduleEvent/tva:PublishedStartTime".

"didl-lite:item/@id", "didl-lite:container/@parentID".

How the property names listed in the <propertyName> elements can be used in the FreeFormQuery() action is described in subclause 2.3.27, "*A_ARG_TYPE_QueryRequest*".

Note that since the value of *FFQCapabilities* is XML, it needs to be properly escaped (using the normal XML rules: [XML] subclause 2.4 Character Data and Markup) before embedding in a SOAP response message.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Capabilities xmlns="urn:schemas-upnp-org:av:avs"
 xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xsi:schemaLocation="urn:schemas-upnp-org:av:avs
   http://www.upnp.org/schemas/av/avs.xsd">
   <namespaceList>
      <namespaceName>
         dc=http://purl.org/dc/elements/1.1/
      </namespaceName>
      <namespaceName>
         upnp=urn:schemas-upnp-org:metadata-1-0/upnp/
      </namespaceName>
      <namespaceName>
         didl-lite=urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
      </namespaceName>
   </namespaceList>
   <propertyList>
      <propertyName>didl-lite:item/dc:title</propertyName>
      <propertyName>didl-lite:item/upnp:class</propertyName>
      <propertyName>didl-lite:item/upnp:genre</propertyName>
      <propertyName>didl-lite:item/upnp:album</propertyName>
      <propertyName>didl-lite:item/upnp:artist</propertyName>
      <propertyName>
         didl-lite:item/upnp:foreignMetadata/upnp:fmBody/upnp:fmURI
      </propertyName>
      <propertyName>didl-lite:item/@id</propertyName>
      <propertyName>didl-lite:container/@parentID</propertyName>
   </propertyList>
</Capabilities>
```


2.4 Eventing and Moderation

Table 2-7 — Event moderation

Variable Name	Evented	Moderated Event	Max Event Rate [a]	Logical Combination	Min Delta per Event [b]
TransferIDs	*YES*	*NO*			
A_ARG_TYPE_ObjectID	*NO*	*NO*			
A_ARG_TYPE_Result	*NO*	*NO*			
A_ARG_TYPE_SearchCriteria	*NO*	*NO*			

Variable Name	Evented	Moderated Event	Max Event Rate [a]	Logical Combination	Min Delta per Event [b]
A_ARG_TYPE_SortCriteria	*NO*	*NO*			
A_ARG_TYPE_UpdateID	*NO*	*NO*			
A_ARG_TYPE_BrowseFlag	*NO*	*NO*			
A_ARG_TYPE_Filter	*NO*	*NO*			
A_ARG_TYPE_Index	*NO*	*NO*			
A_ARG_TYPE_Count	*NO*	*NO*			
A_ARG_Type_TransferID	*NO*	*NO*			
A_ARG_Type_TransferStatus	*NO*	*NO*			
A_ARG_Type_TransferLength	*NO*	*NO*			
A_ARG_Type_TransferTotal	*NO*	*NO*			
A_ARG_TYPE_TagValueList	*NO*	*NO*			
A_ARG_TYPE_URI	*NO*	*NO*			
A_ARG_TYPE_CDSView	*NO*	*NO*			
A_ARG_TYPE_QueryRequest	*NO*	*NO*			
A_ARG_TYPE_QueryResult	*NO*	*NO*			
A_ARG_TYPE_FFQCapabilities	*NO*	*NO*			
SearchCapabilities	*NO*	*NO*			
SortCapabilities	*NO*	*NO*			
SortExtensionCapabilities	*NO*	*NO*			
FeatureList	*NO*	*NO*			
SystemUpdateID	*YES*	*YES*	0.2 seconds		
ContainerUpdateIDs	*YES*	*YES*	0.2 seconds		
ServiceResetToken	*NO*	*NO*			
LastChange	*YES*	*YES*	0.2 seconds		
Non-standard state variables implemented by a UPnP vendor go here	*TBD*	*TBD*	*TBD*	*TBD*	*TBD*

a Determined by N, where Rate = (Event)/(N seconds).

b (N) * (allowedValueRange Step)

2.5 Actions

The following tables and subclauses define the various ContentDirectory service actions.

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

Table 2-8 — Actions

Name	R/O [a]
GetSearchCapabilities()	*R*
GetSortCapabilities()	*R*
GetSortExtensionCapabilities()	*O* [b]
GetFeatureList()	*R*
GetSystemUpdateID()	*R*
GetServiceResetToken()	*R*

Name	R/O [a]
Browse()	*R*
Search()	*O*
CreateObject()	*O*
DestroyObject()	*O*
UpdateObject()	*O*
MoveObject()	*O*
ImportResource()	*O*
ExportResource()	*O*
DeleteResource()	*O*
StopTransferResource()	*O*
GetTransferProgress()	*O*
CreateReference()	*O*
FreeFormQuery()	*O*
GetFreeFormQueryCapabilities()	*O*
Non-standard actions implemented by an UPnP vendor go here.	*X*
a *R* = REQUIRED, *O* = OPTIONAL, *X* = Non-standard.	
b REQUIRED if implementation supports sort modifiers other than "+" and "-"	

Note: Non-standard actions MUST be implemented in such a way that they do not interfere with the basic operation of the ContentDirectory service, that is: these actions MUST be OPTIONAL and do not need to be invoked for the ContentDirectory service to operate normally.

2.5.1 *GetSearchCapabilities()*

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

2.5.1.1 Arguments

Table 2-9 — Arguments for *GetSearchCapabilities()*

Argument	Direction	Related State Variable
SearchCaps	*OUT*	*SearchCapabilities*

2.5.1.2 Dependency on State

None.

2.5.1.3 Effect on State

None.

2.5.1.4 Errors

Table 2-10 — Error Codes for *GetSearchCapabilities()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.

2.5.2 *GetSortCapabilities()*

This action returns a CSV list of property names that can be used in the *sortCriteria* argument.

2.5.2.1 Arguments

Table 2-11 — Arguments for *GetSortCapabilities()*

Argument	Direction	Related State Variable
SortCaps	*OUT*	*SortCapabilities*

2.5.2.2 Dependency on State

None.

2.5.2.3 Effect on State

None.

2.5.2.4 Errors

Table 2-12 — Error Codes for *GetSortCapabilities()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.

2.5.3 *GetSortExtensionCapabilities()*

This action returns the CSV list of sort modifiers supported by the ContentDirectory service. This action MUST be implemented if modifiers other than "+" and "-" are supported.

2.5.3.1 Arguments

Table 2-13 — Arguments for *GetSortExtensionCapabilities()*

Argument	Direction	Related State Variable
SortExtensionCaps	*OUT*	*SortExtensionCapabilities*

2.5.3.2 Dependency on State

None.

2.5.3.3 Effect on State

None.

2.5.3.4 Errors

Table 2-14 — Error Codes for *GetSortExtensionCapabilities()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.

2.5.4 GetFeatureList()

This action returns a *Features XML Document* describing which optional *CDS Features* this device supports, if any.

2.5.4.1 Arguments

Table 2-15 — Arguments for GetFeatureList()

Argument	Direction	Related State Variable
FeatureList	OUT	FeatureList

2.5.4.2 Dependency on State

None.

2.5.4.3 Effect on State

None.

2.5.4.4 Errors

Table 2-16 — Error Codes for GetFeatureList()

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.

2.5.5 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 ContentDirectory service (as opposed to subscribing to events).

2.5.5.1 Arguments

Table 2-17 — Arguments for GetSystemUpdateID()

Argument	Direction	Related State Variable
Id	OUT	SystemUpdateID

2.5.5.2 Dependency on State

None.

2.5.5.3 Effect on State

None.

2.5.5.4 Errors

Table 2-18 — Error Codes for GetSystemUpdateID()

Error Code	Error Description	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.

2.5.6 *GetServiceResetToken()*

This action returns the current value of the *ServiceResetToken* state variable. The returned value can be compared to a previously known value to determine if the ContentDirectory service implementation can no longer maintain a consistent progression of internal state. See subclause 2.3.7, "*ServiceResetToken*" state variable for details.

2.5.6.1 Arguments

Table 2-19 — Arguments for *GetServiceResetToken()*

Argument	Direction	Related State Variable
ResetToken	*OUT*	*ServiceResetToken*

2.5.6.2 Dependency on State

None.

2.5.6.3 Effect on State

None.

2.5.6.4 Errors

Table 2-20 — Error Codes for *GetServiceResetToken()*

Error Code	Error Description	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.

2.5.7 *Browse()*

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

2.5.7.1 Arguments

The following list presents an overview of the *Browse()* action arguments.

- *ObjectID*: The *@id* of the object currently being browsed. An *ObjectID* value of zero corresponds to the root object of the ContentDirectory service.
- *BrowseFlag*: See subclause 2.3.14, "*A_ARG_TYPE_BrowseFlag*."
- *Filter*: See subclause 2.3.15, "*A_ARG_TYPE_Filter*."
- *StartingIndex*: Zero-based offset to enumerate children under the container specified by *ObjectID*. *StartingIndex* MUST be set to 0 if *BrowseFlag* is equal to "*BrowseMetadata*".
- *RequestedCount*: Requested number of entries under the object specified by *ObjectID*. *RequestedCount* = 0 indicates request all entries.
- *SortCriteria*: See subclause 2.3.16, "*A_ARG_TYPE_SortCriteria*."
- *Result*: See subclause 2.3.12, "*A_ARG_TYPE_Result*."
- *NumberReturned*: Number of objects returned in the *Result* argument. If *BrowseFlag* is set to "*BrowseMetadata*", then *NumberReturned* MUST be set to 1.
- *TotalMatches*: If *BrowseFlag* is set to "*BrowseMetadata*", then *TotalMatches* MUST be set to 1. Else if *BrowseFlag* is set to "*BrowseDirectChildren*", then *TotalMatches* MUST be set to the total number of objects in the object specified for the *Browse()* action

(independent of the starting index specified by the *StartingIndex* argument). If the ContentDirectory service implementation cannot timely compute the value of *TotalMatches*, but there are matching objects that have been found by the ContentDirectory service implementation, then the *Browse()* action MUST successfully return with the *TotalMatches* argument set to zero and the *NumberReturned* argument indicating the number of returned objects.
If the ContentDirectory service implementation cannot timely compute the value of *TotalMatches*, and there are no matching objects found, then the *Browse()* action MUST return error code 720.

- *UpdateID*: The value returned in the *UpdateID* argument MUST be the *SystemUpdateID* state variable value at the time the *Browse()* response was generated. If a control point finds that the current *SystemUpdateID* state variable value is not equal to the value returned in the *UpdateID* argument, then a change within the ContentDirectory service has occurred between the time the result was generated and the time that the control point is processing the result. The control point may therefore want to re-invoke the *Browse()* action to ensure that it has the latest property values. Note however that the change in the value of the *SystemUpdateID* state variable may have been caused by a change that occurred in a location in the ContentDirectory tree hierarchy that is not part of the returned result. In this case, the re-invocation of the *Browse()* action will return the exact same result.

  Note: This definition is not backwards compatible with previous versions of this specification. However, the previous definition did not indicate changes to properties of child containers. Therefore the control point would not have been aware that it had stale data.

Table 2-21 — Arguments for *Browse()*

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*
UpdateID	*OUT*	*A_ARG_TYPE_UpdateID*

2.5.7.2 Dependency on State

None.

2.5.7.3 Effect on State

None.

2.5.7.4 Errors

Table 2-22 — Error Codes for *Browse()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.

errorCode	errorDescription	Description
701	No such object	The *Browse()* request failed because the specified *ObjectID* argument is invalid.
709	Unsupported or invalid sort criteria	The *Browse()* request failed because the specified *SortCriteria* is not supported or is invalid.
720	Cannot process the request	The *Browse()* request failed because the ContentDirectory service is unable to compute, in the time allotted, the total number of objects that are a match for the browse criteria and is additionally unable to return, in the time allotted, any objects that match the browse criteria.

2.5.8 *Search()*

This action allows the caller to search a ContentDirectory service subtree for objects that match some search criteria. The subtree root container is specified in the *ContainerID* input argument. The search criteria are specified as a query string operating on properties with comparison and logical operators.

2.5.8.1 Arguments

The *Filter*, *StartingIndex*, *RequestedCount*, *SortCriteria* input arguments are the same as the corresponding input arguments for the *Browse()* action. The *Result* and *UpdateID* output arguments are the same as the corresponding output arguments for the *Browse()* action. (See subclause 2.5.7, "*Browse()*").In addition, the following arguments are defined:

- *ContainerID*: Unique identifier of the root container of the subtree in which to perform the search. A *ContainerID* value of zero corresponds to the root object of the ContentDirectory service.
- *NumberReturned*: Number of ContentDirectory service objects returned in the *Result* argument.
- *TotalMatches*: Total number of ContentDirectory service 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 ContentDirectory service implementation cannot timely compute the value of *TotalMatches*, but there are matching objects that have been found by the ContentDirectory service implementation, then the *Search()* action MUST successfully return with the *TotalMatches* argument set to zero and the *NumberReturned* argument indicating the number of returned objects.
  If the ContentDirectory service implementation cannot timely compute the value of *TotalMatches*, and there are no matching objects found, then the *Search()* action MUST return error code 720.
- *SearchCriteria*: See subclause 2.3.13, "*A_ARG_TYPE_SearchCriteria*."

Table 2-23 — Arguments for *Search()*

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*

Argument	Direction	Related State Variable
UpdateID	*OUT*	*A_ARG_TYPE_UpdateID*

2.5.8.2 Dependency on State

None.

2.5.8.3 Effect on State

None.

2.5.8.4 Errors

Table 2-24 — Error Codes for *Search()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
708	Unsupported or invalid search criteria	The *Search()* request failed because the *SearchCriteria* argument is not supported or is invalid
709	Unsupported or invalid sort criteria	The *Search()* request failed because the *SortCriteria* argument is not supported or is invalid
710	No such container	The *Search()* request failed because the *ContainerID* argument is invalid or identifies an object that is not a container.
720	Cannot process the request	The *Search()* request failed because the ContentDirectory service is unable to compute, in the time allotted, the total number of objects that are a match for the search criteria and is additionally unable to return, in the time allotted, any objects that match the search criteria.

2.5.9 *CreateObject()*

This action creates a new object in the container identified by *ContainerID*. The *Elements* input argument MUST conform to the DIDL-Lite schema [DIDL-LITE-XSD]. Consequently, the minimum information that MUST be included is the *@id*, *@parentID*, *@restricted*, *dc:title*, and *upnp:class* properties. Since the value of the *@id* property is assigned by the *CreateObject()* action, it MUST initially be set to "". The value of the *@parentID* property MUST match the value specified by the *ContainerID* input argument. Additionally, the *@restricted* property MUST be set to "*0*" (false). If any of these requirements are not met, the device MUST return error code 712 – "Bad metadata".

The ContentDirectory service MUST prevent control points from initializing the *@restricted* property to "*1*" (true) since restricted objects can only be deleted and/or modified by the service itself according to some service-internal rules. Allowing a control point to initialize the *@restricted* property to "*1*" would create an object that can not be deleted and/or modified by the service because the service does not know the rules for that object. If this were allowed to happen, the new object would become both permanent and un-modifiable.

The other properties of the new object are initialized according to the specified input properties. In addition, the ContentDirectory service MAY create additional properties, for example, to ensure consistency across the whole directory. The unique *@id* assigned to the newly created object is returned in the output argument *ObjectID*. The complete object description is returned in output argument *Result* in DIDL-Lite form.

2.5.9.1 *res* Property Creation

When the new object will have one or more *res* properties, the *res* properties MUST be generated in one of two ways:

a) **The control point specifies a value of the *res* property and other known associated *res@xxx* properties.** The value of the *res* property MUST identify a pre-existing resource, for example, an Internet radio station. When a *res* value is present, the resource is available immediately and there is no need to invoke the *ImportResource()* action.

The following is an example of the *res* property and its associated *res@xxx* properties as returned in the *Result* argument of the *CreateObject()* action when the control point specifies a value for both the *res* property and the *res@protocolInfo* property:

Request:

```
CreateObject("10", "
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="" parentID="10" restricted="0">
      <dc:title>New Song</dc:title>
      <upnp:class>
         object.item.audioItem
      </upnp:class>
      <res protocolInfo="http:*:audio/mp3:*">
         http://10.0.0.1/contentdir?id=10
      </res>
   </item>
</DIDL-Lite>")
```

Response:

```
CreateObject("12", "
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="12" parentID="10" restricted="0">
      <dc:title>New Song</dc:title>
      <dc:creator></dc:creator>
      <upnp:class>
         object.item.audioItem
      </upnp:class>
      <res protocolInfo="http:*:audio/mp3:*">
         http://10.0.0.1/contentdir?id=10
      </res>
   </item>
</DIDL-Lite>")
```

If the ContentDirectory service implementation allows the resource to be updated, then in addition, the *res@importUri* property will be returned. It can be used to *update* the resource at a later stage (using the *ImportResource()* action):

```
…
<res
 protocolInfo="http:*:audio/mp3:*"
 importUri="http://10.0.0.1/postdir?id=10">
   http://10.0.0.1/contentdir?id=10
</res>
…
```

b) **The control point does not specify a value for the *res* property**. In this case, the ContentDirectory service MUST create the *res@importUri* property whose value is used for importing the resource at a later time. The *res* property returned to the control point (as part of the *CreateObject()* *Result* output argument) has no value (actually set to “”). The resource is therefore not yet accessible.

To make the content accessible, one of the following must occur for each *res* property that does not have a value:

1) Some external entity (for example, the device that has an external copy of the desired content) uses the value of the associated *res@importUri* property to push (for example, via HTTP-POST) the desired content to the ContentDirectory service implementation. This creates a local copy of the external content.

2) The control point invokes the *ImportResource()* action with the *SourceURI* argument set to the external location of the desired content and the *DestinationURI* argument set to the value of the associated *res@importUri* property of the target item. The *ImportResource()* action uses HTTP-GET on the *SourceURI* to retrieve the target content and to create a local copy of it. The *DestinationURI* argument (which is set to the value of the associated *res@importUri* property) is simply used to uniquely identify the local destination location that will receive the content.

The following is an example of the *res* property and its associated *res@xxx* properties as returned in the *Result* argument of the *CreateObject()* action when the control point does not specify a value for the *res* property, but provides the *res@protocolInfo* property and a value for the *res@importUri* property that MUST used to bind the resource to the object:

Request:

```
CreateObject("10", "
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="" parentID="10" restricted="0">
      <dc:title>New Song</dc:title>
      <upnp:class>
         object.item.audioItem
      </upnp:class>
      <res protocolInfo="*:*:*:*">
      </res>
   </item>
</DIDL-Lite>")
```

Response:

```
CreateObject("12", "
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="12" parentID="10" restricted="0">
      <dc:title>New Song</dc:title>
      <dc:creator></dc:creator>
      <upnp:class>
```

```
            object.item.audioItem
          </upnp:class>
          <res
           protocolInfo="http:*:audio/mp3:*"
           importUri="http://10.0.0.1/postdir?id=10">
          </res>
      </item>
  </DIDL-Lite>")
```

Once the local copy has been created, the ContentDirectory service implementation sets the value of the *res* property to a URI that resolves to the new local copy, and the content is then accessible. This new content URI MAY be different from the value of the associated *res@importUri*property. The ContentDirectory service implementation MAY subsequently remove the associated *res@importUri* property, or keep it for the purpose of updating the content in the future.

In both cases a) and b), if the control point knows the MIME-type of the resource being added, the associated *res@protocolInfo* property should be set to "`*:*:MIME-type:*`" (for example, "`*:*:audio/m3u:*`"). Otherwise, it should be set to "`*:*:*:*`". It is the responsibility of the ContentDirectory service to fill in the appropriate values for the *protocol*, *network* and *additionalInfo* fields of the associated *res@protocolInfo* property (for example, "`http:*:audio/m3u:*`") when it knows them (typically after importing the resource). This information is used to enable compatibility checking between MediaServer and MediaRenderer devices for this resource.

The ability to allow *res* properties in container objects is vendor dependent. If a ContentDirectory service implementation does not allow container objects to have *res* properties, attempting to create a container object with a *res* property MUST generate error code 712 – "Bad metadata".

CreateObject() can not be used to create *reference items*. *Reference items* are actually references to other existing ContentDirectory service items and are generated with the *CreateReference()* action.

CreateObject() can also be used to create a new bookmark item. A bookmark item can be created in any container. When a bookmark item is created, the associated content item MUST be updated so that one of its *upnp:bookmarkID* properties contains the object ID of the newly created bookmark item. After the bookmark is created, it MUST contain the object ID of the bookmarked content item in its *upnp:bookmarkedObjectID* property. (See also clause E.3, "Requirements for the *BOOKMARK feature*, Version 1")

In the *Elements* input argument, the *upnp:bookmarkedObjectID*, *dc:title*, *upnp:deviceUDN* (AVT and RCS) *upnp:deviceUDN@serviceId*, *upnp:deviceUDN@serviceType*, and *upnp:stateVariableCollection* properties are specified to create a bookmark item. The *upnp:class* property MUST be set to "*object.item.bookmarkItem*" or a derived class if the *upnp:createClass* property in the bookmark container allows this. Other bookmark related information such as creation time *(dc:date*) is created by the ContentDirectory service or the control point. If the control point has a clock, it sets creation time to the current time. If available, the ContentDirectory service can overwrite the control point-supplied creation time with its own notion of creation time. If the ContentDirectory service does not have a clock, then it MUST NOT update or remove creation time from the object. "Table C.16 — *bookmarkItem:item* Properties" shows the structure of each bookmark item.

Observe particularly that:

c) AVTransport service implementations that want to participate in scenarios that use bookmarks MUST implement the *AVTransportURIMetaData* state variable to store the relevant *DIDL-Lite XML fragment* that includes the object ID of the current content.

d) A control point embedded with a private MediaServer or MediaRenderer MUST provide a persistent UDN that is not exposed to the network but is used in a data structure that contains a UDN field. Additionally, serviceType and serviceId must be supported by the device.

Vendors who want to enhance a bookmark application can add vendor-specific fields to bookmark items.

In addition, the *CreateObject()* action can be used to create a new bookmark container. The newly created container MUST have the bookmark container class type and SHOULD set the *@neverPlayable* property to "*1*" if it will never contain content other than (non-playable) bookmarks.

2.5.9.2 Arguments

Table 2-25 — Arguments for *CreateObject()*

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*

2.5.9.3 Dependency on State

None.

2.5.9.4 Effect on State

This action updates the *SystemUpdateID* state variable. Also, various properties of the parent container of the created object are modified, such as its *@childCount* property and *ContainerUpdateIDValue* indicator. Consequently, the *ContainerUpdateIDs* state variable, if supported, is updated as well.

2.5.9.5 Errors

Table 2-26 — Error codes for *CreateObject()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
710	No such container	*CreateObject()* failed because the *ContainerID* argument is invalid or identifies an object that is not a container.
712	Bad metadata	*CreateObject()* failed because the *Elements* argument is not supported or is invalid.
713	Restricted parent object	*CreateObject()* failed because the *@restricted* property of the container specified by *ContainerID* argument is set to "*1*".

2.5.10 *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 results of *DestroyObject()* in the case that the targeted object is a container with *@restricted* property set to "*0*" and one or more direct child or descendant child objects with

@restricted properties set to "*1*" are vendor-dependent. There are three likely outcomes when this condition prevails:

- The ContentDirectory service implementation destroys the specified container as well as all direct child and descendant objects of the specified container, regardless of whether or not they are restricted. The *DestroyObject()* action returns successfully.
- The ContentDirectory service implementation does not destroy any objects. The *DestroyObject()* action fails and returns error code 711.
- The ContentDirectory service implementation does not destroy the specified container, but does destroy all of the *non-restricted* direct child and descendant objects of the specified container that are not needed to preserve the original object structure hierarchy, and returns successfully.

Because the results of the *DestroyObject()* action are vendor dependent when the above condition prevails, control points are strongly recommended to execute *DestroyObject()* on all of the descendant and child objects in the targeted container object individually before attempting to destroy the container.

The *DestroyObject()* action MUST fail with error code 711 in the case that the targeted object has its *@restricted* property set to "*1*" .

The ContentDirectory service implementation MAY delete a resource when it detects, with absolute certainty, that there are no references to it left anywhere in the ContentDirectory service after the successful *DestroyObject()* action. For ContentDirectory service implementations that *do not* attempt to delete resources, *DestroyObject()* returns successfully. These ContentDirectory service implementations may possess some means of handling resources that are no longer referenced by the ContentDirectory service as a result of the *DestroyObject()* action. For ContentDirectory service implementations that *do* attempt to delete resources, there are three likely outcomes of the *DestroyObject()* action:

- The ContentDirectory service implementation deletes all or some portion of the resources that are no longer referenced. The *DestroyObject()* action returns successfully even if only a portion or no resources at all are deleted.
- *DestroyObject()* fails and returns error code 714 indicating that an unsuccessful attempt was made to delete one or more resources referenced in the target object because one or more resources were not found. No resources are deleted and there is no change in state of the ContentDirectory service.
- *DestroyObject()* fails and returns error code 715 indicating that an unsuccessful attempt was made to delete one or more resources referenced in the target object because one or more resources can not be accessed. No resources are deleted and there is no change in state of the ContentDirectory service.

This action can also be used to destroy a bookmark item or a bookmark container. When a bookmark item is to be destroyed, the ContentDirectory service MUST first find the associated content item using the *upnp:bookmarkedID* property of the bookmark item and it MUST remove the associated *upnp:bookmarkID* property from the content item. Similarly, when a content item that contains one or more *upnp:bookmarkID* properties is destroyed, the ContentDirectory service MUST find all associated bookmark items and MUST also delete those bookmark items.

2.5.10.1 Arguments

Table 2-27 — Arguments for *DestroyObject()*

Argument	Direction	Related State Variable
ObjectID	*IN*	*A_ARG_TYPE_ObjectID*

2.5.10.2 Dependency on State

None.

2.5.10.3 Effect on State

This action updates the *SystemUpdateID* state variable. Also, various properties of the parent container of the destroyed object are modified, such as its *@childCount* property and *ContainerUpdateIDValue* indicator. Consequently, the *ContainerUpdateIDs* state variable, if supported, is updated as well.

2.5.10.4 Errors

Table 2-28 — Error Codes for *DestroyObject()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
701	No such object	*DestroyObject()* failed because the object specified by the *ObjectID* argument is invalid.
711	Restricted object	*DestroyObject()* failed because the *@restricted* property of the object specified by the *ObjectID* argument is set to "*1*".
713	Restricted parent object	*DestroyObject()* failed because the *@restricted* property of the parent object of the object specified by the *ObjectID* argument is set to "*1*".
714	No such resource	*DestroyObject()* failed because the resource referenced by the object specified by the *ObjectID* argument cannot be identified.
715	Source resource access denied	*DestroyObject()* failed because the resource referenced by the object specified by the *ObjectID* argument cannot be accessed.

2.5.11 *UpdateObject()*

This action adds, deletes, or modifies object metadata. The object to be updated is specified by the *ObjectID* argument. The *CurrentTagValue* argument identifies the set of existing object properties (and their values) that are to be updated. Each independent property is represented by a single entry in the CSV list contained in the *CurrentTagValue* argument. The *NewTagValue* argument identifies how the object is to be updated. Both the *CurrentTagValue* and *NewTagValue* arguments are a CSV list containing the same number of entries. The property identified in each entry of the *CurrentTagValue* argument is updated based on the contents of the corresponding entry of the *NewTagValue* argument. For example, the property identified in the 5th entry of the *CurrentTagValue* argument is updated based on the contents of the 5th entry of the *NewTagValue* argument. Each entry of the *CurrentTagValue* and *NewTagValue* arguments is either empty (i.e. contains no data) or contains a *DIDL-Lite XML fragment* that represents the complete XML representation of an independent property.

Within the *CurrentTagValue* argument, each XML fragment MUST contain a complete, exact copy of the XML representation of an existing independent property of the object (including the property's full value plus any associated XML attributes). For example, the XML fragment can be copied directly from the results of a *Browse()* or *Search()* action. Each XML fragment MUST match the current representation of the property. Otherwise, the action MUST return error code 728 – "Outdated object metadata" to indicate that the contents of the *CurrentTagValue* argument is outdated. The *UpdateObject()* action MUST not be used to add, delete, or modify any read-only properties. If an attempt is made to add, delete, or modify a read-only property, the action MUST return error code 705 – "Read only tag". See Table B.1, "ContentDirectory Service Properties Overview" in Annex B, "**(normative)**

AV Working Committee Properties" for a list of properties designated as read-only. When the

CurrentTagValue argument contains multiple entries, those entries MUST be processed in order starting with the first entry.

Within the *NewTagValue* argument, each XML fragment MUST contain the complete XML fragment that is to replace the XML fragment listed in the corresponding element of the *CurrentTagValue* argument. The replacement XML fragment MUST contain the name of the independent property that is being updated, its value, and any associated XML attributes. The independent property name in a *NewTagValue* entry MUST match the independent property name of the corresponding *CurrentTagValue* entry. The *UpdateObject()* action MUST not be used to replace one property by a different property. However, this can be accomplished by first deleting the old property and then adding the new one. Both operations may be accomplished with a single invocation of the *UpdateObject()* action.

An empty entry in the *NewTagValue* argument indicates that the property identified by the corresponding entry of the *CurrentTagValue* argument MUST be deleted from the object. Similarly, an empty entry in the *CurrentTagValue* argument indicates that the property (and its value) contained within the corresponding entry of the *NewTagValue* argument MUST be added to the object. If adding, deleting, or modifying any of the specified properties would result in an invalid object, the *UpdateObject()* action MUST fail without any change to the object. Some examples include:

- Attempting to delete a REQUIRED property, unless the property appears multiple times and this single removal leaves the object with a valid set of REQUIRED occurrences.
- Attempting to change the value of the *dc:date* property to a person's name.
- Attempting to changing the object's class.

When deleting a *res* property, the ContentDirectory service MAY delete the corresponding resource when it detects, with absolute certainty, that there are no other references to that resource anywhere in the ContentDirectory service. Additionally, when one or more *res* properties are to be added, the procedure described in subclause 2.5.9.1, "*res* Property Creation" MUST be followed.

When multiple updates are specified (in other words, when the *CurrentTagValue*/*NewTagValue* arguments each have more than one entry) the request MUST be performed as an atomic operation. Specifically, all modifications to the object MUST be made before any change is visible to an external observer. The action either succeeds entirely (except for ignoring unsupported property additions, see below) or the object MUST NOT be modified and an error MUST be returned. In other words, a partial update MUST never occur. An implementation MAY silently ignore an attempt to add properties that are not supported. However, if no change to the object results, an error MUST be returned. Whenever the action is successful, the object has experienced an *Object Modification* as defined in subclause 2.2.5, "Object Modification".

2.5.11.1 Reference Items

For *reference items*, some properties are inherited from the *referenced item* identified via the *@refID* property. (See subclause 2.2.21, "*reference*, *reference item*, *referenced item*". These inherited properties belong to the *referenced item* but are also exposed as properties of the *reference item*. Due to the unique nature of inherited properties, certain *UpdateObject()* operations require special handling when applied to the inherited properties of a *reference item*.

- **Deleting an Inherited Property**: When an attempt is made to delete an inherited property from a *reference item*, the inherited property becomes hidden (within the context of the *reference item*) even though the property remains unchanged within the context of the *referenced item*. As described below, inheritance of the property can be re-established, if desired.

- **Modifying an Inherited Property**: When an attempt is made to modify an inherited property, the inherited value of the property is replaced with the new value but only within the context of the *reference item*. As with deleting an inherited property, the original value of the property within the context of the *referenced item* remains unchanged. The original value is hidden (and in this case replaced) within the context of the *reference item*. The modified property value (in the *reference item*) is distinct from the corresponding property in the *referenced item* and remains disassociated until inheritance of the property is explicitly re-established as described below.

  After an inherited property has been modified (as described above), all subsequent modifications of that property affect the local replacement value (i.e. the value stored exclusively within the context of the *reference item*) and do not affect the original inherited value stored within the context of the *referenced item*. In other words, the original inherited property value from the *referenced item* remains hidden.

  After an inherited property has been modified (as described above), a subsequent deletion of that property results in the removal of the property from the context of the *reference item*. The hidden inherited property belonging to the *referenced item* remains intact. However, it remains hidden until inheritance is re-established (see below).

- **Re-establishing Inheritance of a Property**: When dealing with a *reference item*, the concept of deleting a hidden inherited property is invalid since the property does not appear in the context of the *reference item*. Consequently, the *UpdateObject()*'s delete syntax is used to re-establish the hidden inherited property within the context of the *reference item*. In this case, the contents of the *CurrentTagValue* argument MUST include the complete XML representation of the hidden inherited property from the context of the *referenced item*. Upon successful completion of the action, the inherited property will once again appear within the context of the *reference item*. Note: To re-establish an inherited property that has been modified, a delete operation MUST first be invoked to remove the local value that exists (exclusively) within the context of the *reference item*. Then, inheritance can be re-established via a subsequent delete operation as described above.

Table 2-29 — Update examples

Operation	*CurrentTagValue*	*NewTagValue*	Notes
Change the *dc:title* property of a song	<dc:title> My Favorite Song </dc:title>	<dc:title> My Second Favorite Song </dc:title>	
Delete the *dc:date* property	<dc:date> 1990-01-01 </dc:date>	(Empty entry)	
Insert a *upnp:genre* property	(Empty entry)	<upnp:genre> Swing </upnp:genre>	
Insert a second value to the multi-value *upnp:genre* property (Option-1)	(Empty entry)	<upnp:genre> Jazz </upnp:genre>	Assuming the "Swing" genre already exists, this operation results in two genre properties with a value of "Swing" and "Jazz".
Insert a second value to the multi-value *upnp:genre* property (Option-2)	<upnp:genre> Swing </upnp:genre>	<upnp:genre> Swing </upnp:genre> <upnp:genre> Jazz </upnp:genre>	Assuming the "Swing" genre already exists, this operation results in two genre properties with a value of "Swing" and "Jazz".
Change the *upnp:artist* property from Singer1 to Singer2	<upnp:artist> Singer1 </upnp:artist>	<upnp:artist> Singer2 </upnp:artist>	The entire top-level XML element (that is: `<upnp:artist>`) is included in both the *CurrentTagValue* and *NewTagValue* arguments.

Operation	*CurrentTagValue*	*NewTagValue*	Notes
Change the *dc:title* property, insert another *upnp:genre* property, and delete the *dc: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 the *CurrentTagValue* argument, note the empty entry, indicated by the double-comma placeholder just after the `</dc:title>` XML element. In the *NewTagValue* argument, note that the trailing comma at the end represents an empty entry that is a placeholder for the deleted *dc:publisher* property.
Modifying an inherited property, for example, *upnp:artist*.	<upnp:artist> Somebody </upnp:artist>	<upnp:artist> Somebody else </upnp:artist>	Prior to this *UpdateObject()* action invocation, a *Browse()* action on this *reference item* will return the *upnp:artist* property stored in the *referenced item*. Following this *UpdateObject()* action invocation, a *Browse()* action on this *reference item* will return "Somebody else" regardless of any change to the *referenced item*.
Deleting a modified inherited property, for example, the *upnp:artist* property from above.	<upnp:artist> Somebody else </upnp:artist>	(Empty entry)	Prior to this *UpdateObject()* action invocation, a *Browse()* action on this *reference item* will return the *upnp:artist* property stored in the *reference item* i.e. "Somebody else". Following this *UpdateObject()* action invocation, a *Browse()* action on this *reference item* will not return an *upnp:artist* property because it has been deleted from the *reference item* and the inherited *upnp:artist* property from the *referenced item* remains hidden.
Re-establishing inheritance from the *referenced item*, for example, the *upnp:artist* property from above.	<upnp:artist> Somebody </upnp:artist> Note: This is the current value from the *referenced item*.	(Empty entry)	Prior to this *UpdateObject()* action invocation, a *Browse()* action on this *reference item* will not return an *upnp:artist* property because it has been deleted from the *reference item* and the inherited *upnp:artist* property from the *referenced item* is hidden. Following this *UpdateObject()* action invocation, a *Browse()* action on this *reference item* will return the *upnp:artist* property from the *referenced item* because inheritance has been re-established.

Operation	*CurrentTagValue*	*NewTagValue*	Notes
Changing the value of the *upnp:desc* property.	`<desc id:"xyz" namespace=”MyNS”> <MyNS:Tag1> value1 </MyNS:Tag1> <MyNS:Tag2> old_value </MyNS:Tag2> </desc>`	`<desc id:"xyz" namespace=”MyNS”> <MyNS:Tag1> value1 </MyNS:Tag1> <MyNS:Tag2> new_value </MyNS:Tag2> </desc>`	Even though just one element is modified, the full contents of the *upnp:desc* property must be included in both the *CurrentTagValue* and *NewTagValue* arguments.

2.5.11.2 Arguments

Table 2-30 — Arguments for *UpdateObject()*

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

2.5.11.3 Dependency on State

None.

2.5.11.4 Effect on State

This action changes the metadata of the specified object. It also updates the *SystemUpdateID* state variable. Also, various properties of the parent container of the modified object are modified, such as its *ContainerUpdateIDs* indicator. Consequently, the *ContainerUpdateIDs* state variable, if supported, is updated as well.

2.5.11.5 Errors

Table 2-31 — Error Codes for *UpdateObject()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
701	No such object	*UpdateObject()* failed because the specified *ObjectID* is invalid.
702	Invalid currentTagValue	*UpdateObject()* failed because one or more entries listed in the *CurrentTagValue* argument do not match the current state of the ContentDirectory service. The specified data is likely out of date.
703	Invalid newTagValue	*UpdateObject()* failed because one or more entries listed in the *NewTagValue* argument has an unsupported or invalid property value..
704	Required tag	*UpdateObject()* failed because the request included a request to delete a required property.
705	Read only tag	*UpdateObject()* failed because the request included a request to update a read-only property.
706	Parameter Mismatch	*UpdateObject()* failed because the number of entries (including empty entries) in the *CurrentTagValue* and *NewTagValue* arguments do not match.
711	Restricted object	*UpdateObject()* failed because the *@restricted* property of the object specified by the *ObjectID* argument is set to “*1*”.
712	Bad metadata	*UpdateObject()* failed because one or more entries listed in the *CurrentTagValue* argument has an unsupported or invalid property value.

errorCode	errorDescription	Description
713	Restricted parent object	*UpdateObject()* failed because the *@restricted* property of the parent object of the object specified by the *ObjectID* argument is set to "*1*".

2.5.12 *MoveObject()*

This optional action moves ContentDirectory objects within the ContentDirectory service hierarchy when permitted. The caller specifies the ID of the object to move in the *ObjectID* input argument and the ID of the destination container in the *NewParentID* input argument and the action returns the object ID of the moved object after the move has completed in the *NewObjectID* output argument. The *MoveObject()* action may be invoked to move either containers or items. A container move action is a hierarchical move. If a container contains other objects, all contained objects must be moved along with the parent object. The object ID of the moved object or any of its descendent children MAY be changed by the move operation but all other object IDs MUST remain unchanged by the move operation. If a moved object is referenced by other objects, all references to the moved object must remain valid after the ContentDirectory service has completed the move operation. While implementers MAY choose to provide an implementation which changes the object ID of the objects being moved, this requirement may create a significant database problem for ContentDirectory service implementations with many entries. If a *MoveObject()* implementation changes the object IDs of moving objects, it MUST also send *SystemUpdateID* events and, if it supports the *ContainerUpdateIDs* state variable, it MUST send *ContainerUpdateIDs* events indicating which containers have changed. The *ContainerUpdateIDs* state variable MUST contain the object IDs of the old parent container and the new parent container.

If the *NewObjectID* output argument is identical to the *ObjectID* input argument, a control point can conclude that no object IDs changed during the execution of the *MoveObject()* action. That is, it is illegal, during a container move, to change the object ID of any contained object without also changing the object ID of the container that the action specified in the *MoveObject()* action.

The entire requested move MUST complete or it MUST fail and leave the ContentDirectory service hierarchy unchanged.

Browse() and *Search()* actions depend upon the presence of a coherent ContentDirectory service hierarchy. If a *Browse()* or *Search()* action is invoked by a control point while the *MoveObject()* action is executing, the ContentDirectory service implementation is responsible for coordinating ContentDirectory service operations so that control points receive coherent results.

- If the object to be moved is restricted (indicated by its *@restricted* property set to true), the action must fail with error code 711 (Restricted object).
- If the destination container is restricted (indicated by its *@restricted* property set to true), the action must fail with error code 713 (Restricted destimation parent object).
- If the parent of the object to be moved is restricted (indicated by its *@restricted* property set to true), the action must fail with error code 721 (Restricted source parent object).
- The class of the object to be moved must be compatible with the *upnp:createClass* property of the destination container. If the class is not compatible, the action must fail with error code 722.
- If the move operation would create an illegal configuration for the ContentDirectory service hierarchy, the action must fail with error code 723 (Illegal move destination). This may happen, for example, if the requested destination container is a child of the container to be moved.

2.5.12.1 Arguments

Table 2-32 — Arguments for *MoveObject()*

Argument	Direction	Related State Variable
ObjectID	*IN*	*A_ARG_TYPE_ObjectID*
NewParentID	*IN*	*A_ARG_TYPE_ObjectID*
NewObjectID	*OUT*	*A_ARG_TYPE_ObjectID*

2.5.12.2 Dependency on State

None.

2.5.12.3 Effect on State

This action updates the *SystemUpdateID* state variable. Also, various properties of both the source and destination parent containers of the moved object are modified, such as their *@childCount* properties and *ContainerUpdateIDValue* indicators. Consequently, the *ContainerUpdateIDs* state variable, if supported, is updated as well.

2.5.12.4 Errors

Table 2-33 — Error Codes for *MoveObject()*

Error Code	Error Description	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
701	No such object	The object identified by *ObjectID* does not exist.
710	No such container	The container identified by *NewParentID* does not exist.
711	Restricted object	Cannot move the object because the object's *@restricted* property is set to "*1*".
713	Restricted parent object	*MoveObject()* failed because the *@restricted* property of the destination parent container is set to "*1*".
721	Restricted source parent object	*MoveObject()* failed because the *@restricted* property of the source parent container of the object to move is set to "*1*".
722	Incompatible parent class	*MoveObject()* failed because the class of the object to move is not compatible with the *upnp:createClass* property of the destination parent container.
723	Illegal destination	*MoveObject()* failed because the specified move would create an illegal configuration.

2.5.13 *ImportResource()*

This action transfers a file from an external source, specified by the *SourceURI* argument, to a local destination in the ContentDirectory service, specified by the *DestinationURI* argument. The control point invokes the *ImportResource()* action with the *SourceURI* argument set to the URI of the external location and the *DestinationURI* argument set to the value of the *res@importURI* property associated with the destination object's *res* property.The *ImportResource()* action MUST use HTTP-GET on the *SourceURI* to retrieve the external content and to create a local copy of it.

The *DestinationURI* should correspond to an existing *res@importUri* property in the ContentDirectory service implementation. The *res@importUri* property identifies a *download portal* for the associated *res* property of a specific target object. It is used to create a local copy of the external content. After the transfer finishes successfully, the local content is then associated with the target object by setting the target object's *res* property value to a URI for that content, which MAY or MAY NOT be the same URI as the one specified in the

res@importUri property, depending on the ContentDirectory service implementation. If the *res* property of the target object already has a value when the *ImportResource()* action is invoked, the resource is updated and the value of the *res* property MAY be changed.

When the ContentDirectory service validates the destination location in the ContentDirectory service implementation, the action returns a unique *TransferID* in the response and starts transferring the content. A control point can monitor the progress of the transfer by invoking the *GetTransferProgress()* action.

2.5.13.1 Arguments

Table 2-34 — Arguments for *ImportResource()*

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

2.5.13.2 Dependency on State

None.

2.5.13.3 Effect on State

This action updates the *SystemUpdateID* state variable. Also, various properties of the object are modified, such as its *upnp:objectUpdateID* and *res@updateCount* properties. When the file transfer is started, the *TransferID* value returned by the *ImportResource()* action is added into the *TransferIDs* state variable. When the file transfer is finished, the *TransferID* value is removed from the *TransferIDs* state variable.

2.5.13.4 Errors

Table 2-35 — Error Codes for *ImportResource()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
714	No such source resource	*ImportResource()* failed because the source resource specified by the *SourceURI* argument cannot be identified.
715	Source resource access denied	*ImportResource()* failed because the source resource specified by the *SourceURI* argument cannot be accessed.
716	Transfer busy	*ImportResource()* failed because the source resource specified by the *SourceURI* argument refuses to perform another file transfer.
718	No such destination resource	*ImportResource()* failed because the destination resource specified by the *DestinationURI* argument cannot be identified.
719	Destination resource access denied	*ImportResource()* failed because the destination resource specified by the *DestinationURI* argument cannot be accessed.

2.5.14 *ExportResource()*

This action transfers a file, using HTTP POST, from a local source, specified by the *SourceURI* input argument, to an external destination, specified by the *DestinationURI* input argument. When the ContentDirectory service validates the source location, the action returns a unique *TransferID* in the response and starts the HTTP POST. A control point can monitor the progress of the file transfer by using the *GetTransferProgress()* action. Note that the transfer does not remove the resource from the ContentDirectory service. The transfer simply copies the existing resource to an external destination.

2.5.14.1 Arguments

Table 2-36 — Arguments for *ExportResource()*

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

2.5.14.2 Dependency on State

None.

2.5.14.3 Effect on State

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

2.5.14.4 Errors

Table 2-37 — Error Codes for *ExportResource()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
714	No such source resource	*ExportResource()* failed because the source resource specified by the *SourceURI* argument cannot be identified.
715	Source resource access denied	*ExportResource()* failed because the source resource specified by the *SourceURI* argument cannot be accessed.
716	Transfer busy	*ExportResource()* failed because the source resource specified by the *SourceURI* argument refuses to perform another file transfer.
718	No such destination resource	*ExportResource()* failed because the destination resource specified by the *DestinationURI* argument cannot be identified.
719	Destination resource access denied	*ExportResource()* failed because the destination resource specified by the *DestinationURI* argument cannot be accessed.

2.5.15 *DeleteResource()*

This action uses the specified *ResourceURI* to locate all of the *res* properties whose value equals the value specified in the *ResourceURI* input argument in the ContentDirectory service, and then delete those *res* properties and all of their associated *res@xxx* properties from the respective objects. As a result, all located objects will end up with one less *res* property and in some cases some objects may end up without any *res* properties.

Whether or not the resource identified by *ResourceURI* is actually deleted is implementation dependent. For ContentDirectory service implementations that *do* attempt to delete resources identified by *ResourceURI*, there are three likely results of the *DeleteResource()* action:

- The *DeleteResource()* action returns successfully, indicating that the resource identified by *ResourceURI* was found and deleted.
- The *DeleteResource()* action fails and returns error code 714 indicating that an unsuccessful attempt was made to delete the resource identified by *ResourceURI* because the resource was not found. No resources are deleted and there is no change in state of the ContentDirectory service.

- The *DeleteResource()* action fails and returns error code 715 indicating that an unsuccessful attempt was made to delete the resource identified by *ResourceURI* because the resource could not be accessed. No resources are deleted and there is no change in state of the ContentDirectory service.

2.5.15.1 Arguments

Table 2-38 — Arguments for *DeleteResource()*

Argument	Direction	Related State Variable
ResourceURI	*IN*	*A_ARG_TYPE_URI*

2.5.15.2 Dependency on State

None.

2.5.15.3 Effect on State

This action changes the metadata of the affected objects. It also updates the *SystemUpdateID* state variable. Also, various properties of the parent containers of the affected objects are modified, such as their *ContainerUpdateIDValue* indicators. Consequently, the *ContainerUpdateIDs* state variable, if supported, is updated as well.

2.5.15.4 Errors

Table 2-39 — Error Codes for *DeleteResource()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
714	No such resource	*DeleteResource()* failed because the resource specified by *ResourceURI* argument was not found.
715	Source resource access denied	*DeleteResource()* failed because the resource specified by *ResourceURI* argument cannot be accessed.

2.5.16 *StopTransferResource()*

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

2.5.16.1 Arguments

Table 2-40 — Arguments for *StopTransferResource()*

Argument	Direction	Related State Variable
TransferID	*IN*	*A_ARG_TYPE_TransferID*

2.5.16.2 Dependency on State

None.

2.5.16.3 Effect on State

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

2.5.16.4 Errors

Table 2-41 — Error Codes for *StopTransferResource()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
717	No such file transfer	*StopTransferResource()* failed because the file transfer task specified by the *TransferID* argument does not exist.

2.5.17 *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. The *TransferStatus* argument indicates the status of the file transfer. It can be either "*IN_PROGRESS*", "*STOPPED*", "*ERROR*", or "*COMPLETED*". The *TransferLength* argument specifies the length in bytes that has been transferred so far. The *TransferTotal* argument specifies the total length of the file in bytes that is expected to be transferred. If the ContentDirectory serivce cannot determine the total length, the *TransferTotal* argument MUST be 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. The ContentDirectory service MUST maintain the status of a file transfer for at least 30 seconds after the file transfer has finished allowing a control point to query the result of the file transfer.

2.5.17.1 Arguments

Table 2-42 — Arguments for *GetTransferProgress()*

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*

2.5.17.2 Dependency on State

None.

2.5.17.3 Effect on State

None.

2.5.17.4 Errors

Table 2-43 — Error Codes for *GetTransferProgress()*

Error Code	Error Description	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
717	No such file transfer	*GetTransferProgress()* failed because the file transfer task specified by the *TransferID* argument does not exist.

2.5.18 *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 ContentDirectory service. A unique, new object ID is assigned to the newly created *reference item* (in its *@id* property) and returned in the *NewID* output argument.

Refer to subclause 2.2, “Key Concepts” for detailed information about *reference items*.

2.5.18.1 Arguments

Table 2-44 — Arguments for *CreateReference()*

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

2.5.18.2 Dependency on State

None.

2.5.18.3 Effect on State

This action updates the *SystemUpdateID* state variable. Also, various properties of the parent container of the created *reference item* are modified, such as its *@childCount* property and *ContainerUpdateIDValue* indicator. Consequently, the *ContainerUpdateIDs* state variable, if supported, is updated as well.

2.5.18.4 Errors

Table 2-45 — Error Codes for *CreateReference()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
701	No such object	*CreateReference()* failed because the specified *ObjectID* argument is invalid.
710	No such container	*CreateReference()* failed because the *ContainerID* argument is invalid or identifies an object that is not a container.
713	Restricted parent object	*CreateReference()* failed because the *@restricted* property of the parent object of the object specified by the *ObjectID* argument is set to “*1*”.

2.5.19 *FreeFormQuery()*

This action provides a powerful interface to search and process objects exposed by the ContentDirectory service.

A control point invoking this action creates an XQuery request as specified by the W3C XQuery 1.0 language recommendation [XQUERY10]. The XQuery language provides a rich set of tools and operators to locate and process data in XML documents. In addition, the submitted query controls the formatting of the output results so that the control point may create unique output that is convenient for its specific requirements.

The invoking control point begins the process by constructing an XQuery request and selecting a starting container as indicated by the *ContainerID* argument.

Since the XQuery language is intended to process XML formatted documents, the ContentDirectory service implementation MUST construct input to its XQuery processor that effectively complies with XML format. This input formatting process is specified by the *CDSView* argument. Currently the only supported formatting defined is the *DIDL-Lite View* (See subclause 2.2.19.1, "*DIDL-Lite View*"). The ContentDirectory service implementation MUST set the "context node" for the XQuery processor to the root node of the *CDSView*.

Since an XQuery request submitted by a control point specifies the formatting of the *FreeFormQuery()* action output, the results of the *FreeFormQuery*() action are not constrained to be DIDL-Lite or XML compliant. For example, a control point may construct an output result in the form of a CSV list.

It is RECOMMENDED that control points construct XQuery requests that limit the maximum number of data items that may be returned by the *FreeFormQuery*() action. The XQuery language provides robust facilities to implement these types of constraints. (See example in subclause 2.6.13.3, "Retrieving a limited number of photo items" and see also [XQUERY10] for more details).

The search restrictions that constrain the *Search()* action do not apply to the *FreeFormQuery()* action. Any properties defined in the ContentDirectory service that restrict the behavior of the *Search()* action, such as the *searchable* property, are ignored and do not restrict the behavior of the *FreeFormQuery()* action. Instead, the search restrictions that constrain the *FreeFormQuery()* action can be retrieved by invoking the *GetFreeFormQueryCapabilities()* action, which returns an *FFQCapabilities XML Document* that lists the properties and their namespaces that can be used in the XQuery request.

If a ContentDirectory service implementation supports the *FreeFormQuery()* action, then the *GetFreeFormQueryCapabilities()* action MUST also be supported.

2.5.19.1 Arguments

The following arguments are defined:

- *ContainerID*: Unique identifier of the container in which to start the query. A *ContainerID* value of zero corresponds to the root object of the ContentDirectory service. This argument is used to constrain the scope of the XQuery request to a ContentDirectory subtree.
- *CDSView*: specifies the type of *CDS View* to process (See subclause 2.2.19, "*CDS View*" and subclause 2.3.26, "*A_ARG_TYPE_CDSView*").
- *QueryRequest*: specifies an XQuery 1.0 request that is to be applied to the selected *CDS View*. The XQuery request contains instructions that will be applied to the input document (*CDS View*) in order to generate the result that will be returned in the *QueryResult* output argument. See subclause 2.3.26, "*A_ARG_TYPE_QueryRequest*".
- *QueryResult*: contains the result generated by processing the instructions, specified in the *QueryRequest* argument (see subclause 2.3.28, "*A_ARG_TYPE_QueryResult*"). Note that the structure of the result solely depends on the instructions provided in the *QueryRequest* argument. For example, the result could be a simple list of item titles (see example in subclause 2.6.13.1, "Retrieving the title of all music albums") or it could be a valid *DIDL-Lite XML Document* (see example in subclause 2.6.13.2, "Retrieving the audio items of Album 1").
- *UpdateID*: The *UpdateID* output argument is the same as the *UpdateID* output argument as specified in the *Browse()* action (See subclause 2.5.7, "*Browse()*").

Table 2-46 — Arguments for *FreeFormQuery()*

Argument	Direction	Related State Variable
ContainerID	*IN*	*A_ARG_TYPE_ObjectID*

Argument	Direction	Related State Variable
CDSView	*IN*	*A_ARG_TYPE_CDSView*
QueryRequest	*IN*	*A_ARG_TYPE_QueryRequest*
QueryResult	*OUT*	*A_ARG_TYPE_QueryResult*
UpdateID	*OUT*	*A_ARG_TYPE_UpdateID*

2.5.19.2 Dependency on State

None.

2.5.19.3 Effect on State

None.

2.5.19.4 Errors

Table 2-47 — Error Codes for *FreeFormQuery()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
708	Unsupported or invalid search criteria	The action failed because a specified search criteria is not supported or is invalid. This is likely caused by a reference to an unsupported property.
710	No such container	The *FreeFormQuery()* request failed because the *ContainerID* argument is invalid or identifies an object that is not a container.
720	Cannot process the request	The *FreeFormQuery()* request failed because the ContentDirectory service is unable to generate the query result in the time allotted.
724	Unsupported or invalid CDS View	The *FreeFormQuery()* request failed because the value specified in the *CDSView* argument is not supported or is invalid.
725	Invalid Query Request	The *FreeFormQuery()* request failed because the *XQuery* XML document specified in the *QueryRequest* argument is invalid. This is likely caused by an invalid XML document that does not conform to the XQuery specificiation [XQUERY10]
726	Unsupported Query Request instruction(s)	The *FreeFormQuery()* request failed because the *XQuery XML document* specified in the *QueryRequest* argument contains unsupported instructions for this particular implementation.

2.5.20 *GetFreeFormQueryCapabilities()*

This action provides a list of property names and their associated namespaces that can be used in an XQuery request on this ContentDirectory service implementation.

If a ContentDirectory service implementation supports the *GetFreeFormQueryCapabilities()* action, then the *FreeFormQuery()* action MUST also be supported.

2.5.20.1 Arguments

The following arguments are defined:

- *FFQCapabilities*: This output argument contains a *FFQCapabilities XML Document* that contains a list of property names and a list of their associated namespaces and namespace prefixes. See subclause 2.3.29, "*A_ARG_TYPE_FFQCapabilities*" and [AVS-XSD] for details.

Table 2-48 — Arguments for *GetFreeFormQueryCapabilities()*

Argument	Direction	Related State Variable

Argument	Direction	Related State Variable
FFQCapabilities	*OUT*	*A_ARG_TYPE_FFQCapabilities*

2.5.20.2 Dependency on State

None.

2.5.20.3 Effect on State

None.

2.5.20.4 Errors

Table 2-49 — Error Codes for *GetFreeFormQueryCapabilities()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.

2.5.21 Non-Standard Actions Implemented by a UPnP Vendor

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

2.5.22 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 MUST be returned.

Table 2-50 — Common Error Codes

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture subclause on Control.
500-599	TBD	See UPnP Device Architecture subclause on Control.
600-699	TBD	See UPnP Device Architecture subclause on Control.
701	No such object	The action failed because a specified object is invalid.
702	Invalid CurrentTagValue	The action failed because a specified tag/value pair does not match the current state of the ContentDirectory service.
703	Invalid NewTagValue	The action failed because the specified tag value is invalid.
704	Required tag	The action failed because the request included an implicit request to delete a required tag.
705	Read only tag	The action failed because the request included an implicit request to modify a read-only tag.
706	Parameter Mismatch	The action failed because two separate references to the number of tag/value pairs (including empty placeholders) do not match.
707	<Reserved>	Reserved for future use.
708	Unsupported or invalid search criteria	The action failed because a specified search criteria is not supported or is invalid.
709	Unsupported or invalid sort criteria	The action failed because a specified sort criteria argument is not supported or is invalid.
710	No such container	The action failed because an argument specifying a container is invalid or identifies an object that is not a container.

errorCode	errorDescription	Description
711	Restricted object	The action failed because it would result in the modification of a restricted object.
712	Bad metadata	The action failed because a specified XML tag is not supported or because a specified *DIDL-Lite XML Document* or *Fragment* is invalid.
713	Restricted parent object	The action failed because it would result in the modification of the restricted parent object of the target object.
714	No such source resource	The action failed because a specified source resource was not found.
715	Source resource access denied	The action failed because a specified source resource is busy.
716	Transfer busy	The action failed because a specified resource refuses to perform another file transfer.
717	No such file transfer	The action failed because a specified file transfer task does not exist.
718	No such destination resource	The action failed because a specified destination resource cannot be identified.
719	Destination resource access denied	The action failed because a specified destination resource is busy.
720	Cannot process the request	The action failed because the ContentDirectory service was unable to complete the necessary computations in the time allotted.
721	Restricted source parent object	The action failed because the *@restricted* property of the source parent container of the object to move is set to *true*.
722	Incompatible parent class	The action failed because the class of the object to move is not compatible with the *upnp:createClass* property of the destination parent container.
723	Illegal destination	The action failed because it would create an illegal configuration.
724	Unsupported or invalid CDS View	The request failed because the value specified in the *CDSView* argument is not supported or is invalid.
725	Invalid Query Request	The request failed because the *XQuery* XML document specified in the *QueryRequest* argument is invalid.
726	Unsupported Query Request instruction(s)	The request failed because the *XQuery XML document* specified in the *QueryRequest* argument contains unsupported instructions for this particular implementation.

Note 1: The errorDescription field returned by an action does not necessarily contain human-readable text (for example, as indicated in the second column of the Error Code tables.) It may contain machine-readable information that provides more detailed information about the error. It is therefore not advisable for a control point to blindly display the errorDescription field contents to the user.

Note 2: 800-899 Error Codes are not permitted for standard actions. See UPnP Device Architecture subclause on Control for more details.

2.6 Theory of Operation (Informative)

2.6.1 Introduction

This subclause (2.6) walks through several scenarios to illustrate the various actions supported by the ContentDirectory service. These include browsing, searching, object creation, update, and deletion, property creation, update and deletion, content transfer, playlist manipulation, Internet content representation, and bookmark manipulation.

2.6.2 Generating Object ID Values

As discussed in subclause 2.2.3, “Object Identity” and subclause 2.2.4, “Object Lifetime”, control points can benefit when objects preserve their identify (i.e. retain the value of their *@id* property even when going *off-line*)

In order to preserve an object’s identity, the value of the object’s *@id* property cannot change since that is the value used by a control point to identify an object. Additionally, when an object is deleted, the value of that object’s *@id* property cannot be assigned to another, object. Otherwise, control points might mistakenly conclude that the second object is the object that was deleted which, of course, it is not. However, if an object is deleted then subsequently restored, then the original value of that object’s *@id* property can again be assigned to the restored object – thus preserving the object’s original identity. Consequently, control points that detect an *@id* property value that have been seen before can safely conclude that this object is the same object as before but perhaps with some updated property values. See subclause 2.2.3, “Object Identity” and subclause 2.2.4, “Object Lifetime”for more details.

If for any reason, an implementation needs to change the value of an object’s *@id* property for an object for which it is tracking changes (i.e. an object with the *upnp:objectUpdateID* or *upnp:containerUpdateID* properties), it has the option of treating the action as a change to the *@id* property value and calling the necessary *Service Reset Procedure* as defined in subclause 2.3.7, “*ServiceResetToken*” and subclause 2.3.7.1, “*Service Reset Procedure*”. Alternatively, the implementation can treat the change as a separate object deletion followed by a new and different object creation. As with all object creations and/or deletions, the implementation will need to comply with all functional requirements that are mandated by the ContentDirectory specification, for example, generating events (when *on-line*), updating various state variables and/or properties including the parent container’s *@childCount* and *upnp:totalDeletedChildCount* properties, if present. See subclause 2.3.8, “*LastChange*” state variable for more details.

Although many devices do not explicitly store the value of each object’s *@id* property, preserving the identity of each object is still possible for most devices. For example, a file system-based implementation could consistently generate the same *@id* property value of each object by using the full file system pathname of the content that the object represents. Unless the content file is moved, its pathname is both unique and persistent, which, in turn, yields an object identity that is also unique and persistent. Additionally, in order to generate a unique object identity even when the filename and/or the file system’s unique identifier is reused (i.e. assigned to a different content file), the implementation may be able to generate both persistent and fully unique object IDs by appending the *@id* property value with the file’s *time of creation* which is stored by most file systems. For non file system-based implementations, it is possible to generate unique IDs by maintaining and persisting a counter of the items that have been exposed by the ContentDirectory service. When a new object is created, the value of the counter is assigned to the *@id* property of the new object and the counter is incremented. Since the *@id* property is a string, the counter can be stored as a set of characters representing the decimal or hexadecimal digits of the counter and the increment is performed by arithmetic on those character digits. In this way, a system that uses a string of length n characters can represent 16^n objects over the lifetime of the ContentDirectory service before the *@id* properties would be reused. In this environment, persistence could be ensured by storing the *@id* value along with the metadata that is stored for the object. These are just a few examples that illustrate that creating long-lived and non-reused *@id* property values is possible even thought the device does not have a lot of permanent storage dedicated to the Content Directory service implementation.

2.6.3 Content Setup for Browsing and Searching

The following illustrates the logical structure of a ContentDirectory 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"

2.6.4 Browsing

The *Browse()* action is designed to allow the control point to navigate the *native* content hierarchy exposed by the ContentDirectory service. 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 whose class is derived from the container class can be retrieved.
- **Incremental navigation** that is: the full hierarchy is never returned in one action 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 subclause 2.3.15, "*A_ARG_TYPE_Filter*.") Note that certain properties MUST NOT be filtered out in order to maintain validity of the resulting *DIDL-Lite XML Document*. If a non-filterable property is left out of the *filter* list, it will still be included in the *Result* argument.

The following examples illustrate the typical *Browse()* request-response interaction between a control point and a ContentDirectory service. It assumes the content setup specified in subclause 2.6.3, "Content Setup for Browsing and Searching".

2.6.4.1 Retrieving Sort Capabilities

When it connects to the ContentDirectory service, 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")
```

2.6.4.2 Browsing the Root Level Metadata

The control point needs to retrieve the root level metadata for the ContentDirectory service. It does this via the following *Browse()* action:

```
Request:
Browse("0", "BrowseMetadata", "*", 0, 0, "")

Response:
Browse("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <container id="0" parentID="-1" childCount="3"
    restricted="1" searchable="1">
      <dc:title>Content</dc:title>
      <upnp:class>object.container.storageFolder</upnp:class>
      <upnp:storageUsed>847000</upnp:storageUsed>
      <upnp:writeStatus>WRITABLE</upnp:writeStatus>
      <upnp:searchClass includeDerived="0">
         object.container.album.musicAlbum
      </upnp:searchClass>
      <upnp:searchClass includeDerived="0">
         object.container.album.photoAlbum
      </upnp:searchClass>
      <upnp:searchClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:searchClass>
      <upnp:searchClass includeDerived="0">
         object.item.imageItem.photo
      </upnp:searchClass>
      <upnp:searchClass name="Vendor Album Art"
       includeDerived="1">
         object.item.imageItem.photo.vendorAlbumArt
      </upnp:searchClass>
   </container>
</DIDL-Lite>", 1, 1, 10)
```

Note that the response contains the *DIDL-Lite XML Document* with the metadata corresponding to the root container of the ContentDirectory service (container *@id* = 0), and the other output arguments *NumberReturned*, *TotalMatches*, and *UpdateID*, respectively.

2.6.4.3 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* argument. It does this via the following *Browse()* action:

```
Request:
Browse("0", "BrowseDirectChildren", "*", 0, 3, "")

Response:
Browse("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <container id="1" parentID="0" childCount="2" restricted="0">
      <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="0">
         object.container.album.musicAlbum
      </upnp:searchClass>
      <upnp:searchClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         object.container.album.musicAlbum
      </upnp:createClass>
   </container>
   <container id="2" parentID="0" childCount="2" restricted="0">
      <dc:title>My Photos</dc:title>
      <upnp:class>object.container.storageFolder</upnp:class>
      <upnp:storageUsed>77000</upnp:storageUsed>
      <upnp:writeStatus>WRITABLE</upnp:writeStatus>
      <upnp:searchClass includeDerived="0">
         object.container.album.photoAlbum
      </upnp:searchClass>
      <upnp:searchClass includeDerived="0">
         object.item.imageItem.photo
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         object.container.album.photoAlbum
      </upnp:createClass>
   </container>
   <container id="30" parentID="0" childCount="2" restricted="0">
      <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="1">
         object.item.imageItem.photo.vendorAlbumArt
      </upnp:searchClass>
      <upnp:createClass includeDerived="1">
         object.item.imageItem.photo.vendorAlbumArt
      </upnp:createClass>
   </container>
</DIDL-Lite>", 3, 3, 10)
```

2.6.4.4 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* argument. In addition, it specifies the *Result* argument 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("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <container id="4" parentID="1" childCount="3" restricted="0">
      <dc:title>Brand New Day</dc:title>
      <dc:creator>Sting</dc:creator>
      <upnp:class>object.container.album.musicAlbum</upnp:class>
      <upnp:searchClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:createClass>
   </container>
   <container id="3" parentID="1" childCount="4" restricted="0">
      <dc:title>Singles Soundtrack</dc:title>
      <dc:creator>Various Artists</dc:creator>
      <upnp:class>object.container.album.musicAlbum</upnp:class>
      <upnp:searchClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:createClass>
   </container>
</DIDL-Lite>", 2, 2, 21)
```

2.6.4.5 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* argument. In addition, it specifies the *Result* argument to be sorted in ascending order by the *dc:title* property. It does this via the following *Browse()* action:

```
Request:
Browse("3", "BrowseDirectChildren", "*", 0, 3, "+dc:title")

Response:
Browse("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
```

```
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="6" parentID="3" restricted="0">
      <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="0">
      <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="0">
      <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("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="5" parentID="3" restricted="0">
      <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)
```

2.6.4.6 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* argument. It does this via the following *Browse()* action:

```
Request:
Browse("30", "BrowseDirectChildren", "*", 0, 3, "")

Response:
Browse("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
```

```
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="31" parentID="30" restricted="0">
      <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="0">
      <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)
```

2.6.5 Searching

The *Search()* action is designed to allow a control point to search for objects in the ContentDirectory service that match a given search criteria (see subclause 2.3.13, "*A_ARG_TYPE_SearchCriteria*."). In addition, the *Search()* action supports the following features:

- **Incremental result retrieval** that is: in the context of a particular request the control point can restrict the number (and the starting offset) of objects returned in the *Result* argument.
- **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 subclause 2.3.15, "*A_ARG_TYPE_Filter*.") Note that certain properties MUST NOT be filtered out in order to maintain the validity of the resulting *DIDL-Lite XML Document*. If a non-filterable property is left out of the filter set, it will still be included in the *Result* argument.

The following examples illustrate the typical *Search()* request-response interaction between a control point and a ContentDirectory service. It assumes the content setup specified in subclause 2.6.3, "Content Setup for Browsing and Searching".

2.6.5.1 Retrieving Search Capabilities

When it connects to the ContentDirectory service, the control point determines which properties can be used in the *SearchCriteria* argument of the *Search()* action. It does this via the *GetSearchCapabilities()* action:

```
Request:
GetSearchCapabilities()

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

2.6.5.2 Search for All Content Created by the performer Sting

Search for all objects where *dc:creator* is *Sting* and sort the *Result* argument 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("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="9" parentID="4" restricted="0">
      <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="0">
      <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="0"
    searchable="1">
      <dc:title>Brand New Day</dc:title>
      <dc:creator>Sting</dc:creator>
      <upnp:class>object.container.album.musicAlbum</upnp:class>
      <upnp:searchClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:createClass>
   </container>
</DIDL-Lite>", 3, 4, 10)

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

Response:
Search("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="10" parentID="4" restricted="0">
```

```
      <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)
```

2.6.5.3 Search for all Photos Taken During the Month of October

Search for all photo objects whose *dc:date* is in October and sort the *Result* argument 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 derivedfrom "object.item.imageItem.photo" and (dc:date >= "2001-10-01"
and dc:date <= "2001-10-31")", "*", 0, 3, "+dc:date")

Response:
Search("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="14" parentID="12" restricted="0">
      <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="0">
      <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)
```

2.6.5.4 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* argument 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("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
```

```
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <container id="13" parentID="2" restricted="0"
    searchable="1">
      <dc:title>Christmas</dc:title>
      <upnp:class>object.container.album.photoAlbum</upnp:class>
      <upnp:searchClass includeDerived="0">
         object.item.imageItem.photo
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         object.item.imageItem.photo
      </upnp:createClass>
   </container>
   <item id="17" parentID="13" restricted="0">
      <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)
```

2.6.5.5 Search for all *album* objects in the ContentDirectory service

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("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <container id="3" parentID="1" childCount="4" restricted="0"
    searchable="1">
      <dc:title>Singles Soundtrack</dc:title>
      <dc:creator>Various Artists</dc:creator>
      <upnp:class>object.container.album.musicAlbum</upnp:class>
      <upnp:searchClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:createClass>
   </container>
   <container id="4" parentID="1" childCount="3" restricted="0"
    searchable="1">
      <dc:title>Brand New Day</dc:title>
      <dc:creator>Sting</dc:creator>
      <upnp:class>object.container.album.musicAlbum</upnp:class>
      <upnp:searchClass includeDerived="0">
         object.item.audioItem.musicTrack
```

```
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:createClass>
   </container>
   <container id="12" parentID="2" restricted="0"
    searchable="1">
      <dc:title>Mexico Trip</dc:title>
      <upnp:class>object.container.album.photoAlbum</upnp:class>
      <upnp:searchClass includeDerived="0" >
         object.item.imageItem.photo
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         object.item.imageItem.photo
      </upnp:createClass>
   </container>
   <container id="13" parentID="2" restricted="0"
    searchable="1">
      <dc:title>Christmas</dc:title>
      <upnp:class>object.container.album.photoAlbum</upnp:class>
      <upnp:searchClass includeDerived="0" >
         object.item.imageItem.photo
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         object.item.imageItem.photo
      </upnp:createClass>
   </container>
</DIDL-Lite>", 4, 4, 10)
```

2.6.6 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.

2.6.6.1 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")
```

2.6.6.2 Search for All Photos Taken During the Month of October

Search for all photo objects whose *dc:date* is in October and sort the *Result* argument 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 derivedfrom "object.item.imageItem.photo" and (dc:date >= "2001-10-01"
and dc:date <= "2001-10-31")", "*", 0, 3, "+dc:date")

Response:
Search("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
```

```
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="14" parentID="12" restricted="0">
      <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="0">
      <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="0">
      <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)
```

2.6.6.3 Deletion of the Reference to the Photo in the Mexico Trip Album

A *reference item* is deleted via the *DestroyObject()* action:

```
Request:
DestroyObject("20")

Response:
DestroyObject()
```

2.6.7 Object Creation

2.6.7.1 Creating a New Object

The *CreateObject()* action is used to create a new object in the specified container. The ContentDirectory service will create an object according to the specified metadata. Additional metadata MAY be added by the ContentDirectory service. The action returns *ObjectID* and metadata of the created object. Note that all REQUIRED elements MUST exist in the returned *Result* argument.

2.6.7.2 Example: Creating a New MusicTrack in the Album1 (*@id* = 10)

Invoke *CreateObject()* with the *ContainerID* argument set to 10 and the *Elements* argument set to the metadata describing the new object to be created. This must include the *upnp:class* property, and in this example, its value is set to "*object.item.audioItem.musicTrack*".

```
Request:
CreateObject("10", "
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
```

```
   <item id="" parentID="10" restricted="0">
      <dc:title>New Track</dc:title>
      <upnp:class>
         object.item.audioItem.musicTrack
      </upnp:class>
   </item>
</DIDL-Lite>")

Response:
CreateObject("12", "
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="12" parentID="10" restricted="0">
      <dc:title>New Track</dc:title>
      <dc:creator></dc:creator>
      <res importUri="http://10.0.0.1/pc/item?id=12"
       protocolInfo="*:*:audio:*">
      </res>
      <upnp:class>
         object.item.audioItem.musicTrack
      </upnp:class>
      <upnp:genre></upnp:genre>
      <upnp:album>Album1</upnp:album>
   </item>
</DIDL-Lite>")
```

2.6.8 Object Resource Binding (Importing a Resource)

There are two ContentDirectory service mechanisms defined to import content into the ContentDirectory service:

- The *ImportResource()* action, which uses HTTP GET and the *res@ImportUri* property.
- HTTP POST, executed by the control point.

2.6.8.1 Transfer Using the *ImportResource()* Action

The destination (for example `http://10.0.0.1/cd/import?id=3`) is located in the ContentDirectory service and the source that needs to be imported (for example `http://server/song.mp3`) is external to the ContentDirectory service. (Any resource identified by a URL can be used). If a control point wants to create a new object whose resource needs to be imported from an external source, it can first invoke the *CreateObject()* action and then import the file.

After the *CreateObject()* action, the *res* property of the newly created object holds the following value:

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

A control point then invokes the *ImportResource()* action and a *TransferID* value (for example “1234”) is returned. The *TransferID* can be used by the control point to manipulate the transfer while it is progressing.

```
Request:
```

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

Response:
ImportResource("1234")
```

The ContentDirectory service initiates the HTTP GET to the external source and begins receiving data, which is directed to the local destination.

```
Request:
GET /song.mp3 HTTP/1.1

Response:
HTTP/1.1 200 OK
```

The control point may monitor the progress of the transfer using the *GetTransferProgress()* action:

```
Request:
GetTransferProgress("1234")

Response:
GetTransferProgress("IN_PROGRESS", 43852, 125327)
```

After the HTTP GET has finished successfully, the control point can query the result of the file transfer:

```
Request:
GetTransferProgress("1234")

Response:
GetTransferProgress("COMPLETED", 125327, 125327)
```

If a control point has subscribed to events from the ContentDirectory service, the control point receives two events from the *TransferIDs* state variable during the transfer described above:

The following event is generated when the actual transfer starts:

```
Event:
TransferIDs="1234"
```

When the transfer ends (either successfully or when it fails due to an error or is stopped by the *StopTransferResource()* action) a second event is generated:

```
Event:
TransferIDs=""
```

After the file transfer has completed successfully, the *res* property of the newly created object contains the following value (as an example):

```
<res protocolInfo="http-get:*:audio/mp3:*">
   http://10.0.0.1/cd/content?id=3
</res>
```

2.6.8.2 Transfer Using Direct HTTP POST

When the control point has direct access to the content (such as when the content is local to the control point), it is possible for a control point to post the desired content directly to the ContentDirectory service.

A control point initiates HTTP POST to the destination and begins sending the data.

```
Request:
POST /cd/content?id=3 HTTP/1.1

Response:
HTTP/1.1 200 OK
```

2.6.9 Exporting ContentDirectory Resources

There are two ContentDirectory service mechanisms defined to export content from the ContentDirectory service:

- The *ExportResource()* action.
- HTTP GET executed by the control point (only for resources that have the HTTP GET protocol specified in their *res@protocolInfo* property).

2.6.9.1 Transfer Using the *ExportResource()* Action

The source (for example `http://10.0.0.1/cd/content?id=3`) is located internal to the ContentDirectory service and the destination (for example `http://server/content?id=6`) is located externally and is identified by a URL.

For example, the *res* property of a ContentDirectory object contains the following value before the export:

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

A control point invokes the *ExportResource()* action and a *TransferID* value (for example “1235”) is returned:

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

Response:
ExportResource("1235")
```

The ContentDirectory service initiates the HTTP POST to the external destination and begins sending data from the local source.

```
Request:
POST content?id=6  HTTP/1.1

Response:
HTTP/1.1 200 OK
```

The control point may monitor the progress of the transfer using the *GetTransferProgress()* action:

```
Request:
GetTransferProgress("1235")

Response:
GetTransferProgress("IN_PROGRESS", 43852, 125327)
```

After the HTTP POST has finished successfully, the control point can query the result of the file transfer:

```
Request:
GetTransferProgress("1235")

Response:
GetTransferProgress("COMPLETED", 125327, 125327)
```

If a control point has subscribed to events from the ContentDirectory service, the control point receives two events from the *TransferIDs* state variable during the transfer described above:

The following event is generated when the actual transfer starts:

```
Event:
TransferIDs="1235"
```

When the transfer ends (either successfully or when it fails due to an error or is stopped by the *StopTransferResource()* action) a second event is generated:

```
Event:
TransferIDs=""
```

After the file transfer has completed successfully, the *res* property of the object that contains the source, is unaltered:

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

2.6.9.2 Transfer using HTTP GET

For any resource that supports the HTTP GET protocol as specified in its *res@protocolInfo* property, a control point initiates the transfer at the remote source, using HTTP GET. The resource is then copied from the ContentDirectory service to the control point.

2.6.10 Playlist Manipulation

2.6.10.1 Playlist File Representation in the ContentDirectory Service

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 ContentDirectory service, it would return a *Result* of the following form:

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="2" parentID="1" restricted="0">
      <dc:title>Playlist of John and Mary's music</dc:title>
      <dc:creator>John Jones</dc:creator>
      <upnp:class>object.item.playlistItem</upnp:class>
      <res protocolInfo="http-get:*:audio/m3u:*">
         http://pc/k.m3u
      </res>
   </item>
</DIDL-Lite>
```

2.6.10.2 Playlist File Generation

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 container object expose a *res* property, whose value is the URI of a playlist file in a format that is understood by the MediaRenderer. The content of the playlist file is a sequence of individual content items. Its internal format is identified by the *res@protocolInfo* property. 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 this:

- A *Browse()* of a *musicAlbum* object's metadata returns a *Result* of the following form:

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <container id="1" parentID="0" restricted="0"
    searchable="1">
      <dc:title>Brand New Day</dc:title>
      <dc:creator>Sting</dc:creator>
      <upnp:class>
         object.container.album.musicAlbum
      </upnp:class>
      <upnp:searchClass includeDerived="0">
         object.item.audioItem.musicTrack
      </upnp:searchClass>
      <upnp:createClass includeDerived="0">
         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 returns a *Result* of the following form:

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="2" parentID="1" restricted="0">
      <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" restricted="0">
        <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 uses the content of the *res* property on the *musicAlbum* container object in the *AVTransport::SetAVTransportURI()* action on the MediaRenderer. The MediaRenderer then issues an HTTP GET on the URI "`http://pc/genm3u?containerID="1"`" to retrieve the generated M3U resource with the following content:

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

2.6.11 Internet Content Representation

A ContentDirectory service implementation will always reside on a UPnP device. However, various URIs present as metadata inside the ContentDirectory service are allowed to point to locations, for example, web servers, that are outside the UPnP network. For example, an Internet Radio station may be represented by an object in a ContentDirectory service 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 *res* and *res@protocolInfo* properties. MediaServer devices that can serve content using multiple protocols will generally have multiple *res* properties 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 metadata:

```
<item id="InternetStream1" restricted="0">
    <dc:title>Some Stream</dc:title>
    <upnp:class>
        object.item.videoItem
    </upnp:class>
    <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.

2.6.12 Bookmark Manipulation

The *CreateObject()*, *Browse()*, and *DestroyObject()* actions are used to manipulate bookmark objects.

2.6.12.1 *bookmarkItem* Example

The following is an example of a *bookmarkItem*:

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="bookmark-763215" parentID="BC_001"
    restricted="0">
      <dc:title>Gone with the Wind</dc:title>
      <upnp:class>object.item.bookmarkItem</upnp:class>
      <upnp:deviceUDN serviceType="AVTransport:1"
       serviceId="AVTransport">
         uuid:2F5A2466-55EF-44af-953A-74DE96FF2B14
      </upnp:deviceUDN>
      <upnp:deviceUDN serviceType="RenderingControl:1"
       serviceId="RenderingControl">
         uuid:EF0DB408-3018-4e13-831A-8349CA543538
      </upnp:deviceUDN>
      <upnp:bookmarkedObjectID>1230131</upnp:bookmarkedObjectID>
      <dc:date>2003-03-21T15:21:22</dc:date>
      <upnp:stateVariableCollection serviceName="AVTransport">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;stateVariableValuePairs
          xmlns="urn:schemas-upnp-org:av:avs"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="
            urn:schemas-upnp-org:av:avs
      http://www.upnp.org/schemas/av/avs.xsd"&gt;
            &lt;stateVariable variableName="RelativeTimePosition"&gt;
               00:22:01
            &lt;/stateVariable&gt;
            &lt;!-- More state variable value pairs can
            be inserted here --&gt;
         &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

      </upnp:stateVariableCollection>
      <upnp:stateVariableCollection serviceName="RenderingControl"
       rcsInstanceType="pre-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;stateVariableValuePairs
          xmlns="urn:schemas-upnp-org:av:avs"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="
            urn:schemas-upnp-org:av:avs
      http://www.upnp.org/schemas/av/avs.xsd"&gt;
            &lt;stateVariable variableName="Brightness"&gt;
```

```
                50
             &lt;/stateVariable&gt;
             &lt;stateVariable variableName="Sharpness"&gt;
                33
             &lt;/stateVariable&gt;
             &lt;!—- More state variable value pairs can
             be inserted here --&gt;
          &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

       </upnp:stateVariableCollection>
       <upnp:stateVariableCollection serviceName="RenderingControl"
        rcsInstanceType="post-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

          &lt;?xml version="1.0" encoding="UTF-8"?&gt;
          &lt;stateVariableValuePairs
           xmlns="urn:schemas-upnp-org:av:avs"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="
             urn:schemas-upnp-org:av:avs
       http://www.upnp.org/schemas/av/avs.xsd"&gt;
             &lt;stateVariable variableName="Brightness"&gt;
                70
             &lt;/stateVariable&gt;
             &lt;stateVariable variableName="Sharpness"&gt;
                21
             &lt;/stateVariable&gt;
             &lt;!-- More state variable value pairs can
             be inserted here --&gt;
          &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

       </upnp:stateVariableCollection>
   </item>
</DIDL-Lite>
```

2.6.12.2 Creating and Destroying Bookmarks

The control point can use the *GetFeatureList()* action to determine whether the *BOOKMARK feature* is supported by the ContentDirectory service and retrieve the object IDs of all the bookmark root containers within the ContentDirectory service. After the control point has gathered state information from the relevant AVTransport and RenderingControl services through the respective *GetStateVariables()* actions, the control point can then decide where to create the bookmark. It can then proceed and create the new bookmark within one of the exposed bookmark subtrees. Alternatively, the control point may decide to create the bookmark outside the bookmark subtrees. Bookmark items can be created anywhere in the ContentDirectory data structure. However, the ContentDirectory MUST create a reference item to that bookmark within one of the exposed bookmark subtrees. The location of that reference item within the bookmark subtrees is vendor dependent.

If a bookmark container has its *@restricted* property set to "*0*" and its *upnp:createClass* set to "*object.container.bookmarkFolder*", then only a bookmark container can be created. If a bookmark container has its *@restricted* property set to "*0*" and its *upnp:createClass* set to "*object.container.bookmarkItem*", then only a bookmark item can be created. If a bookmark container has its *@restricted* property set to "*0*" and the *upnp:createClass* property is not specified, then both bookmark container and bookmark item can be created within that container.

The *Elements* input argument of the *CreateObject()* action contains the title of the bookmark (*dc:title*), the UDN of the device that contains the AVTransport service, the UDN of the device that contains the RenderingControl service, the bookmark timestamp, and the respective state snapshots. The output of the *CreateObject()* action contains the object ID of the newly created bookmark object in the *ObjectID* output argument and the *DIDL-Lite XML Document*, describing the created bookmark object in the *Result* output argument.

The following paragraph shows an example invocation of the *CreateObject()* action. When a bookmark is created, the associated content item must be updated to contain the object ID of the newly created bookmark in its *upnp:bookmarkID* property. In this example, “BC_001” is used as the parent container’s object ID.

```
Request:
CreateObject("BC_001", "
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="" parentID="BC_001" restricted="0">
      <dc:title>Gone with the wind</dc:title>
      <upnp:class>object.item.bookmarkItem</upnp:class>
      <upnp:deviceUDN serviceType="AVTransport:1"
       serviceId="AVTransport">
         uuid:2F5A2466-55EF-44af-953A-74DE96FF2B14
      </upnp:deviceUDN>
      <upnp:deviceUDN serviceType="RenderingControl:1"
       serviceId="RenderingControl">
         uuid:EF0DB408-3018-4e13-831A-8349CA543538
      </upnp:deviceUDN>
      <upnp:bookmarkedObjectID>1230131</upnp:bookmarkedObjectID>
      <upnp:stateVariableCollection serviceName="AVTransport">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;stateVariableValuePairs
          xmlns="urn:schemas-upnp-org:av:avs"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="
             urn:schemas-upnp-org:av:avs
       http://www.upnp.org/schemas/av/avs.xsd"&gt;
             &lt;stateVariable variableName="RelativeTimePosition"&gt;
                00:22:01
             &lt;stateVariable&gt;
             &lt;!-- More state variable value pairs can
             be inserted here --&gt;
         &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

      </upnp:stateVariableCollection>
      <upnp:stateVariableCollection serviceName="RenderingControl"
       rcsInstanceType="pre-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->
```

```
           &lt;?xml version="1.0" encoding="UTF-8"?&gt;
           &lt;stateVariableValuePairs
            xmlns="urn:schemas-upnp-org:av:avs"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="
              urn:schemas-upnp-org:av:avs
       http://www.upnp.org/schemas/av/avs.xsd"&gt;
              &lt;stateVariable variableName="Brightness"&gt;
                 50
              &lt;/stateVariable&gt;
              &lt;stateVariable variableName="Sharpness"&gt;
                 33
              &lt;/stateVariable&gt;
              &lt;!-- More state variable value pairs can
              be inserted here --&gt;
           &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

       </upnp:stateVariableCollection>
       <upnp:stateVariableCollection serviceName="RenderingControl"
        rcsInstanceType="post-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

           &lt;?xml version="1.0" encoding="UTF-8"?&gt;
           &lt;stateVariableValuePairs
            xmlns="urn:schemas-upnp-org:av:avs"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="
              urn:schemas-upnp-org:av:avs
       http://www.upnp.org/schemas/av/avs.xsd"&gt;
              &lt;stateVariable variableName="Brightness"&gt;
                 70
              &lt;/stateVariable&gt;
              &lt;stateVariable variableName="Sharpness"&gt;
                 21
              &lt;/stateVariable&gt;
              &lt;!-- More state variable value pairs can
              be inserted here --&gt;
           &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

       </upnp:stateVariableCollection>
   </item>
</DIDL-Lite>")

Response:
CreateObject("bookmark-763215", "
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="bookmark-763215" parentID="BC_001" restricted="0">
      <dc:title>Gone with the wind</dc:title>
      <upnp:class>object.item.bookmarkItem</upnp:class>
      <upnp:deviceUDN serviceType="AVTransport:1"
       serviceId="AVTransport">
```

```
         uuid:2F5A2466-55EF-44af-953A-74DE96FF2B14
      </upnp:deviceUDN>
      <upnp:deviceUDN serviceType="RenderingControl:1"
       serviceId="RenderingControl">
         uuid:EF0DB408-3018-4e13-831A-8349CA543538
      </upnp:deviceUDN>
      <upnp:bookmarkedObjectID>1230131</upnp:bookmarkedObjectID>
      <dc:date>2003-04-21T15:33:44</dc:date>
      <upnp:stateVariableCollection serviceName="AVTransport">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;stateVariableValuePairs
          xmlns="urn:schemas-upnp-org:av:avs"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="
            urn:schemas-upnp-org:av:avs
      http://www.upnp.org/schemas/av/avs.xsd"&gt;
            &lt;stateVariable variableName="RelativeTimePosition"&gt;
               00:22:01
            &lt;stateVariable&gt;
            &lt;!-- More state variable value pairs can
            be inserted here --&gt;
         &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

      </upnp:stateVariableCollection>
      <upnp:stateVariableCollection serviceName="RenderingControl"
       rcsInstanceType="pre-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;stateVariableValuePairs
          xmlns="urn:schemas-upnp-org:av:avs"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="
            urn:schemas-upnp-org:av:avs
      http://www.upnp.org/schemas/av/avs.xsd"&gt;
            &lt;stateVariable variableName="Brightness"&gt;
               50
            &lt;/stateVariable&gt;
            &lt;stateVariable variableName="Sharpness"&gt;
               33
            &lt;/stateVariable&gt;
            &lt;!-- More state variable value pairs can
            be inserted here --&gt;
         &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

      </upnp:stateVariableCollection>
      <upnp:stateVariableCollection serviceName="RenderingControl"
       rcsInstanceType="post-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;stateVariableValuePairs
```

```
            xmlns="urn:schemas-upnp-org:av:avs"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="
              urn:schemas-upnp-org:av:avs
         http://www.upnp.org/schemas/av/avs.xsd"&gt;
              &lt;stateVariable variableName="Brightness"&gt;
                 70
              &lt;/stateVariable&gt;
              &lt;stateVariable variableName="Sharpness"&gt;
                 21
              &lt;/stateVariable&gt;
              &lt;!-- More state variable value pairs can
              be inserted here --&gt;
           &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

         </upnp:stateVariableCollection>
      </item>
</DIDL-Lite>")
```

The *DestroyObject()* action destroys the bookmark object specified by the *ObjectID* argument. The ContentDirectory service is REQUIRED to maintain consistency; that is, when a bookmark is destroyed, the associated content item's *upnp:bookmarkID* property MUST be removed. Likewise, when a content item that contains bookmark references is destroyed, the corresponding bookmark items (and their reference items, if any) MUST also be destroyed.

The following is an example of a *DestroyObject()* action invocation:

```
Request:
DestroyObject("bookmark-763215")

Response:
DestroyObject()
```

2.6.12.3 Browsing Bookmarks

The bookmark list is obtained by invoking the *Browse()* action with the *BrowseFlag* argument set to "*BrowseDirectChildren*". The *GetFeatureList()* action MAY be used to find the bookmark root containers in the ContentDirectory service.

The following is an example where "BC_001" is used as the parent container's object ID.

```
Request:
Browse("BC_001", "BrowseDirectChildren", "*", 0, 2, "")

Response:
Browse("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="bookmark-00001" parentID="BC_001" restricted="0">
      <dc:title>The Matrix</dc:title>
      <upnp:class>object.item.bookmarkItem</upnp:class>
      <upnp:deviceUDN serviceType="AVTransport:1"
       serviceId="AVTransport">
```

```
         uuid:2F5A2466-55EF-44af-953A-74DE96FF2B14
      </upnp:deviceUDN>
      <upnp:deviceUDN serviceType="RenderingControl:1"
       serviceId="RenderingControl">
         uuid:EF0DB408-3018-4e13-831A-8349CA543538
      </upnp:deviceUDN>
      <upnp:bookmarkedObjectID>1230131</upnp:bookmarkedObjectID>
      <dc:date>2003-04-21T15:33:44</dc:date>
      <upnp:stateVariableCollection serviceName="AVTransport">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;stateVariableValuePairs
          xmlns="urn:schemas-upnp-org:av:avs"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="
            urn:schemas-upnp-org:av:avs
      http://www.upnp.org/schemas/av/avs.xsd"&gt;
            &lt;stateVariable variableName="RelativeTimePosition"&gt;
               01:01:21
            &lt;stateVariable&gt;
            &lt;!—- More state variable value pairs can
            be inserted here --&gt;
         &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

      </upnp:stateVariableCollection>
      <upnp:stateVariableCollection serviceName="RenderingControl"
       rcsInstanceType="pre-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;stateVariableValuePairs
          xmlns="urn:schemas-upnp-org:av:avs"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="
            urn:schemas-upnp-org:av:avs
      http://www.upnp.org/schemas/av/avs.xsd"&gt;
            &lt;stateVariable variableName="Brightness"&gt;
               40
            &lt;/stateVariable&gt;
            &lt;stateVariable variableName="Sharpness"&gt;
               27
            &lt;/stateVariable&gt;
            &lt;!—- More state variable value pairs can
            be inserted here --&gt;
         &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

      </upnp:stateVariableCollection>
      <upnp:stateVariableCollection serviceName="RenderingControl"
       rcsInstanceType="post-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;stateVariableValuePairs
```

```
            xmlns="urn:schemas-upnp-org:av:avs"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="
              urn:schemas-upnp-org:av:avs
        http://www.upnp.org/schemas/av/avs.xsd"&gt;
              &lt;stateVariable variableName="Brightness"&gt;
                 70
              &lt;/stateVariable&gt;
              &lt;stateVariable variableName="Sharpness"&gt;
                 21
              &lt;/stateVariable&gt;
              &lt;!—- More state variable value pairs can
              be inserted here --&gt;
           &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

       </upnp:stateVariableCollection>
    </item>
    <item id="bookmark-00002" parentID="BC_001" restricted="0">
       <dc:title>The Matrix Reloaded</dc:title>
       <upnp:class>object.item.bookmarkItem</upnp:class>
       <upnp:deviceUDN serviceType="AVTransport:1"
        serviceId="AVTransport">
          uuid:858733A8-E64C-4a2b-A407-38518D96AA0E
       </upnp:deviceUDN>
       <upnp:deviceUDN serviceType="RenderingControl:1"
        serviceId="RenderingControl">
          uuid:65AD5B9D-557E-4ddb-8EDE-F5A4C5190E57
       </upnp:deviceUDN>
       <upnp:bookmarkedObjectID>1230131</upnp:bookmarkedObjectID>
       <dc:date>2003-04-18T15:33:44</dc:date>
       <upnp:stateVariableCollection serviceName="AVTransport">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

          &lt;?xml version="1.0" encoding="UTF-8"?&gt;
          &lt;stateVariableValuePairs
           xmlns="urn:schemas-upnp-org:av:avs"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="
             urn:schemas-upnp-org:av:avs
       http://www.upnp.org/schemas/av/avs.xsd"&gt;
             &lt;stateVariable variableName="RelativeTimePosition"&gt;
                01:55:22
             &lt;/stateVariable&gt;
             &lt;!—- More state variable value pairs can
             be inserted here --&gt;
          &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

       </upnp:stateVariableCollection>
       <upnp:stateVariableCollection serviceName="RenderingControl"
        rcsInstanceType="pre-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

          &lt;?xml version="1.0" encoding="UTF-8"?&gt;
          &lt;stateVariableValuePairs
           xmlns="urn:schemas-upnp-org:av:avs"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="
             urn:schemas-upnp-org:av:avs
```

```
      http://www.upnp.org/schemas/av/avs.xsd"&gt;
            &lt;stateVariable variableName="Brightness"&gt;
               30
            &lt;/stateVariable&gt;
            &lt;stateVariable variableName="Sharpness"&gt;
               23
            &lt;/stateVariable&gt;
            &lt;!—- More state variable value pairs can
            be inserted here --&gt;
         &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

      </upnp:stateVariableCollection>
      <upnp:stateVariableCollection serviceName="RenderingControl"
       rcsInstanceType="post-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

         &lt;?xml version="1.0" encoding="UTF-8"?&gt;
         &lt;stateVariableValuePairs
          xmlns="urn:schemas-upnp-org:av:avs"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="
            urn:schemas-upnp-org:av:avs
      http://www.upnp.org/schemas/av/avs.xsd"&gt;
            &lt;stateVariable variableName="Brightness"&gt;
               70
            &lt;/stateVariable&gt;
            &lt;stateVariable variableName="Sharpness"&gt;
               21
            &lt;/stateVariable&gt;
            &lt;!—- More state variable value pairs can
            be inserted here --&gt;
         &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

      </upnp:stateVariableCollection>
   </item>
</DIDL-Lite>", 2, 2, 10)
```

Utilizing filters will reduce the size of the response. Once the user has selected a certain bookmark, another *Browse()* action can be invoked to obtain the rest of the bookmark information:

```
Request:
Browse("BC_001","BrowseDirectChildren","@id,@parentId,@restricted,
dc:title,upnp:class,dc:date",0,2,"")

Response:
Browse("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="bookmark-00001" parentID="BC_001" restricted="0">
      <dc:title>The Matrix</dc:title>
```

```
      <upnp:class>object.item.bookmarkItem</upnp:class>
      <dc:date>2003-04-13T15:33:44</dc:date>
   </item>
   <item id="bookmark-00002" parentID="BC_001" restricted="0">
      <dc:title>The Matrix Reloaded</dc:title>
      <upnp:class>object.item.bookmarkItem</upnp:class>
      <dc:date>2003-04-22T15:33:44</dc:date>
   </item>
</DIDL-Lite>", 2, 2, 10)
```

The following example shows how to browse the container metadata of a bookmark container:

```
Request:
Browse("BC_001", "BrowseMetadata", "*", 0, 0, "")

Response:
Browse("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <container id="BC_001" parentID="0" restricted="0"
    neverPlayable="1">
      <dc:title>BookMark Container</dc:title>
      <upnp:class>object.container.bookmarkFolder</upnp:class>
   </container>
</DIDL-Lite>, 1, 1, 20)
```

To obtain a particular bookmark, the bookmark id MUST be provided in the *ObjectID* argument and the *BrowseFlag* argument must be set to "*BrowseMetadata*". The following is an example:

```
Request:
Browse("bookmark-00001", "BrowseMetadata", "*", 0, 0, "")

Response:
Browse("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="bookmark-00001" parentID="BC_001" restricted="0">
      <dc:title>The Matrix</dc:title>
      <upnp:class>object.item.bookmarkItem</upnp:class>
      <upnp:deviceUDN serviceType="AVTransport:1"
       serviceId="AVTransport">
         uuid:2F5A2466-55EF-44af-953A-74DE96FF2B14
      </upnp:deviceUDN>
      <upnp:deviceUDN serviceType="RenderingControl:1"
       serviceId="RenderingControl">
         uuid:EF0DB408-3018-4e13-831A-8349CA543538
      </upnp:deviceUDN>
      <upnp:bookmarkedObjectID>1230131</upnp:bookmarkedObjectID>
```

```
       <dc:date>2003-04-17T15:33:44</dc:date>
       <upnp:stateVariableCollection serviceName="AVTransport">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

          &lt;?xml version="1.0" encoding="UTF-8"?&gt;
          &lt;stateVariableValuePairs
           xmlns="urn:schemas-upnp-org:av:avs"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="
             urn:schemas-upnp-org:av:avs
       http://www.upnp.org/schemas/av/avs.xsd"&gt;
             &lt;stateVariable variableName="RelativeTimePosition"&gt;
                01:01:21
             &lt;stateVariable&gt;
             &lt;!—- More state variable value pairs can
             be inserted here --&gt;
          &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

       </upnp:stateVariableCollection>
       <upnp:stateVariableCollection serviceName="RenderingControl"
        rcsInstanceType="pre-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

          &lt;?xml version="1.0" encoding="UTF-8"?&gt;
          &lt;stateVariableValuePairs
           xmlns="urn:schemas-upnp-org:av:avs"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="
             urn:schemas-upnp-org:av:avs
       http://www.upnp.org/schemas/av/avs.xsd"&gt;
             &lt;stateVariable variableName="Brightness"&gt;
                40
             &lt;/stateVariable&gt;
             &lt;stateVariable variableName="Sharpness"&gt;
                27
             &lt;/stateVariable&gt;
             &lt;!-- More state variable value pairs can
             be inserted here --&gt;
          &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

       </upnp:stateVariableCollection>
       <upnp:stateVariableCollection serviceName="RenderingControl"
        rcsInstanceType="post-mix">

<!--
The following stateVariableValuePairs XML Document needs to be interpreted as a
simple string and therefore needs to be properly escaped
-->

          &lt;?xml version="1.0" encoding="UTF-8"?&gt;
          &lt;stateVariableValuePairs
           xmlns="urn:schemas-upnp-org:av:avs"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="
             urn:schemas-upnp-org:av:avs
       http://www.upnp.org/schemas/av/avs.xsd"&gt;
             &lt;stateVariable variableName="Brightness"&gt;
                70
```

```
            &lt;/stateVariable&gt;
            &lt;stateVariable variableName="Sharpness"&gt;
               21
            &lt;/stateVariable&gt;
            &lt;!—- More state variable value pairs can
            be inserted here --&gt;
         &lt;/stateVariableValuePairs&gt;

<!-- End of stateVariableValuePairs XML Document -->

      </upnp:stateVariableCollection>
   </item>
</DIDL-Lite>, 1, 1, 30)
```

2.6.13 Processing FreeForm Queries

The *FreeFormQuery()* action is designed to allow a control point to extract freely any piece of information available from the ContentDirectory service. The control point creates an XQuery request that will be executed on a set of ContentDirectory objects organized as indicated by the *CDSView.* The control point should first invoke the *GetFreeFormQueryCapabilities()* action to determine which properties can be used in the XQuery request. Upon completion, the result of the processing is returned to the control point. Note that the control point is solely responsible for the type of information that is returned. The XQuery request created by the control point determines among others, the syntax, the formatting and the sort order of the returned information.

The following subclauses provide some examples on the use and syntax of *XQuery XML Documents* in the context of the *FreeFormQuery()* action. The examples are based on a ContentDirectory hierarchy as outlined in Annex F, “ **(informative)**

Example ContentDirectory Hierarchy”

2.6.13.1 Retrieving the title of all music albums

The following request queries for the title of all available music albums. In other words, it retrieves all *dc:title* container property values for which the *upnp:class* property equals “*object.container.album.musicAlbum*”. Note that the result is a simple node set of *dc:title* values, potentially with duplicates. The result is most likely meaningless in a ContentDirectory service context and is merely provided to illustrate the flexibility and power of the *FreeFormQuery()* action. In this example the result is *not* a valid *DIDL-Lite XML Document*.

```
Request:
FreeFormQuery("0", "0", "
<albums
 xmlns:didl-lite="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/">
{
   //didl-lite:container[upnp:class="object.container.album.musicAlbum"]
   /dc:title
}
</albums>")

Response:
FreeFormQuery("
<?xml version="1.0" encoding="UTF-8"?>
<albums
 xmlns:didl-lite="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/">
   <dc:title>Album 1</dc:title>
   <dc:title>Album 2</dc:title>
</albums>
", 18)
```

2.6.13.2 Retrieving the audio items of Album 1

The following request queries for all items for which the *upnp:class* property equals "*object.item.audioItem*" and for which the *@parentID* property is equal to the *@id* property of the container(s) that have their *dc:title* property set to "Album 1". The result of the query is formatted to comply with the DIDL-Lite syntax so that the final output is a valid *DIDL-Lite XML Document*.

```
Request:
FreeFormQuery("0", "0", "
<DIDL-Lite
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:didl-lite="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/">
{
   for $object in //didl-lite:item[upnp:class = "object.item.audioItem"]
   let $containerId := $object/@parentID
   where //didl-lite:container[@id=$containerId and dc:title="Album 1"]
   return $object
}
</DIDL-Lite>")

Response:
FreeFormQuery("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:didl-lite="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/">
   <item id="1-1-1-1" parentID="1-1-1" restricted="1">
      <dc:title>Album 1 Song 1</dc:title>
      <upnp:class>object.item.audioItem</upnp:class>
      <upnp:album>Album 1</upnp:album>
      <upnp:genre>Unknown</upnp:genre>
      <upnp:artist>Unknown</upnp:artist>
      <res protocolInfo="http-get:*:audio/mpeg:*">
         http://10.0.0.1/audio/O-MP3-11.mp3
      </res>
   </item>
   <item id="1-1-1-2" parentID="1-1-1" restricted="1">
      <dc:title>Album 1 Song 2</dc:title>
      <upnp:class>object.item.audioItem</upnp:class>
      <upnp:album>Album 1</upnp:album>
      <upnp:genre>Unknown</upnp:genre>
      <upnp:artist>Unknown</upnp:artist>
      <res protocolInfo="http-get:*:audio/mpeg:*">
         http://10.0.0.1/audio/O-MP3-12.mp3
      </res>
   </item>
   <item id="1-1-1-3" parentID="1-1-1" restricted="1">
      <dc:title>Album 1 Song 3</dc:title>
      <upnp:class>object.item.audioItem</upnp:class>
      <upnp:album>Album 1</upnp:album>
      <upnp:genre>Unknown</upnp:genre>
      <upnp:artist>Unknown</upnp:artist>
      <res protocolInfo="http-get:*:audio/mpeg:*">
         http://10.0.0.1/audio/O-MP3-13.mp3
      </res>
   </item>

<…Additional results omitted…>

</DIDL-Lite>", 18)
```

2.6.13.3 Retrieving a limited number of photo items

The following request queries for items that have their *upnp:class* property set to "*object.item.imageItem.*". Only the second half-dozen items are returned, that is: those items for which the item's *position()* is in the range [7-12]. The result of the query is formatted to comply with the DIDL-Lite syntax so that the final output is a valid *DIDL-Lite XML Document*.

Note: The less than ("<") and greater than (">") characters in the position predicate need to be escaped in order to provide a valid *XQuery Stylesheet XML Document* to the XQuery processor.

```
Request:
FreeFormQuery("0", "0", "
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:didl-lite="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/">
{
   //didl-lite:item[upnp:class="object.item.imageItem"]
      [position() > 6 and position() < 12]
}
")

Response:
FreeFormQuery("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:didl-lite="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/">
   <item id="3-1-7" parentID="3-1" restricted="1">
      <dc:title>Album 1 Photo 7</dc:title>
      <upnp:class>object.item.imageItem</upnp:class>
      <upnp:album>Album 1</upnp:album>
      <res protocolInfo="http-get:*:image/jpeg:*">
         http://10.0.0.1/image/B-JPEG_M-17.jpg
      </res>
   </item>
   <item id="3-1-8" parentID="3-1" restricted="1">
      <dc:title>Album 1 Photo 8</dc:title>
      <upnp:class>object.item.imageItem</upnp:class>
      <upnp:album>Album 1</upnp:album>
      <res protocolInfo="http-get:*:image/jpeg:*">
         http://10.0.0.1/image/B-JPEG_M-18.jpg
      </res>
   </item>
   <item id="3-1-9" parentID="3-1" restricted="1">
      <dc:title>Album 1 Photo 9</dc:title>
      <upnp:class>object.item.imageItem</upnp:class>
      <upnp:album>Album 1</upnp:album>
      <res protocolInfo="http-get:*:image/jpeg:*">
         http://10.0.0.1/image/B-JPEG_M-19.jpg
      </res>
   </item>
   <item id="3-1-10" parentID="3-1" restricted="1">
      <dc:title>Album 1 Photo 10</dc:title>
      <upnp:class>object.item.imageItem</upnp:class>
      <upnp:album>Album 1</upnp:album>
      <res protocolInfo="http-get:*:image/jpeg:*">
         http://10.0.0.1/image/B-JPEG_M-110.jpg
      </res>
   </item>
   <item id="3-1-11" parentID="3-1" restricted="1">
      <dc:title>Album 1 Photo 11</dc:title>
      <upnp:class>object.item.imageItem</upnp:class>
```

```
      <upnp:album>Album 1</upnp:album>
      <res protocolInfo="http-get:*:image/jpeg:*">
         http://10.0.0.1/image/B-JPEG_M-111.jpg
      </res>
   </item>
<DIDL-Lite>", 18)
```

2.6.14 Foreign Metadata

2.6.14.1 Determining the Supported Foreign Metadata Types

Different ContentDirectory service implementations may support different types of foreign metadata. In order for a control point to take advantage of any foreign metadata that is included within an object, the control point must be able to parse and interpret the foreign metadata's format. If the control point does not understand the foreign metadata type then it will not be able to process the foreign metadata. Therefore, to determine if a given implementation supports a specific foreign metadata type (one of the types that the control point understands), the control point can use the GetFeatureList() action to enumerate the foreign metadata types that are supported by the ContentDirectory service.

The following example shows how a control point can retrieve the list of foreign metadata types that are supported by a ContentDirectory service. As shown below, the *Features XML Document* returned by the *GetFeatureList()* action indicates that this particular implementation supports the *FOREIGN_METADATA feature* and is capable of returning foreign metadata based on the "acme.org_MD1" and "acme.org_MD2" metadata types. Additionally, this particular implementation obtains its foreign metadata from the "acme_metadata.org" service provider.

```
Request:
GetFeatureList()

Response:
GetFeatureList("
<?xml version="1.0" encoding="UTF-8"?>
<Features
 xmlns="urn:schemas-upnp-org:av:avs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:avs
    http://www.upnp.org/schemas/av/avs.xsd">
   <Feature name="FOREIGN_METADATA" version="1">
      <type id="acme.org_MD1" provider="acme_metadata.org"></type>
      <type id="acme.org_MD2" provider="acme_metadata.org"></type>
   </Feature>
</Features>")
```

2.6.14.2 Determining Whether an Object Contains Foreign Metadata

When a control point retrieves an object (for example, via the *Browse()* or *Search()* action) and is able to process certain types of foreign metadata, the control point will need to determine if the returned object contains any foreign metadata corresponding to one of the types supported by the control point. When an object is returned, the control point first needs to determine if the object contains one or more instances of the *upnp:foreignMetadata* property. If not, then the object does not contain any foreign metadata.

However, if the object does contain one or more *upnp:foreignMetadata* properties, the control point needs to examine each instance of the *upnp:foreignMetadata@type* property to determine if any of them match the foreign metadata types that are supported by the control point. If not, then the object does not contain any foreign metadata that the control point is capable of processing. However, if a match exists, then the control point can extract and process the actual foreign metadata via the *upnp:foreignMetadata::fmBody* properties.

The following example shows how a control point can determine if a returned object includes any foreign metadata that the control point is capable of processing. In this particular example, the foreign metadata that is returned is identified as foreign metadata of type “openepg.org_v1” and the actual foreign metadata is retrieved via the *upnp:foreignMetadata::fmBody::embeddedXML* property.

```
Request:
Browse("BC_001", "BrowseMetadata", "*", 0, 0, "")

Response:
Browse("
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp.xsd">
   <item id="BC_001" parentID="3" restricted="1">
      <dc:title>ABC Nightly News</dc:title>
      <upnp:class>object.item.epgItem</upnp:class>
      <upnp:longDescription>
         News of the day for January 6th 2006
      </upnp:longDescription>
      <upnp:channelID type="ANALOG"
       distriNetworkName="ECHOSTAR"
       distriNetworkID="DISH">13</upnp:channelID>
      <upnp:channelName>ABC New York</upnp:channelName>
      <upnp:scheduledStartTime usage="SCHEDULED_PROGRAM">
      </upnp:scheduledStartTime>
         2006-01-06T23:59:59-8:00
      <upnp:scheduledEndTime daylightSaving="">
         2006-01-07T00:29:59Z-8:00
      </upnp:scheduledEndTime>
      <upnp:scheduledDuration>P0D00:30:00</upnp:scheduledDuration>
      <upnp:channelGroupName id="DISH">EchoStar</upnp:channelGroupName>
      <upnp:foreignMetadata type="openepg.org_v1">
         <upnp:fmId>1234567890</upnp:fmId>
         <upnp:fmClass></upnp:fmClass>
         <upnp:fmProvider>acme.org</upnp:fmProvider>
         <upnp:fmBody xmlFlag="1" mimeType="text/xml">
            <upnp:fmEmbeddedXML>
               <OpenEpg
                  xmlns="urn:ce:cea-2033:OpenEPG:2006"
                  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                  xsi:schemaLocation=".\OpenEPG-V1.xsd">
                    <DistributionNetwork distributionNetworkId="DISH">
                       <Name>EchoStar</Name>
                       <ContentService ContentServiceSourceId="ABC">
                          <ContentServiceMapping>
                             <Channel>13</Channel>
                             <MinorChannel>0</MinorChannel>
                          </ContentServiceMapping>
                       </ContentService>
                    </DistributionNetwork>
                    <ContentServiceSource contentServiceSourceId="ABC">
                       <CallSign>WABC</CallSign>
                       <Name>ABC New York</Name>
                       <Event eventId="1234567890">
                          <StartTime>
                             2006-01-06T23:59:59-8:00
                          </StartTime>
                          <Duration>P0DT00H30M00S</Duration>
                          <ContentCRID crid="ABC://NightlyNews/6-jan-2006"/>
                       </Event>
```

```
                    </ContentServiceSource>
                    <Content crid="ABC://NightlyNews/6-jan-2006">
                        <ShortTitle xml:lang="en-us">
                            ABC Nightly News
                        </ShortTitle>
                        <ShortDescription xml:lang="en-us">
                            News of the day for January 6th 2006
                        </ShortDescription>
                    </Content>
                </OpenEpg>
            </upnp:fmEmbeddedXML>
        </upnp:fmBody>
    </upnp:foreignMetadata>
</item>
</DIDL-Lite>, 1, 1, 20)
```

2.6.15 Monitoring Changes

The following scenarios assume that the ContentDirectory service implementation supports the *Tracking Changes Option* and that the value of the ContentDirectory service implementation's *ServiceResetToken* state variable remains constant. If the control point detects that the value of the *ServiceResetToken* state variable has changed, it should invalidate any cached information about that ContentDirectory service implementation.

2.6.15.1 Monitoring Changes while *on-line*

2.6.15.1.1 Monitoring Individual Changes

The following example shows the type of information that a ContentDirectory service implementation will make available to control points that are *on-line* when individual objects are added, modified, or deleted. Control points that wish to track changes to a ContentDirectory service implementation can use the ContentDirectory service's *LastChange* state variable to receive event notifications indicating which objects within the ContentDirectory service hierarchy have changed. Once a control point has subscribed to events (using the normal UPnP event subscription mechanism), updates to the *LastChange* state variable are evented to the control point. The *LastChange* state variable will identify the objects that have been modified since the end of the previous moderation period. The following example sequence illustrates the *LastChange* events that are generated by a ContentDirectory service implementation when various changes occur.

Example Sequence:

0) Device is installed on the network for the first time.
1) Control point subscribes to events.
2) New container object is created in the Root container.
3) New object is created in a new container.
4) Another new object is created in the new container and the first object is deleted from the new container.
5) Moderation period expires.
6) New object is created
7) Subscription is cancelled.
8) Event subscription.

Time T0: Device First Installed:

The device containing the ContentDirectory service is attached to the network for the first time.

SystemUpdateID = 100

(100 is used as an example and also represents the maximum initial value of all of the *upnp:objectUpdateID* properties of objects within the ContentDirectory service)

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent>
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
</StateEvent>
```

GENA behavior:

None – No GENA requirements at installation time.

Note: The device's event moderation timer may be started now. However, in this example, the moderation timer is started some time later.

Time T1: Initial Event Subscription:

A first control point (since power-up) subscribes to ContentDirectory service events.

SystemUpdateID = 100

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
</StateEvent>
```

GENA behavior:

Event the initial **Notify** message for the *LastChange* state variable. The contents of the `<StateEvent>` element is empty since this is the first subscriber.

Note: The device's event moderation timer may be started now. However, in this example, the moderation timer is started some time later.

Time T2: Container created in root container:

A new container object (*@id*="Album001") is created as a child of the root container (*@id*="0"). The *LastChange* state variable is updated to reflect the new object plus the modification of the *@childCount* property in the root container.

SystemUpdateID = 102 (after the container is created)

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
   <objAdd objID="Album001" updateID="101" objParentId="0"
    objClass="object.container.album.musicAlbum" stUpdate="0"/>
   <objMod objID="0" updateID="102" stUpdate="0"/>
</StateEvent>
```

GENA behavior:

Event a **Notify** message with the current value of the *LastChange* state variable and start the moderation timer.

Note: In this example, the moderation timer is started when the first event is actually sent to the first subscriber. However, if desired, the moderation timer may be started some time earlier in which case this event would be delayed until the moderation timer expires.

Time T3: Child Object Created:

A new object (*@id*="Song001") is created as a child of the newly created container (*@id*="Album001") whose *@childCount* property is updated to reflect the presence of the new child object.

SystemUpdateID = 104 (after the objects are created)

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
   <objAdd objID="Album001" updateID="101" objParentId="0"
    objClass="object.container.album.musicAlbum" stUpdate="0"/>
   <objMod objID="0" updateID="102" stUpdate="0"/>
   <objAdd objID="Song001" updateID="103" objParentId="Album001"
    objClass="object.item.audioItem" stUpdate="0"/>
   <objMod objID="Album001" updateID="104" stUpdate="0"/>
</StateEvent>
```

Note: Since the event moderation period has not yet expired, `updateID` values 101-102 are left over from time T2.

GENA behavior:

None – The current moderation period needs to expire before the event is sent out.

Time-T4: One child object created and one object deleted within one moderation period:

Another new object (*@id* = "Song002") is created as a child of the newly created container (*@id* = "Album001") whose *@childCount* property is updated to reflect the presence of the new child object. Within the same moderation period, the first child object (*@id* = "Song001") is deleted from the newly created container (*@id* = "Album001"). The container's *@childCount* and *upnp:totalDeletedChildCount* properties are updated again to reflect the removal of the child object (*@id* = "Song001").

SystemUpdateID = 108 (after the object is created and the other object is deleted)

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
   <objAdd objID="Album001" updateID="101" objParentId="0"
    objClass="object.container.album.musicAlbum" stUpdate="0"/>
   <objMod objID="0" updateID="102" stUpdate="0"/>
   <objAdd objID="Song001" updateID="103" objParentId="Album001"
    objClass="object.item.audioItem" stUpdate="0"/>
   <objMod objID="Album001" updateID="104" stUpdate="0"/>
   <objAdd objID="Song002" updateID="105" objParentId="Album001"
    objClass="object.item.audioItem" stUpdate="0"/>
   <objMod objID="Album001" updateID="106" stUpdate="0"/>
   <objDel objID="Song001" updateID="107" stUpdate="0"/>
   <objMod objID="Album001" updateID="108" stUpdate="0"/>
</StateEvent>
```

Note: Since the event moderation period has not yet expired, `updateID` values 101-102 are left over from time T2 and `updateID` values 103-104 are left over from time T3.

GENA behavior:

None – The current moderation period needs to expire before the event is sent out.

Time T5: Moderation Period Expires

The event moderation period expires. The changes that have occurred since the previous event will be sent to those control points that have subscribed for events.

SystemUpdateID = 108

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
   <objAdd objID="Album001" updateID="101" objParentId="0"
    objClass="object.container.album.musicAlbum" stUpdate="0"/>
   <objMod objID="0" updateID="102" stUpdate="0"/>
   <objAdd objID="Song001" updateID="103" objParentId="Album001"
    objClass="object.item.audioItem" stUpdate="0"/>
   <objMod objID="Album001" updateID="104" stUpdate="0"/>
   <objAdd objID="Song002" updateID="105" objParentId="Album001"
    objClass="object.item.audioItem" stUpdate="0"/>
   <objMod objID="Album001" updateID="106" stUpdate="0"/>
   <objDel objID="Song001" updateID="107" stUpdate="0"/>
   <objMod objID="Album001" updateID="108" stUpdate="0"/>
</StateEvent>
```

GENA behavior:

Event a **Notify** message with the current value of the *LastChange* state variable. The event moderation timer is re-started to prevent any subsequent event from being sent out too soon. The value of *LastChange* remains unmodified until the next event occurs.

Time T6: Child object created:

A new object (*@id* = “Song003”) is created as a child of an existing container (*@id* = “Album001”). The container’s *@childCount* property is updated to reflect the presence of the new child object.

SystemUpdateID = 110 (after the object is created)

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
   <objAdd objID="Song003" updateID="109" objParentId="Album001"
    objClass="object.item.audioItem" stUpdate="0"/>
   <objMod objID="Album001" updateID="110" stUpdate="0"/>
</StateEvent>
```

GENA behavior:

None – The current moderation period needs to expire before the event is sent out. However, since this is the first event following the completion of the moderation period, the previously evented value of the *LastChange* state variable is replaced with the new event.

Time T7: Unsubscribe:

The last remaining control point subscribed to ContentDirectory service events unsubscribes itself.

SystemUpdateID = 110

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
   <objAdd objID="Song003" updateID="109" objParentId="Album001"
    objClass="object.item.audioItem" stUpdate="0"/>
   <objMod objID="Album001" updateID="110" stUpdate="0"/>
</StateEvent>
```

Note: Since the event moderation period has not yet expired, `updateID` values 109-110 are left over from time T6.

GENA behavior:

None – The moderation period has not yet expired plus there are no control points subscribed to ContentDirectory events.

Time T8: Event Subscription:

Before the moderation period expires, a control point subscribes to ContentDirectory service events.

SystemUpdateID = 110

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
   <objAdd objID="Song003" updateID="109" objParentId="Album001"
    objClass="object.item.audioItem" stUpdate="0"/>
   <objMod objID="Album001" updateID="110" stUpdate="0"/>
</StateEvent>
```

Note: Since the event moderation period has not yet expired, `updateID` values 109-110 are left over from time T6.

GENA behavior:

Event the initial **Notify** message for the current value of the *LastChange* state variable.

2.6.15.1.2 Monitoring Subtree Updates

The following example shows the type of information that a control point can receive when a ContentDirectory service implementation updates an entire subtree. Control points that wish to track subtree updates can use the ContentDirectory service's *LastChange* state variable to receive event messages that indicate which objects within the subtree have been updated. Once a control point has subscribed to events (using the normal UPnP event subscription mechanism), updates to the *LastChange* state variable are evented to the control point. The *LastChange* state variable will identify the ContentDirectory objects that have been modified since the end of the previous moderation period. Those objects that are part of a subtree update will have an `stUpdate` attribute value of one ("1").

The following example shows the value of the *LastChange* state variable when an existing subtree (*@id* = "Album001") is updated as follows:

- A new object (*@id* = "Song003") is created as a child of the root container of the subtree (*@id* = "Album001").
- An existing (descendant) object (*@id* = "Song001") is modified.

- An existing (descendant) object (*@id* = “Song002”) is deleted.

Note: This example assumes that all changes occur within the same moderation period. Otherwise, the <objAdd>, <objMod>, <objDel>, and <stDone> elements could be separated into different event messages. See subclause 2.4, “Eventing and Moderation” for details. Also, events from individual objects that are not part of the subtree update may be interleaved with the events that do belong to the subtree update. However, this example does not illustrate the mixing of individual and subtree events.

LastChange (before XML escaping):

```
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
 xmlns="urn:schemas-upnp-org:av:cds-event"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:cds-event
   http://www.upnp.org/schemas/av/cds-event.xsd">
   <objAdd objID="Song003" updateID="234" objParentId="Album001"
    objClass="object.item.audioItem" stUpdate="1"/>
   <objMod objID="Album001" updateID="235" stUpdate="1"/>
   <objMod objID="Song001" updateID="236" stUpdate="1"/>
   <objDel objID="Song002" updateID="237" stUpdate="1"/>
   <objMod objID="Album001" updateID="238" stUpdate="1"/>
   <stDone objID="Album001" updateID="238"/>
</StateEvent>
```

Note: Since the <stDone> event does not reflect a change in the ContentDirectory service its updateID attribute value will not be unique and simply reflects the current value of the *SystemUpdateID* state variable when the operation finished.

In this example, control points that receive the above event message can process the event message a number of different ways. However, the following two options are provided to help illustrate the flexibility that a control point has in choosing an event processing strategy that best suites the control point’s internal design and any constraints that may exist when event message is received.

- Option 1: A control point may process the individual events as if the stUpdate attribute was set to “0” in which case, the control point would simply ignore the <stDone> event.
- Option 2: A control point may ignore all of the events whose stUpdate attribute was set to “1” and when the <stDone> event is received, the control point could then process the entire subtree starting with the container “Album001” working its way down through the subtree.

2.6.15.2 Monitoring Changes while *off-line*

The following examples show how a control point can detect changes to certain objects that occur while a ContentDirectory service implementation or the control point is *off-line*. Obviously, these changes can only be processed when both the ContentDirectory service implementation and the control point are simultaneously *on-line*. Although these examples focus on a control point that disconnects itself from the network, the same algorithms can be used by a control point that remains *on-line* and wants to detect changes that have occurred during periods while a ContentDirectory service implementation was *off-line*.

This procedure applies only to those objects that expose the *upnp:objectUpdateID* or *upnp:containerUpdateID* properties.

Control points that frequently disconnect from or reconnect themselves to the network (for example, a cell phone capable of controlling a home stereo), may want to determine the changes that were made to one or more ContentDirectory service objects while the control point was *off-line*. Prior to going *off-line*, the control point needs to retrieve and persist the current value of the *SytemUpdateID* state variable for each ContentDirectory service it wishes

to examine when it reconnects. Additionally, it needs to store some state information about each of the objects that it is monitoring for changes.

In all of these examples, it is assumed that the control point maintains a local cache of information about objects that it is monitoring on the ContentDirectory service implementation. It may choose to update this cache just prior to going *off-line* and update the cache again immediately upon reconnecting to the network. Alternatively, it may choose to update its cache at arbitrary times across multiple periods when the control point and/or the ContentDirectory service implementation have gone *off-line*.

Note: The detection algorithms described below are only one possible option. There may be other algorithms that can be used that provide various efficiencies.

2.6.15.2.1 Detecting Modified Objects

A control point can determine which objects were modified since it last updated its local object information cache by searching the ContentDirectory service (via the *Search()* action on the root container) for any object whose *upnp:objectUpdateID* property is greater than the value of the ContentDirectory service's *SystemUpdateID* state variable that was saved by the control point at the time that it updated its cached information. Any object that was updated since the control point updated its local information cache will have an *upnp:objectUpdateID* property value that is greater than the *SystemUpdateID* state variable value that was saved by the control point. See subclause 2.2.5, "Object Modification" and Annex B.15.2, "*upnp:objectUpdateID*" for details. Note that the *upnp:objectUpdateID* property is optional for an object. If the ContentDirectory service implementation supports tracking changes for that object it will expose that property on the object. A control point cannot expect an object to be returned in the above *Search()* operation if that object does not support the *upnp:objectUpdateID* property. The value of the *SystemUpdateID* state variable may have been incremented since the last time that the control point updated its local information cache. However, the control point may receive an empty result from the above *Search()* action. This implies that the changes that occurred were on objects that did not expose the *upnp:objectUpdateID* or *upnp:containerUpdateID* properties.

In some situations, a control point may only be interested in determining which containers have experienced a container modification. One way of accomplishing this is for the control point to invoke the *Search()* action as described above. However, the control point can search for any container whose *upnp:containerUpdateID* property is greater than the value of the ContentDirectory service's *SystemUpdateID* state variable that was saved by the control point at the time that it updated its cached information. See subclause 2.5.8, "*Search()*" for details.

Example:

- A *searchCriteria* argument value of "`objectUpdateID > 235`" will return all objects in the queried CDS subtree that are supporting tracking changes and that were modified after the *SystemUpdateID* state variable was set to 235.
- A *searchCriteria* argument value of "`containerUpdateID > 235`" will return all of the containers in the queried CDS subtree that are supporting tracking changes and that have experienced a container modification since the *SystemUpdateID* state variable was set to 235.

2.6.15.2.1.1 Determining Which Properties were Modified

A control point can determine the exact set of properties that were modified since the last time that it updated its local information cache as follows. The control point must store a copy of the property values of each object that it is monitoring. When the control point updates its cache, it first needs to determine which objects were modified (as described above in subclause 2.6.15.2.1, "Detecting Modified Objects". Then, for each modified object, the control point retrieves the object's current property values and compares them with the property values that were saved. The property values that do not match indicate that the property was modified.

2.6.15.2.1.2 Determining Which Properties were Added

A control point can determine which properties, if any, were added to an object since the last time that it updated its local information cache as follows. The control point needs to follow the procedure for determining which properties were modified (subclause 2.6.15.2.1.1, “Determining Which Properties were Modified”). However, rather than comparing individual property values, the control point simply needs to identify those properties that currently exist in a given object but did not exist when the object’s properties were saved. Those “new” properties were added.

2.6.15.2.1.3 Determining Which Properties were Deleted

A control point can determine which properties, if any, were deleted from an object since the last time that it updated its local information cache as follows. The control point needs to follow the procedure for determining which properties were modified (subclause 2.6.15.2.1.1, “Determining Which Properties were Modified”). However, rather than comparing individual property values, the control point simply needs to identify those properties that existed when the object’s properties were saved but no longer exist in the object’s current set of properties. Those properties were deleted.

2.6.15.2.2 Detecting Added Objects

A control point can determine which objects were added to the ContentDirectory service since the last time that it updated its local information cache as follows. The control point follows the procedure for determining which objects were modified (subclause 2.6.15.2.1, “Detecting Modified Objects”). It then can compare the list of objects that was returned by the *Search()* action with the objects that exist in its local information cache. If an object is returned by the *Search()* action but does not exist in the local information cache, that object was added. The results of the *Search()* action may contain objects that the control point is not monitoring and the control point must differentiate between new objects within containers that it is monitoring and modifications to or additions of objects that it is not monitoring. Either of these situations will result in an object returned by the *Search()* action that does not exist within the information stored by the control point. The control point can then further decide whether it is an object that it wants to monitor based on certain criteria, such as the object’s *@parentID* property value.

As an alternative, a control point can determine which objects were added using a slightly different procedure. The control point follows the procedure for detecting modified properties as described above (subclause 2.6.15.2.1.1, “Determining Which Properties were Modified”). From those results, the control point identifies any containers whose *@childCount* and/or *upnp:totalDeletedChildCount* properties have changed. For each container where the sum of the *@childCount* and *upnp:totalDeletedChildCount* properties has changed, the control point compares the current list of child objects for that container with that container’s list of child objects that is saved in the local information cache. Any child object, within a container that the control point is monitoring, that currently exists but does not exist in the control point’s local information cache was added to that container.

2.6.15.2.3 Detecting Deleted Objects

When reconnecting to the network, a control point can determine which objects were deleted from the ContentDirectory service since the last time that the control point updated its local cache of information. This process is a little more complicated than the other detection procedures because the ContentDirectory service does not preserve any information about the deleted object since the deleted object is no longer accessible.

To begin, a control point follows the procedure for determining which properties were modified (subclause 2.6.15.2.1.1, “Determining Which Properties were Modified”). However, instead of examining all the properties of each modified object, the control point only needs to examine and compare the *upnp:totalDeletedChildCount* property of each modified container object. If the modified container’s *upnp:totalDeletedChildCount* property is greater than its stored value, then one or more objects were deleted from that container.

In order to determine which objects were deleted, the control point needs to compare the container's current list of child objects (obtained via the *Browse()* action) with the list of child objects that the control point has stored locally. If an object exists in the local storage but no longer exists as a child object of that container in the ContentDirectory service implementation, then that object was deleted from that container since the last time that the control point updated its local cache of information.

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>GetSortExtensionCapabilities</name>
         <argumentList>
            <argument>
               <name>SortExtensionCaps</name>
               <direction>out</direction>
               <relatedStateVariable>
                  SortExtensionCapabilities
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetFeatureList</name>
         <argumentList>
            <argument>
               <name>FeatureList</name>
               <direction>out</direction>
               <relatedStateVariable>
                  FeatureList
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetSystemUpdateID</name>
         <argumentList>
            <argument>
```

```
            <name>Id</name>
            <direction>out</direction>
            <relatedStateVariable>
               SystemUpdateID
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>
   <action>
      <name>GetServiceResetToken</name>
      <argumentList>
         <argument>
            <name>ResetToken</name>
            <direction>out</direction>
            <relatedStateVariable>
               ServiceResetToken
            </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>MoveObject</name>
    <argumentList>
        <argument>
            <name>ObjectID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_ObjectID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>NewParentID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_ObjectID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>NewObjectID</name>
            <direction>out</direction>
            <relatedStateVariable>
                A_ARG_TYPE_ObjectID
            </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>DeleteResource</name>
      <argumentList>
         <argument>
            <name>ResourceURI</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_URI
            </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>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>
<action>
    <name>FreeFormQuery</name>
    <argumentList>
        <argument>
            <name>ContainerID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_ObjectID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>CDSView</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_CDSView
            </relatedStateVariable>
        </argument>
        <argument>
```

```
                <name>QueryRequest</name>
                <direction>in</direction>
                <relatedStateVariable>
                   A_ARG_TYPE_QueryRequest
                </relatedStateVariable>
             </argument>
             <argument>
                <name>QueryResult</name>
                <direction>out</direction>
                <relatedStateVariable>
                   A_ARG_TYPE_QueryResult
                </relatedStateVariable>
             </argument>
             <argument>
                <name>UpdateID</name>
                <direction>out</direction>
                <relatedStateVariable>
                   A_ARG_TYPE_UpdateID
                </relatedStateVariable>
             </argument>
          </argumentList>
       </action>
       <action>
          <name>GetFreeFormQueryCapabilities</name>
          <argumentList>
             <argument>
                <name>FFQCapabilities</name>
                <direction>out</direction>
                <relatedStateVariable>
                   A_ARG_TYPE_FFQCapabilities
                </relatedStateVariable>
             </argument>
          </argumentList>
       </action>
       Declarations for other actions added by UPnP vendor
       (if any) go here
    </actionList>
    <serviceStateTable>
       <stateVariable sendEvents="no">
          <name>SearchCapabilities</name>
          <dataType>string</dataType>
       </stateVariable>
       <stateVariable sendEvents="no">
          <name>SortCapabilities</name>
          <dataType>string</dataType>
       </stateVariable>
       <stateVariable sendEvents="no">
          <name>SortExtensionCapabilities</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>
       <stateVariable sendEvents="no">
          <name>ServiceResetToken</name>
          <dataType>string</dataType>
       </stateVariable>
       <stateVariable sendEvents="yes">
          <name>LastChange</name>
          <dataType>string</dataType>
       </stateVariable>
       <stateVariable sendEvents="yes">
          <name>TransferIDs</name>
          <dataType>string</dataType>
       </stateVariable>
```

```
<stateVariable sendEvents="no">
   <name>FeatureList</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>A_ARG_TYPE_CDSView</name>
           <dataType>ui4</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
           <name>A_ARG_TYPE_QueryRequest</name>
           <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
           <name>A_ARG_TYPE_QueryResult</name>
           <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
           <name>A_ARG_TYPE_FFQCapabilities</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.

Annex A
(normative)

Schemas

This annex describes the XML schemas for the DIDL-Lite element set. The UPnP, Dublin Core and XML namespaces are imported into the DIDL-Lite schema.

A.1 DIDL-Lite

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

The referenced DIDL-Lite schema [DIDL-LITE-XSD] may be downloaded from the UPnP Forum website and saved into a local 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 ContentDirectory services will employ schema-based validation in the implementation of A/V functionality. The schema serves as a reference for the format of *DIDL-Lite XML Documents* and *DIDL-Lite XML Fragments*. Any discrepancies between this specification and the schema MUST be resolved in favor of the specification.

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 ContentDirectory services (see subclause 4, “Test.”)

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

A.2 UPnP Elements

The referenced schema [UPNP-XSD] defines the *upnp* properties that are implemented as XML elements and attributes and used in DIDL-Lite. The schema may be downloaded from the UPnP Forum website and saved into a local file for use in a validating parser or instance document-editing tool.

A.3 Dublin Core Subset Elements

The referenced schema [DC-XSD] defines the *dc* namespace tags that are employed as descriptors under DIDL-Lite. They represent a subset of Dublin Core elements.

A.4 Event Schema

The XML schema [CDS-EVENT-XSD] describes the format of the *LastChange* state variable which is used to indicate that one or more ContentDirectory objects has changed. For more details see subclause 2.3.8 “*LastChange*”.

A.5 *FeatureList* State Variable Schema

The external XML schema [AVS-XSD] describes the format of the *FeatureList* state variable, which is used to indicate supported *CDS features* defined in Annex E, “ **(normative)**

CDS features”.

Annex B
(normative)

AV Working Committee Properties

The tables and clauses below list all properties of ContentDirectory service objects as defined by the AV Working Committee.

ContentDirectory service object descriptions are serialized into *DIDL-Lite XML Documents* in response to *Browse()* and *Search()* requests. *DIDL-Lite XML Documents* are formatted according to the DIDL-Lite schema in [DIDL-LITE-XSD]. The DIDL-Lite schema includes elements from the upnp schema [UPNP-XSD] and a subset of the Dublin Core schema [DC-XSD].

The tables and clauses below describe each object property that can appear in serialized form in a *DIDL-Lite XML Document*, as well as the XML data type [XML SCHEMA-2] from which each property is derived. Properties that are directly based on XML datatypes are listed with the xsd: prefix.

Note: The NS column in the tables contains the namespace prefix of the namespace to which the property name belongs. The M-Val column indicates whether the property is multi-valued (M-Val = *YES*) or single-valued (M-Val = *NO*). See subclause 2.2.20.1, “Multi-valued property” and subclause 2.2.20.2, “Single-valued property”. The R/W column indicates whether the property may be modified by a control point using ContentDirectory actions such as UpdateObject(). A property may be marked “*R*” to indicate that the property is “read-only”, “*R/W*” to indicate that the property is “read-write”, or “*V*” to indicate that the read/write characteristics of the property are determined by the ContentDirectory service implementation.

In the description clauses below, each property is either marked as *read-only*, *read-write*, or not marked indicating that the read/write character is defined by the ContentDirectory service implementation.

The following table presents an overview of all ContentDirectory service defined properties.

Table B.1 — ContentDirectory Service Properties Overview

#	Property Name	NS	Data Type	M-Val	R/W	Reference
1.	*@id*	DIDL-Lite	xsd:string	*NO*	*R*	Subclause B.1.1
2.	*@parentID*	DIDL-Lite	xsd:string	*NO*	*R*	Subclause B.1.2
3.	*@refID*	DIDL-Lite	xsd:string	*NO*	*R*	Subclause B.1.3
4.	*@restricted*	DIDL-Lite	xsd:boolean	*NO*	*R*	Subclause B.1.4
5.	*@searchable*	DIDL-Lite	xsd:boolean	*NO*	*R*	Subclause B.1.5
6.	*@childCount*	DIDL-Lite	xsd:unsignedInt	*NO*	*R*	Subclause B.1.6
7.	*dc:title*	dc	xsd:string	*NO*	*V*	Subclause B.1.7
8.	*dc:creator*	dc	xsd:string	*NO*	*V*	Subclause B.1.8
9.	*res*	DIDL-Lite	xsd:anyURI	*YES*	*V*	Subclause B.1.9
10.	*upnp:class*	upnp	xsd:string	*NO*	*V*	Subclause B.1.10
11.	*upnp:class@name*	upnp	xsd:string	*NO*	*V*	Subclause B.1.10.1
12.	*upnp:searchClass*	upnp	xsd:string	*YES*	*R*	Subclause B.1.11
13.	*upnp:searchClass@name*	upnp	xsd:string	*NO*	*R*	Subclause B.1.11.1
14.	*upnp:searchClass@includeDerive*	upnp	xsd:boolean	*NO*	*R*	Subclause B.1.11.2

#	Property Name	NS	Data Type	M-Val	R/W	Reference
	d					
15.	upnp:createClass	upnp	xsd:string	YES	R	Subclause B.1.12
16.	upnp:createClass@name	upnp	xsd:string	NO	R	Subclause B.1.12.1
17.	upnp:createClass@includeDerived	upnp	xsd:boolean	NO	R	Subclause B.1.12.2
18.	upnp:writeStatus	upnp	xsd:string	NO	R	Subclause B.1.13
19.	res@protocolInfo	DIDL-Lite	xsd:string	NO	V	Subclause B.2.1.1
20.	res@importUri	DIDL-Lite	xsd:anyURI	NO	R	Subclause B.2.1.2
21.	res@size	DIDL-Lite	xsd:unsignedLong	NO	V	Subclause B.2.1.3
22.	res@duration	DIDL-Lite	xsd:string	NO	V	Subclause B.2.1.4
23.	res@protection	DIDL-Lite	xsd:string	NO	V	Subclause B.2.1.5
24.	res@bitrate	DIDL-Lite	xsd:unsignedInt	NO	V	Subclause B.2.1.6
25.	res@bitsPerSample	DIDL-Lite	xsd:unsignedInt	NO	V	Subclause B.2.1.7
26.	res@sampleFrequency	DIDL-Lite	xsd:unsignedInt	NO	V	Subclause B.2.1.8
27.	res@nrAudioChannels	DIDL-Lite	xsd:unsignedInt	NO	V	Subclause B.2.1.9
28.	res@resolution	DIDL-Lite	xsd:string	NO	V	Subclause B.2.1.10
29.	res@colorDepth	DIDL-Lite	xsd:unsignedInt	NO	V	Subclause B.2.1.11
30.	res@tspec	DIDL-Lite	xsd:string	NO	V	Subclause B.2.1.12
31.	res@allowedUse	DIDL-Lite	CSV (xsd:string)	NO	V	Subclause B.2.1.13
32.	res@validityStart	DIDL-Lite	xsd:string	NO	V	Subclause B.2.1.14
33.	res@validityEnd	DIDL-Lite	xsd:string	NO	V	Subclause B.2.1.15
34.	res@remainingTime	DIDL-Lite	xsd:string	NO	V	Subclause B.2.1.16
35.	res@usageInfo	DIDL-Lite	xsd:string	NO	V	Subclause B.2.1.17
36.	res@rightsInfoURI	DIDL-Lite	xsd:anyURI	NO	V	Subclause B.2.1.18
37.	res@contentInfoURI	DIDL-Lite	xsd:anyURI	NO	V	Subclause B.2.1.19
38.	res@recordQuality	DIDL-Lite	CSV (xsd:string)	NO	V	Subclause B.2.1.20
39.	res@daylightSaving	DIDL-Lite	xsd:string	NO	V	Subclause B.2.1.21
40.	upnp:artist	upnp	xsd:string	YES	V	Subclause B.3.1
41.	upnp:artist@role	upnp	xsd:string	NO	V	Subclause B.3.1.1
42.	upnp:actor	upnp	xsd:string	YES	V	Subclause B.3.2
43.	upnp:actor@role	upnp	xsd:string	NO	V	Subclause B.3.2.1
44.	upnp:author	upnp	xsd:string	YES	V	Subclause B.3.3
45.	upnp:author@role	upnp	xsd:string	NO	V	Subclause B.3.3.1
46.	upnp:producer	upnp	xsd:string	YES	V	Subclause B.3.4
47.	upnp:director	upnp	xsd:string	YES	V	Subclause B.3.5
48.	dc:publisher	dc	xsd:string	YES	V	Subclause B.3.6
49.	dc:contributor	dc	xsd:string	YES	V	Subclause B.3.7
50.	upnp:genre	upnp	xsd:string	YES	V	Subclause B.4.1
51.	upnp:genre@id	upnp	xsd:string	NO	V	Subclause B.4.1.1
52.	upnp:genre@extended	upnp	CSV (xsd:string)	NO	V	Subclause B.4.1.2
53.	upnp:album	upnp	xsd:string	YES	V	Subclause B.4.2
54.	upnp:playlist	upnp	xsd:string	YES	V	Subclause B.4.3
55.	upnp:albumArtURI	upnp	xsd:anyURI	YES	V	Subclause B.5.1

#	Property Name	NS	Data Type	M-Val	R/W	Reference
56.	upnp:artistDiscographyURI	upnp	xsd:anyURI	NO	V	Subclause B.5.2
57.	upnp:lyricsURI	upnp	xsd:anyURI	NO	V	Subclause B.5.3
58.	dc:relation	dc	xsd:string	YES	V	Subclause B.5.4
59.	upnp:storageTotal	upnp	xsd:long	NO	R	Subclause B.6.1
60.	upnp:storageUsed	upnp	xsd:long	NO	R	Subclause B.6.2
61.	upnp:storageFree	upnp	xsd:long	NO	R	Subclause B.6.3
62.	upnp:storageMaxPartition	upnp	xsd:long	NO	R	Subclause B.6.4
63.	upnp:storageMedium	upnp	xsd:string	NO	R	Subclause B.6.5
64.	dc:description	dc	xsd:string	NO	V	Subclause B.7.1
65.	upnp:longDescription	upnp	xsd:string	NO	V	Subclause B.7.2
66.	upnp:icon	upnp	xsd:anyURI	NO	V	Subclause B.7.3
67.	upnp:region	upnp	xsd:string	NO	V	Subclause B.7.4
68.	dc:rights	dc	xsd:string	YES	V	Subclause B.7.5
69.	dc:date	dc	xsd:string	NO	V	Subclause B.7.6
70.	dc:date@upnp:daylightSaving	upnp	xsd:string	NO	V	Subclause B.7.6.1
71.	dc:language	dc	xsd:string	YES	V	Subclause B.7.7
72.	upnp:playbackCount	upnp	xsd:int	NO	R	Subclause B.7.8
73.	upnp:lastPlaybackTime	upnp	xsd:string	NO	V	Subclause B.7.9
74.	upnp:lastPlaybackTime @daylightSaving	upnp	xsd:string	NO	V	Subclause B.7.9.1
75.	upnp:lastPlaybackPosition	upnp	xsd:string	NO	V	Subclause B.7.10
76.	upnp:recordedStartDateTime	upnp	xsd:string	NO	V	Subclause B.7.11
77.	upnp:recordedStartDateTime @daylightSaving	upnp	xsd:string	NO	V	Subclause B.7.11.1
78.	upnp:recordedDuration	upnp	xsd:string	NO	V	Subclause B.7.12
79.	upnp:recordedDayOfWeek	upnp	xsd:string	NO	V	Subclause B.7.13
80.	upnp:srsRecordScheduleID	upnp	xsd:string	NO	R	Subclause B.7.14
81.	upnp:srsRecordTaskID	upnp	xsd:string	NO	R	Subclause B.7.15
82.	upnp:recordable	upnp	xsd:boolean	NO	V	Subclause B.7.16
83.	upnp:programTitle	upnp	xsd:string	NO	V	Subclause B.8.1
84.	upnp:seriesTitle	upnp	xsd:string	NO	V	Subclause B.8.2
85.	upnp:programID	upnp	xsd:string	NO	V	Subclause B.8.3
86.	upnp:programID@type	upnp	xsd:string	NO	V	Subclause B.8.3.1
87.	upnp:seriesID	upnp	xsd:string	NO	V	Subclause B.8.4
88.	upnp:seriesID@type	upnp	xsd:string	NO	V	Subclause B.8.4.1
89.	upnp:channelID	upnp	xsd:string	YES	V	Subclause B.8.5
90.	upnp:channelID@type	upnp	xsd:string	NO	V	Subclause B.8.5.1
91.	upnp:channelID@distriNetworkName	upnp	xsd:string	NO	V	Subclause B.8.5.2
92.	upnp:channelID@distriNetworkID	upnp	xsd:string	NO	V	Subclause B.8.5.3
93.	upnp:episodeCount	upnp	xsd:unsignedInt	NO	V	Subclause B.8.5.2
94.	upnp:episodeNumber	upnp	xsd:unsignedInt	NO	V	Subclause B.8.7
95.	upnp:programCode	upnp	xsd:string	NO	V	Subclause B.8.8
96.	upnp:programCode@type	upnp	xsd:string	NO	V	Subclause B.8.8.1

#	Property Name	NS	Data Type	M-Val	R/W	Reference
97.	upnp:rating	upnp	xsd:string	NO	V	Subclause B.8.9
98.	upnp:rating@type	upnp	xsd:string	NO	V	Subclause B.8.9.1
99.	upnp:episodeType	upnp	xsd:string	NO	V	Subclause B.8.10
100.	upnp:channelGroupName	upnp	xsd:string	NO	V	Subclause B.9.1
101.	upnp:channelGroupName@id	upnp	xsd:string	NO	V	Subclause B.9.1.1
102.	upnp:callSign	upnp	xsd:string	NO	V	Subclause B.9.2
103.	upnp:networkAffiliation	upnp	xsd:string	NO	V	Subclause B.9.3
104.	upnp:serviceProvider	upnp	xsd:string	NO	V	Subclause B.9.4
105.	upnp:price	upnp	xsd:float	YES	R	Subclause B.9.5
106.	upnp:price@currency	upnp	xsd:string	NO	R	Subclause B.9.5.1
107.	upnp:payPerView	upnp	xsd:boolean	NO	R	Subclause B.9.6
108.	upnp:epgProviderName	upnp	xsd:string	NO	V	Subclause B.9.7
109.	upnp:dateTimeRange	upnp	xsd:string	NO	R	Subclause B.9.8
110.	upnp:dateTimeRange@daylightSaving	upnp	xsd:string	NO	R	Subclause B.9.8.1
111.	upnp:radioCallSign	upnp	xsd:string	NO	V	Subclause B.10.1
112.	upnp:radioStationID	upnp	xsd:string	NO	V	Subclause B.10.2
113.	upnp:radioBand	upnp	xsd:string	NO	V	Subclause B.10.3
114.	upnp:channelNr	upnp	xsd:int	NO	V	Subclause B.11.1
115.	upnp:channelName	upnp	xsd:string	NO	V	Subclause B.11.2
116.	upnp:scheduledStartTime	upnp	xsd:string	NO	V	Subclause B.11.3
117.	upnp:scheduledStartTime@usage	upnp	xsd:string	NO	V	Subclause B.11.3.1
118.	upnp:scheduledStartTime@daylightSaving	upnp	xsd:string	NO	V	Subclause B.11.3.2
119.	upnp:scheduledEndTime	upnp	xsd:string	NO	V	Subclause B.11.4
120.	upnp:scheduledEndTime@daylightSaving	upnp	xsd:string	NO	V	Subclause B.11.4.1
121.	upnp:scheduledDuration	upnp	xsd:string	NO	V	Subclause B.11.5
122.	upnp:signalStrength	upnp	xsd:int	NO	R	Subclause B.12.1
123.	upnp:signalLocked	upnp	xsd:boolean	NO	R	Subclause B.12.2
124.	upnp:tuned	upnp	xsd:boolean	NO	R	Subclause B.12.3
125.	@neverPlayable	DIDL-Lite	xsd:boolean	NO	V	Subclause B.13.1
126.	upnp:bookmarkID	upnp	xsd:string	YES	R/W	Subclause B.13.2
127.	upnp:bookmarkedObjectID	upnp	xsd:string	NO	R/W	Subclause B.13.3
128.	upnp:deviceUDN	upnp	xsd:string	NO	R/W	Subclause B.13.4
129.	upnp:deviceUDN@serviceType	upnp	xsd:string	NO	R/W	Subclause B.13.4.1
130.	upnp:deviceUDN@serviceId	upnp	xsd:string	NO	R/W	Subclause B.13.4.2
131.	upnp:stateVariableCollection	upnp	xsd:string	YES	R/W	Subclause B.13.5
132.	upnp:stateVariableCollection@serviceName	upnp	xsd:string	NO	R/W	Subclause B.13.5.1

#	Property Name	NS	Data Type	M-Val	R/W	Reference
133.	upnp:stateVariableCollection@rcsInstanceType	upnp	xsd:string	NO	R/W	Subclause B.13.5.2
134.	upnp:DVDRegionCode	upnp	xsd:int	NO	V	Subclause B.14.1
135.	upnp:originalTrackNumber	upnp	xsd:int	NO	V	Subclause B.14.2
136.	upnp:toc	upnp	xsd:string	NO	V	Subclause B.14.3
137.	upnp:userAnnotation	upnp	xsd:string	YES	R/W	Subclause B.14.4
138.	desc	DIDL-Lite	xsd:string	YES	V	Subclause B.14.5
139.	desc@nameSpace	DIDL-Lite	xsd:string	NO	V	Subclause B.14.5.1
140.	upnp:containerUpdateID	upnp	xsd:unsignedInt	NO	R	Subclause B.15.1
141.	upnp:objectUpdateID	upnp	xsd:unsignedInt	NO	R	Subclause B.15.2
142.	upnp:totalDeletedChildCount	upnp	xsd:unsignedInt	NO	R	Subclause B.15.3
143.	res@updateCount	DIDL-Lite	xsd:unsignedInt	NO	R	Subclause B.15.4
144.	upnp:foreignMetadata	upnp	<XML>	YES	V	Subclause B.16.1
145.	upnp:foreignMetadata@type	upnp	xsd:string	NO	V	Subclause B.16.1.1
146.	upnp:foreignMetadata::fmId	upnp	xsd:string	YES	V	Subclause B.16.1.2
147.	upnp:foreignMetadata::fmClass	upnp	xsd:string	YES	V	Subclause B.16.1.3
148.	upnp:foreignMetadata::fmProvider	upnp	xsd:string	NO	V	Subclause B.16.1.4
149.	upnp:foreignMetadata::fmBody	upnp	<XML>	NO	V	Subclause B.16.1.5
150.	upnp:foreignMetadata::fmBody@xmlFlag	upnp	xsd:boolean	NO	V	Subclause B.16.1.5.1
151.	upnp:foreignMetadata::fmBody@mimeType	upnp	xsd:string	NO	V	Subclause B.16.1.5.2
152.	upnp:foreignMetadata::fmBody::fmEmbeddedXML	upnp	<XML>	NO	V	Subclause B.16.1.5.3
153.	upnp:foreignMetadata::fmBody::fmEmbeddedString	upnp	xsd:string	NO	V	Subclause B.16.1.5.4
154.	upnp:foreignMetadata::fmBody::fmURI	upnp	xsd:anyURI	NO	V	Subclause B.16.1.5.5

B.1 Base Properties

Table B.2 — Base Properties Overview

Property Name	NS	Data Type	M-Val	Reference
@id	DIDL-Lite	xsd:string	NO	Subclause B.1.1
@parentID	DIDL-Lite	xsd:string	NO	Subclause B.1.2
@refID	DIDL-Lite	xsd:string	NO	Subclause B.1.3
@restricted	DIDL-Lite	xsd:boolean	NO	Subclause B.1.4
@searchable	DIDL-Lite	xsd:boolean	NO	Subclause B.1.5
@childCount	DIDL-Lite	xsd:unsignedInt	NO	Subclause B.1.6
dc:title	dc	xsd:string	NO	Subclause B.1.7
dc:creator	dc	xsd:string	NO	Subclause B.1.8
res	DIDL-Lite	xsd:anyURI	YES	Subclause B.1.9
upnp:class	upnp	xsd:string	NO	Subclause B.1.10

Property Name	NS	Data Type	M-Val	Reference
upnp:class@name	upnp	xsd:string	NO	Subclause B.1.10.1
upnp:searchClass	upnp	xsd:string	YES	Subclause B.1.11
upnp:searchClass@name	upnp	xsd:string	NO	Subclause B.1.11.1
upnp:searchClass@includeDerived	upnp	xsd:boolean	NO	Subclause B.1.11.2
upnp:createClass	upnp	xsd:string	YES	Subclause B.1.12
upnp:createClass@name	upnp	xsd:string	NO	Subclause B.1.12.1
upnp:createClass@includeDerived	upnp	xsd:boolean	NO	Subclause B.1.12.2
upnp:writeStatus	upnp	xsd:string	NO	Subclause B.1.13

B.1.1 *@id*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** NO

Description: The *read-only* @id property is a REQUIRED property that MUST provide a unique identity for the object with respect to all of the objects within the ContentDirectory service.

For all objects that support tracking of changes (i.e. those that expose the upnp:objectUpdateID or upnp:containerUpdateID properties), as long as the ServiceResetToken remains constant, the ContentDirectory service MUST ensure the persistence of these object's @id property values. If the ContentDirectory service cannot ensure the persistence of these object's @id property values, then it MUST invoke the *Service Reset Procedure*. See subclause 2.3.7, "ServiceResetToken" and subclause 2.3.7.1, "Service Reset Procedure" for details.

For all objects, regardless of whether they support tracking of changes or not, as long as the ServiceResetToken remains constant, the ContentDirectory service MUST ensure the object ID's uniqueness; that is: if an object is created with the same @id property as a previously deleted object, the service is making the claim that these two objects are the same. If the ContentDirectory service cannot ensure the uniqueness of an object's @id property value, then it MUST invoke the *Service Reset procedure*. See subclause 2.3.7, "ServiceResetToken" and subclause 2.3.7.1, "Service Reset Procedure" for details.

For all objects that do not support tracking of changes, as long as the ServiceResetToken remains constant, the ContentDirectory service is RECOMMENDED to ensure the persistence of these object's @id property values. If the ContentDirectory service cannot ensure the persistence of these object's @id property values, then it SHOULD invoke the *Service Reset Procedure*. See subclause 2.3.7, "ServiceResetToken" and subclause 2.3.7.1, "Service Reset Procedure" for details.

B.1.2 *@parentID*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** NO

Description: The *read-only* @parentID property is a REQUIRED property of an item or container object. The @parentID property MUST be set and always remain equal to the @id property of the object's parent, which MUST be a container. The @parentID property of the ContentDirectory service root container MUST be set to the reserved value of -1. The @parentID property of any other ContentDirectory service object MUST NOT take this value.

Default Value: N/A – The property is REQUIRED.

B.1.3 *@refID*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** NO

Description: The *read-only* *@refID* property is only applicable to item objects. The presence of this property indicates that the item is actually referencing another existing item (*reference item*). The *@refID* property MUST be set and always remain equal to the *@id* property of the item that is referenced.

Default Value: None.

B.1.4 *@restricted*

Namespace: DIDL-Lite **Property Data Type:** xsd:boolean **Multi-Valued:** *NO*

Description: The *read-only* REQUIRED *@restricted* property indicates whether the object is modifiable. If set to "*1*", the ability to modify or delete a given object is confined to the ContentDirectory service implementation. Therefore, a control point cannot add, modify or delete metadata from a restricted object. Additionally, control points are not able to add, modify or delete any children of a restricted container. However, the *@restricted* property does not propagate to descendant objects. Note however, that metadata of a restricted object may still change due to internal ContentDirectory service implementation manipulations.

If set to "*0*", a control point can modify the object's metadata and add, delete, or modify the object's children.

Default Value: N/A – The property is REQUIRED.

B.1.5 *@searchable*

Namespace: DIDL-Lite **Property Data Type:** xsd:boolean **Multi-Valued:** *NO*

Description: The *read-only* *@searchable* property is only applicable to container objects. When "*1*" (true), the ability to perform a *Search()* action under a container is enabled, otherwise a *Search()* action under that container will return no results, even when child containers have their *@searchable* property set to "*1*".

Default Value: "*0*".

B.1.6 *@childCount*

Namespace: DIDL-Lite **Property Data Type:** xsd:unsignedInt **Multi-Valued:** *NO*

Description: The *read-only* *@childCount* property is only applicable to container objects. It reflects the number of direct children contained in the container object.

Default Value: None.

B.1.7 *dc:title*

Namespace: dc **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *dc:title* property is a REQUIRED property and indicates a friendly name for the object. See http://dublincore.org/documents/dces.

Default Value: N/A – The property is REQUIRED.

B.1.8 *dc:creator*

Namespace: dc **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *dc:creator* property indicates an entity that owns the content or is primarily responsible for creating the content. Examples include a person, an organization or a service. Typically, the name of the creator should be used to indicate the entity. See http://dublincore.org/documents/dces.

Default Value: None.

B.1.9 *res*

Namespace: DIDL-Lite **Property Data Type:** xsd:anyURI **Multi-Valued:** *YES*

Description: The *res* property indicates a resource, typically a media file, associated with the object. If the value of the *res* property is not present, then the content has not yet been fully imported by the ContentDirectory service and is not yet accessible for playback purposes. Values MUST be properly escaped URIs as described in [RFC 2396].

Default Value: None.

B.1.10 *upnp:class*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:class* property is a REQUIRED property and it indicates the class of the object.

Default Value: N/A – The property is REQUIRED.

B.1.10.1 allowedValueList for the *upnp:class* Property

Table B.3 — allowedValueList for the *upnp:class* Property

Value	R/O	Description
"*object.item*"	*O*	See Subclause C.2.1
"*object.item.imageItem*"	*O*	See Subclause C.2.1.1
"*object.item.imageItem.photo*"	*O*	See Subclause C.2.1.1.1
"*object.item.audioItem*"	*O*	See Subclause C.2.1.2
"*object.item.audioItem.musicTrack*"	*O*	See Subclause C.2.1.2.1
"*object.item.audioItem.audioBroadcast*"	*O*	See Subclause C.2.1.2.2
"*object.item.audioItem.audioBook*"	*O*	See Subclause C.2.1.2.3
"*object.item.videoItem*"	*O*	See Subclause C.2.1.3
"*object.item.videoItem.movie*"	*O*	See Subclause C.2.1.3.1
"*object.item.videoItem.videoBroadcast*"	*O*	See Subclause C.2.1.3.2
"*object.item.videoItem.musicVideoClip*"	*O*	See Subclause C.2.1.3.3
"*object.item.playlistItem*"	*O*	See Subclause C.2.1.4
"*object.item.textItem*"	*O*	See Subclause C.2.1.5
"*object.item.bookmarkItem*"	*O*	See Subclause C.2.1.6
"*object.item.epgItem*"	*O*	See Subclause C.2.1.7
"*object.item.epgItem.audioProgram*"	*O*	See Subclause C.2.1.7.1
"*object.item.epgItem.videoProgram*"	*O*	See Subclause C.2.1.7.2
"*object.container.person*"	*O*	See Subclause C.2.2.1
"*object.container.person.musicArtist*"	*O*	See Subclause C.2.2.1.1
"*object.container.playlistContainer*"	*O*	See Subclause C.2.2.2

Value	R/O	Description
"object.container.album"	O	See Subclause C.2.2.3
"object.container.album.musicAlbum"	O	See Subclause C.2.2.3.1
"object.container.album.photoAlbum"	O	See Subclause C.2.2.3.2
"object.container.genre"	O	See Subclause C.2.2.4
"object.container.genre.musicGenre"	O	See Subclause C.2.2.4.1
"object.container.genre.movieGenre"	O	See Subclause C.2.2.4.2
"object.container.channelGroup"	O	See Subclause C.2.2.5
"object.container.channelGroup.audioChannelGroup"	O	See Subclause C.2.2.5.1
"object.container.channelGroup.videoChannelGroup"	O	See Subclause C.2.2.5.2
"object.container.epgContainer"	O	See Subclause C.2.2.6
"object.container.storageSystem"	O	See Subclause C.2.2.7
"object.container.storageVolume"	O	See Subclause C.2.2.8
"object.container.storageFolder"	O	See Subclause C.2.2.9
"object.container.bookmarkFolder"	O	See Subclause C.2.2.10
Vendor-defined	X	

B.1.10.2 *upnp:class@name*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** NO

Description: The *upnp:class@name* property indicates a 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.

Default Value: None.

B.1.11 *upnp:searchClass*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** YES

Description: The *read-only* *upnp:searchClass* property is only applicable to container objects. It contains a class for which the container object can be searched.

If *@searchable* = "1", then

- If no *upnp:searchClass* properties are specified, then the *Search()* action can return any match.
- If *upnp:searchClass* properties are specified, then the *Search()* action MUST only return matches from the classes specified in the *upnp:searchClass* properties.
- *upnp:searchClass* is OPTIONAL.
- *upnp:searchClass* is always determined by the ContentDirectory service.
- *upnp:searchClass* semantics are per container, there is no parent-child relationship, they only apply to searches started from that container.

else

- The container and its subtrees are not searchable.
- The values of the *upnp:searchClass* properties are meaningless and therefore the *upnp:searchClass* properties SHOULD NOT be included.

Default Value: If *@searchable* = "1", then all classes can be searched.

B.1.11.1 *upnp:searchClass@name*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read-only* *upnp:searchClass@name* property indicates a friendly name for the class.

Default Value: None.

B.1.11.2 *upnp:searchClass@includeDerived*

Namespace: upnp **Property Data Type:** xsd:boolean **Multi-Valued:** *NO*

Description: The *read-only* *upnp:searchClass@includeDerived* property is a REQUIRED property of the associated *upnp:searchClass* property and indicates whether the class specified MUST also include derived classes. When set to "*1*", derived classes MUST be included. When set to "*0*", derived classes MUST be excluded.

Default Value: N/A – The property is REQUIRED when the *upnp:searchClass* property is present.

B.1.12 *upnp:createClass*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *read-only* *upnp:createClass* property is only applicable to container objects. It contains a class that can be created within the container object.

If *@restricted* = "*0*", then

- If no *upnp:createClass* properties are specified, then *CreateObject()* MAY create any class of object under the container.
- If *upnp:createClass* properties are specified, then *CreateObject()* MUST only create classes of objects specified in the *upnp:createClass* properties.
- *upnp:createClass* is OPTIONAL.
- *upnp:createClass* semantics are per container, there is no parent-child relationship, they only apply to *CreateObject()* actions in that container.

else

- *CreateObject()* MUST fail since the container can not be modified.

Default Value: If *@restricted* = "*0*", then any class of object MAY be created under the container.

B.1.12.1 *upnp:createClass@name*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read-only* *upnp:createClass* property indicates a friendly name for the class.

Default Value: None.

B.1.12.2 *upnp:createClass@includeDerived*

Namespace: upnp **Property Data Type:** xsd:boolean **Multi-Valued:** *NO*

Description: The *read-only* *upnp:createClass@includeDerived* property is a REQUIRED property of the associated *upnp:createClass* property and indicates that the class specified also includes derived classes. When set to "*1*", derived classes must be included. When set to "*0*", derived classes must be excluded.

Default Value: N/A – The property is REQUIRED when the *upnp:createClass* property is present.

B.1.13 *upnp:writeStatus*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read-only* *upnp:writeStatus* property controls the modifiability of the resources of a given object. The ability for a control point to change the value of the *upnp:writeStatus* property is implementation dependent.

Default Value: "*UNKNOWN*".

B.1.13.1 allowedValueList for the *upnp:writeStatus* Property

Table B.4 — allowedValueList for the *upnp:writeStatus* Property

Value	R/O	Description
"*WRITABLE*"	*O*	The object's resource(s) MAY be deleted and/or modified.
"*PROTECTED*"	*O*	The object's resource(s) MAY NOT be deleted and/or modified.
"*NOT_WRITABLE*"	*O*	The object's resource(s) MAY NOT be modified.
"*UNKNOWN*"	*O*	The object's resource(s) write status is unknown.
"*MIXED*"	*O*	Some of the object's resource(s) have a different write status.

B.2 Resource Encoding Characteristics Properties

Table B.5 — Resource Encoding Characteristics Properties Overview

Property Name	NS	Data Type	M-Val	Reference
res	DIDL-Lite	xsd:anyURI	*NO*	Subclause B.2.1
res@protocolInfo	DIDL-Lite	xsd:string	*NO*	Subclause B.2.1.1
res@importUri	DIDL-Lite	xsd:anyURI	*NO*	Subclause B.2.1.2
res@size	DIDL-Lite	xsd:unsignedLong	*NO*	Subclause B.2.1.3
res@duration	DIDL-Lite	xsd:string	*NO*	Subclause B.2.1.4
res@protection	DIDL-Lite	xsd:string	*NO*	Subclause B.2.1.5
res@bitrate	DIDL-Lite	xsd:unsignedInt	*NO*	Subclause B.2.1.6
res@bitsPerSample	DIDL-Lite	xsd:unsignedInt	*NO*	Subclause B.2.1.7
res@sampleFrequency	DIDL-Lite	xsd:unsignedInt	*NO*	Subclause B.2.1.8
res@nrAudioChannels	DIDL-Lite	xsd:unsignedInt	*NO*	Subclause B.2.1.9
res@resolution	DIDL-Lite	xsd:string	*NO*	Subclause B.2.1.10
res@colorDepth	DIDL-Lite	xsd:unsignedInt	*NO*	Subclause B.2.1.11
res@tspec	DIDL-Lite	xsd:string	*NO*	Subclause B.2.1.12
res@allowedUse	DIDL-Lite	CSV (xsd:string)	*NO*	Subclause B.2.1.13
res@validityStart	DIDL-Lite	xsd:string	*NO*	Subclause B.2.1.14

Property Name	NS	Data Type	M-Val	Reference
res@validityEnd	DIDL-Lite	xsd:string	NO	Subclause B.2.1.15
res@remainingTime	DIDL-Lite	xsd:string	NO	Subclause B.2.1.16
res@usageInfo	DIDL-Lite	xsd:string	NO	Subclause B.2.1.17
res@rightsInfoURI	DIDL-Lite	xsd:anyURI	NO	Subclause B.2.1.18
res@contentInfoURI	DIDL-Lite	xsd:anyURI	NO	Subclause B.2.1.19
res@recordQuality	DIDL-Lite	CSV (xsd:string)	NO	Subclause B.2.1.20
res@daylightSaving	DIDL-Lite	xsd:string	NO	Subclause B.2.1.21

B.2.1 *res*

See Subclause B.1.9, "*res*".

B.2.1.1 *res@protocolInfo*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** NO

Description: This REQUIRED property identifies the protocol that MUST be used to transmit the resource (see also UPnP A/V Connection Manager Service template, subclause 2.5.2).

Default Value: N/A – The property is REQUIRED when the *res* property is present.

B.2.1.2 *res@importUri*

Namespace: DIDL-Lite **Property Data Type:** xsd:anyURI **Multi-Valued:** NO

Description: The *read-only* *res@importUri* property indicates the URI via which the resource can be imported to the ContentDirectory service via the *ImportResource()* action or HTTP POST. The *res@importUri* property identifies a *download portal* for the associated *res* property of a specific target object. It is used to create a local copy of the external content. After the transfer finishes successfully, the local content is then associated with the target object by setting the target object's *res* property value to a URI for that content, which MAY or MAY NOT be the same URI as the one specified in the *res@importUri* property, depending on the ContentDirectory service implementation.

Default Value: None.

B.2.1.3 *res@size*

Namespace: DIDL-Lite **Property Data Type:** xsd:unsignedLong **Multi-Valued:** NO

Description: The *res@size* property indicates the size in bytes of the resource. This property is a 64-bit unsigned integer.

Default Value: None.

B.2.1.4 *res@duration*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** NO

Description: The *res@duration* property indicates the time 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+: one or more 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 a “+” or “-” sign, and the decimal point itself MUST be omitted if there are no fractional second digits.

Default Value: None.

B.2.1.5 *res@protection*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *res@protection* property contains some identification of a protection system used for the resource.

Default Value: None.

B.2.1.6 *res@bitrate*

Namespace: DIDL-Lite **Property Data Type:** xsd:unsignedInt **Multi-Valued:** *NO*

Description: The *res@bitrate* property indicates the bitrate in **bytes/second** of the encoding of the resource.

Note that there exists an inconsistency with a *res@bitrate* property name and its value being expressed in bytes/sec.

In case the resource has been encoded using variable bitrate (VBR), it is RECOMMENDED that the *res@bitrate* value represents the average bitrate, calculated over the entire duration of the resource (total number of bytes divided by the total duration of the resource).

The *res@bitrate* value SHOULD NOT be taken as sufficient from a QoS or other perspective to prepare for the stream; The protocol used and the physical layer headers may increase the actual required bandwidth.

Default Value: None.

B.2.1.7 *res@bitsPerSample*

Namespace: DIDL-Lite **Property Data Type:** xsd:unsignedInt **Multi-Valued:** *NO*

Description: The *res@bitsPerSample* property indicates the number of bits used to represent one sample of the resource.

Default Value: None.

B.2.1.8 *res@sampleFrequency*

Namespace: DIDL-Lite **Property Data Type:** xsd:unsignedInt **Multi-Valued:** *NO*

Description: The *res@sampleFrequency* property indicates the sample frequency used to digitize the audio resource. Expressed in Hz.

Default Value: None.

B.2.1.9 *res@nrAudioChannels*

Namespace: DIDL-Lite **Property Data Type:** xsd:unsignedInt **Multi-Valued:** *NO*

Description: The *res@nrAudioChannels* property indicates the number of audio channels present in the audio resource, for example, 1 for mono, 2 for stereo, 6 for Dolby Surround.

Default Value: None.

B.2.1.10 *res@resolution*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *res@resolution* property indicates the XxY resolution, in pixels, of the resource (typically an *imageItem* or *videoItem*). The string pattern is of the form: "`[0-9]+x[0-9]+`" (one or more digits, followed by "x", followed by one or more digits).

Default Value: None.

B.2.1.11 *res@colorDepth*

Namespace: DIDL-Lite **Property Data Type:** xsd:unsignedInt **Multi-Valued:** *NO*

Description: The *res@colorDepth* property indicates the number of bits per pixel used to represent the video or image resource.

Default Value: None.

B.2.1.12 *res@tspec*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *res@tspec* property identifies the content's QoS (quality of service) characteristics. It has a maximum length of 256 characters. The details about this property, including its components and formatting constraints, are defined in the QoS Manager service definition document.

Default Value: None.

B.2.1.13 *res@allowedUse*

Namespace: DIDL-Lite **Property Data Type:** CSV (xsd:string) **Multi-Valued:** *NO*

Description: The *res@allowedUse* property is composed of a comma-separated list of value pairs. Each value pair is composed of an enumerated string value, followed by a colon (":"), followed by an integer. For example, "*PLAY:5,COPY:1*".

In each pair, the first value corresponds to an allowed use for the resource referenced by the associated *res* property. RECOMMENDED enumerated values are: "*PLAY*", "*COPY*", "*MOVE*" and "*UNKNOWN*". Vendors may extend this list. The "*UNKNOWN*" value is the default value when new resources are created. A value of "*UNKNOWN*" indicates that allowed uses for this resource may exist, but have not been reflected in the ContentDirectory service.

Any resource that has accompanying constraints on uses must expose a value for the *res@allowedUse* property. Any use of the resource that does not appear explicitly in the *res@allowedUse* property must be assumed to be disallowed. When the *res@allowedUse* property is not present, there are no use constraints on the resource.

The second quantity corresponds to the number of times the specified use is allowed to occur. A value of "*-1*" indicates that there is no limit on the number of times this use may occur.

It is recommended to update this value when the number of allowed uses changes. For example, a resource with the *res@allowedUse* property initially set to "*COPY:1*" should be updated to "*COPY:0*"after a copy has been successfully completed.

Default Value: "*UNKNOWN*".

B.2.1.14 *res@validityStart*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *res@validityStart* property defines the beginning date&time when the corresponding uses described in the *res@allowedUse* property become valid. The format of the *res@validityStart* property MUST comply with the `date-time` syntax as defined in Annex D, "**(normative)** EBNF Syntax Definitions".

The following example value designates May 30, 2004, 1:20pm, as a validity interval beginning value: "2004-05-30T13:20:00-05:00".

When the *res@validityStart* property is not present, the beginning of the validity interval is assumed to have already started.

Default Value: The validity interval is assumed to have already started.

B.2.1.15 *res@validityEnd*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *res@validityEnd* property defines the ending date&time when the corresponding uses described in the *res@allowedUse* property become invalid. The format of the *res@validityEnd* property MUST comply with the `date-time` syntax as defined in Annex D, "**(normative)** EBNF Syntax Definitions".

When the *res@validityEnd* property is not present, there correspondingly is no end to the validity interval.

Default Value: There is no end to the validity interval.

B.2.1.16 *res@remainingTime*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *res@remainingTime* property is used to indicate the amount of time remaining until the use specified in the *res@allowedUse* property is revoked. The remaining time is an aggregate amount of time that the resource may be used either continuously or in discrete intervals. When both *res@remainingTime* and *res@validityEnd* are specified, the use is revoked either when *res@remainingTime* reaches zero, or when the *res@validityEnd* time is reached, whichever occurs first. The format of the *res@remainingTime* property MUST comply with the `duration` syntax as defined in Annex D, “**(normative)**

EBNF Syntax Definitions”.

Example: “P08:03:10” indicates that the resource is available for an additional 8 hours, 3 minutes and 10 seconds.

Note: In order to prevent disruptive network overuse, ContentDirectory implementations should be juditious when deciding how frequently to update this property. If the *res@remainingTime* property represents a continuous change, its value should be modified whenever a key milestone is reached. For example, when the property’s value decreases to a whole number of hours remaining. Alternatively, if the *res@remainingTime* property is used to track descrete usage intervals such as an hour’s worth of viewing, the property should be updated whenever a block of time is subtracted from the remaining time.

Default Value: None.

B.2.1.17 *res@usageInfo*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *res@usageInfo* property contains a user-friendly string with additional information about the allowed use of the resource, as in the example: *"Playing of the movie is allowed in high-definition mode. One copy is allowed to be made, but only the standard definition version may be copied"*.

Default Value: None.

B.2.1.18 *res@rightsInfoURI*

Namespace: DIDL-Lite **Property Data Type:** xsd:anyURI **Multi-Valued:** *NO*

Description: The *res@rightsInfoURI* property references an html page and a web site associated with the rights vendor for the resource. The referenced page SHOULD assist the user interface in documenting the rights and the renewal of the allowed use of the resource.

Default Value: None.

B.2.1.19 *res@contentInfoURI*

Namespace: DIDL-Lite **Property Data Type:** xsd:anyURI **Multi-Valued:** *NO*

Description: Each *res@contentInfoURI* property contains a URI employed to assist the user interface in providing additional information to the user about the content referenced by the resource. The value of this property refers to an html page and a web site associated with the content vendor for the resource.

Default Value: None.

B.2.1.20 *res@recordQuality*

Namespace: DIDL-Lite **Property Data Type:** CSV (xsd:string) **Multi-Valued:** *NO*

Description: When the resource referenced by the *res* property was created by recording, the *res@recordQuality* property can be specified to indicate the quality level(s) used to make the recording. The *res@recordQuality* property is a CSV list of <type> ":" <recording quality> pairs. The type and quality in each pair are separated by a colon character (":"). The type portion indicates what kind of value system is used in the recording quality portion. The recording quality portion is the actual recording quality value used. When there is more than one pair of colon-separated values in the list, all pairs MUST represent the same quality level in different type systems. For detailed descriptions of the type and quality values, refer to the properties *srs:recordQuality@type* and *srs:recordQuality*, respectively, as defined in the ScheduledRecording service specification [SRS].

Default Value: None.

B.2.1.21 *res@daylightSaving*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *res@daylightSaving* property indicates whether the time values used in other *res*-dependent properties, such as the *res@validityStart* property and the *res@validityEnd* property, are expressed using as a reference either Daylight Saving Time or Standard Time. This property is only applicable when the time values in other *res*-dependent properties are expressed in local time. Whenever the time value in those properties are expressed in absolute time, the *res@daylightSaving* property MUST not be present on output and MUST be ignored on input.

Default Value: "*UNKNOWN*".

B.2.1.21.1 allowedValueList for the *res@daylightSaving* Property

Table B.6 — allowedValueList for the *res@daylightSaving* Property

Value	R/O	Description
"*DAYLIGHTSAVING*"	*O*	The reference point for the associated local time value is Daylight Saving Time, even if the indicated time falls outside the period of the year when Daylight Saving Time is actually observed.
"*STANDARD*"	*O*	The reference point for the associated local time value is Standard Time, even if the indicated time falls outside the period of the year when Standard Time is actually observed.
"*UNKNOWN*"	*O*	The reference point for the associated local time value depends on whether Daylight Saving Time is in effect or not. During the time interval starting one hour before the switch is made from Daylight Saving Time back to Standard time and ending one hour after that switching point however, the reference point is ambiguous and is device dependent.

B.3 Contributor-related Properties

Table B.7 — Contributor-related Properties Overview

Property Name	NS	Data Type	M-Val	Reference

Property Name	NS	Data Type	M-Val	Reference
upnp:artist	upnp	xsd:string	*YES*	Subclause B.3.1
upnp:artist@role	upnp	xsd:string	*NO*	Subclause B.3.1.1
upnp:actor	upnp	xsd:string	*YES*	Subclause B.3.2
upnp:actor@role	upnp	xsd:string	*NO*	Subclause B.3.2.1
upnp:author	upnp	xsd:string	*YES*	Subclause B.3.3
upnp:author@role	upnp	xsd:string	*NO*	Subclause B.3.3.1
upnp:producer	upnp	xsd:string	*YES*	Subclause B.3.4
upnp:director	upnp	xsd:string	*YES*	Subclause B.3.5
dc:publisher	dc	xsd:string	*YES*	Subclause B.3.6
dc:contributor	dc	xsd:string	*YES*	Subclause B.3.7

B.3.1 *upnp:artist*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *upnp:artist* property indicates the name of an artist.

Default Value: None.

B.3.1.1 *upnp:artist@role*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:artist@role* property indicates the role of the artist in the work.

Default Value: None.

B.3.2 *upnp:actor*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *upnp:actor* property indicates the name of an actor performing in (part of) the content.

Default Value: None.

B.3.2.1 *upnp:actor@role*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:actor@role* property indicates the role of the actor in the work.

Default Value: None.

B.3.3 *upnp:author*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *upnp:author* property indicates the name of an author contributing to the content (for example, the writer of a text book).

Default Value: None.

B.3.3.1 *upnp:author@role*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:author@role* property indicates the role of the author in the work.

Default Value: None.

B.3.4 *upnp:producer*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *upnp:producer* property indicates the name of a producer of the content (for example, a movie or a CD).

Default Value: None.

B.3.5 *upnp:director*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *upnp:director* property indicates the name of a director of the content (for example, a movie).

Default Value: None.

B.3.6 *dc:publisher*

Namespace: dc **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *dc:publisher* property indicates the name of a publisher of the content. See http://dublincore.org/documents/dces.

Default Value: None.

B.3.7 *dc:contributor*

Namespace: dc **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *dc:contributor* property indicates the name of a contributor to the content item. It is RECOMMENDED that *dc:contributor* property includes the name of the primary content creator or owner (Dublin Core 'creator' property). See http://dublincore.org/documents/dces.

Default Value: None.

B.4 Affiliation-related Properties

Table B.8 — Affiliation-related Properties Overview

Property Name	NS	Data Type	M-Val	Reference
upnp:genre	upnp	xsd:string	*YES*	Subclause B.4.1
upnp:genre@id	upnp	xsd:string	*NO*	Subclause B.4.1.1
upnp:genre@extended	upnp	CSV (xsd:string)	*NO*	Subclause B.4.1.2
upnp:album	upnp	xsd:string	*YES*	Subclause B.4.2

Property Name	NS	Data Type	M-Val	Reference
upnp:playlist	upnp	xsd:string	*YES*	Subclause B.4.3

B.4.1 *upnp:genre*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *upnp:genre* property indicates the genre to which an object belongs.

Default Value: None.

B.4.1.1 *upnp:genre@id*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:genre@id* property identifies the genre scheme which defines the set of names used in the *upnp:genre* and *upnp:genre@extended* property.

The format of the *upnp:genre@id* is:

<ICANN registered domain> "_" <genre_scheme_id>.

Example: "epg.com_GenreSet1"

The *upnp:genre@id* property is REQUIRED if the *upnp:genre@extended* property is specified.

Default Value: N/A – This property is REQUIRED when the *upnp:genre@extended* property is present.

B.4.1.2 *upnp:genre@extended*

Namespace: upnp **Property Data Type:** CSV (xsd:string) **Multi-Valued:** *NO*

Description: The *upnp:genre@extended* property MUST be a CSV list of genre names, which are individually displayable strings, representing increasingly precise (sub)genre names. The list MUST be ordered with the most general genre first. The first entry in the list MUST be equal to the value of the *upnp:genre* property.

Example: "Sports,Basketball,NBA"

Default Value: None.

B.4.2 *upnp:album*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *upnp:album* property indicates the title of the album to which the content item belongs.

Default Value: None.

B.4.3 *upnp:playlist*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *upnp:playlist* property indicates the name of a playlist (the *dc:title* of a *playlistItem*) to which the content item belongs.

Default Value: None.

B.5 Associated Resources Properties

Table B.9 — Associated Resources Properties Overview

Property Name	NS	Data Type	M-Val	Reference
upnp:albumArtURI	upnp	xsd:anyURI	*YES*	Subclause B.5.1
upnp:artistDiscographyURI	upnp	xsd:anyURI	*NO*	Subclause B.5.2
upnp:lyricsURI	upnp	xsd:anyURI	*NO*	Subclause B.5.3
dc:relation	dc	xsd:string	*YES*	Subclause B.5.4

B.5.1 *upnp:albumArtURI*

Namespace: upnp **Property Data Type:** xsd:anyURI **Multi-Valued:** *YES*

Description: The *upnp:albumArtURI* property contains a reference to album art. The value MUST be a properly escaped URI as described in [RFC 2396].

Default Value: None.

B.5.2 *upnp:artistDiscographyURI*

Namespace: upnp **Property Data Type:** xsd:anyURI **Multi-Valued:** *NO*

Description: The *upnp:artistDiscographyURI* property contains a reference to the artist's discography. The value MUST be a properly escaped URI as described in [RFC 2396].

Default Value: None.

B.5.3 *upnp:lyricsURI*

Namespace: upnp **Property Data Type:** xsd:anyURI **Multi-Valued:** *NO*

Description: The *upnp:lyricsURI* property contains a reference to lyrics of the song or of the whole album. The value MUST be a properly escaped URI as described in [RFC 2396].

Default Value: None.

B.5.4 *dc:relation*

Namespace: dc **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: See http://dublincore.org/documents/dces. The value MUST be a properly escaped URI as described in [RFC 2396].

Default Value: None.

B.6 Storage-Related Properties

Table B.10 — Storage-Related Properties Overview

Property Name	NS	Data Type	M-Val	Reference
upnp:storageTotal	upnp	xsd:long	*NO*	Subclause B.6.1

Property Name	NS	Data Type	M-Val	Reference
upnp:storageUsed	upnp	xsd:long	*NO*	Subclause B.6.2
upnp:storageFree	upnp	xsd:long	*NO*	Subclause B.6.3
upnp:storageMaxPartition	upnp	xsd:long	*NO*	Subclause B.6.4
upnp:storageMedium	upnp	xsd:string	*NO*	Subclause B.6.5

B.6.1 *upnp:storageTotal*

Namespace: upnp **Property Data Type:** xsd:long **Multi-Valued:** *NO*

Description: The *read-only* *upnp:storageTotal* property contains the total capacity, in bytes, of the storage represented by the container. Value -1 is reserved to indicate that the capacity is unknown.

Default Value: None.

B.6.2 *upnp:storageUsed*

Namespace: upnp **Property Data Type:** xsd:long **Multi-Valued:** *NO*

Description: The *read-only* *upnp:storageUsed* property contains the 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.

Default Value: None.

B.6.3 *upnp:storageFree*

Namespace: upnp **Property Data Type:** xsd:long **Multi-Valued:** *NO*

Description: The *read-only* *upnp:storageFree* property contains the total free capacity, in bytes, of the storage represented by the container. Value -1 is reserved to indicate that the capacity is unknown.

Default Value: None.

B.6.4 *upnp:storageMaxPartition*

Namespace: upnp **Property Data Type:** xsd:long **Multi-Valued:** *NO*

Description: The *read-only* *upnp:storageMaxPartition* property contains the 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.

Default Value: None.

B.6.5 *upnp:storageMedium*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read-only* *upnp:storageMedium* property indicates the type of storage medium used for the content. Potentially useful for user-interface purposes.

Default Value: "*UNKNOWN*".

B.6.5.1 allowedValueList for the *upnp:storageMedium* Property

See Table 2-4, "allowedValueList for *PlaybackStorageMedium* and *RecordStorageMedium*" of the AVTransport:1 service specification.

B.7 General Description (mainly for UI purposes) Properties

Table B.11 — General Description (mainly for UI purposes) Properties Overview

Property Name	NS	Data Type	M-Val	Reference
dc:description	dc	xsd:string	*NO*	Subclause B.7.1
upnp:longDescription	upnp	xsd:string	*NO*	Subclause B.7.2
upnp:icon	upnp	xsd:anyURI	*NO*	Subclause B.7.3
upnp:region	upnp	xsd:string	*NO*	Subclause B.7.4
dc:rights	dc	xsd:string	*YES*	Subclause B.7.5
dc:date	dc	xsd:string	*NO*	Subclause B.7.6
dc:date@ upnp:daylightSaving	dc	xsd:string	*NO*	Subclause B.7.6.1
dc:language	dc	xsd:string	*YES*	Subclause B.7.7
upnp:playbackCount	upnp	xsd:int	*NO*	Subclause B.7.8
upnp:lastPlaybackTime	upnp	xsd:string	*NO*	Subclause B.7.9
upnp:lastPlaybackTime @daylightSaving	upnp	xsd:string	*NO*	Subclause B.7.9.1
upnp:lastPlaybackPosition	upnp	xsd:string	*NO*	Subclause B.7.10
upnp:recordedStartDateTime	upnp	xsd:string	*NO*	Subclause B.7.11
upnp:recordedStartDateTime @daylightSaving	upnp	xsd:string	*NO*	Subclause B.7.11.1
upnp:recordedDuration	upnp	xsd:string	*NO*	Subclause B.7.12
upnp:recordedDayOfWeek	upnp	xsd:string	*NO*	Subclause B.7.13
upnp:srsRecordScheduleID	upnp	xsd:string	*NO*	Subclause B.7.14
upnp:srsRecordTaskID	upnp	xsd:string	*NO*	Subclause B.7.15
upnp:recordable	upnp	xsd:boolean	*NO*	Subclause B.7.16

B.7.1 *dc:description*

Namespace: dc **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *dc:description* property contains a brief description of the content item. See http://dublincore.org/documents/dces.

Default Value: None.

B.7.2 *upnp:longDescription*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:longDescription* property contains a few lines of description of the content item (longer than the *dc:description* property).

Default Value: None.

B.7.3 *upnp:icon*

Namespace: upnp **Property Data Type:** xsd:anyURI **Multi-Valued:** *NO*

Description: The *upnp:icon* property contains a URI to some icon that a control point can use in its UI to display the content, for example, a CNN logo for a Tuner channel. It is RECOMMENDED that the same format be used as is used for the icon element in the UPnP device description document schema (PNG). The value MUST be a properly escaped URI as described in [RFC 2396].

Default Value: None.

B.7.4 *upnp:region*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:region* property contains some identification of the region, associated with the source of the object, for example, “US”, “Latin America”, “Seattle”.

Default Value: None.

B.7.5 *upnp:rights*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *upnp:rights* property contains some descriptive information about the legal rights held in or over this resource. (http://dublincore.org/documents/dces)

Default Value: None.

B.7.6 *dc:date*

Namespace: dc **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *dc:date* property contains the primary date of the content. The format MUST be compliant to [ISO 8601] and SHOULD be compliant to [RFC 3339]. See http://dublincore.org/documents/dces.

Examples:

- 2004-05-14
- 2004-05-14T14:30:05
- 2004-05-14T14:30:05+09:00

Default Value: None.

B.7.6.1 *dc:date@upnp:daylightSaving*

Namespace: dc **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *dc:date@upnp:daylightSaving* property indicates whether the time value used in the *dc:date* property is expressed using as a reference either Daylight Saving Time or Standard Time. This property is only applicable when the time value in the *dc:date* property is expressed in local time. Whenever the time value in the *dc:date* property is expressed in absolute time, the *dc:date@upnp:daylightSaving* property MUST not be present on output and MUST be ignored on input.

Default Value: "*UNKNOWN*".

B.7.6.1.1 allowedValueList for the *dc:date@upnp:daylightSaving* Property

See Table B.6 in B.2.1.21.1, "allowedValueList for the *res@daylightSaving* Property".

B.7.7 *dc:language*

Namespace: dc **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *dc:language* property indicates one of the languages used in the content as defined by RFC 3066, for example, "en-US". See http://dublincore.org/documents/dces.

Default Value: None.

B.7.8 *upnp:playbackCount*

Namespace: upnp **Property Data Type:** xsd:int **Multi-Valued:** *NO*

Description: The *read-only* *upnp:playbackCount* property contains the number of times the content has been played. The special value -1 means that the content has been played but the count is unknown. The criteria for determining whether the content has been played, is device dependent.

Default Value: None.

B.7.9 *upnp:lastPlaybackTime*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:lastPlaybackTime* property contains the date&ime of the last playback.

The format of the *upnp:lastPlaybackTime* property MUST comply with the `date-time` syntax as defined in Annex D, "**(normative)**

EBNF Syntax Definitions".

The criteria for determining when the content has been played last, is device dependent.

Default Value: None.

B.7.9.1 *upnp:lastPlaybackTime@daylightSaving*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:lastPlaybackTime@daylightSaving* property indicates whether the time value used in the *upnp:lastPlaybackTime* property is expressed using as a reference either Daylight Saving Time or Standard Time. This property is only applicable when the time value in the *upnp:lastPlaybackTime* property is expressed in local time. Whenever the time value in the *upnp:lastPlaybackTime* property is expressed in absolute time, the *upnp:lastPlaybackTime@daylightSaving* property MUST not be present on output and MUST be ignored on input.

Default Value: "*UNKNOWN*".

B.7.9.1.1 allowedValueList for the *upnp:lastPlaybackTime@daylightSaving* Property

See Table B.6 in B.2.1.21.1, “allowedValueList for the *res@daylightSaving* Property”.

B.7.10 *upnp:lastPlaybackPosition*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:lastPlaybackPosition* property conbtains the time offset within the content where the last playback was suspended.

The format of the *upnp:lastPlaybackPosition* property MUST comply with the `duration` syntax as defined in Annex D, “**(normative)**

EBNF Syntax Definitions”.

The criteria for determining the time offset in the content where the playback of the content has been suspended, is device dependent.

Default Value: None.

B.7.11 *upnp:recordedStartDateTime*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:recordedStartDateTime* property contains the date&time when the recording *started*.

The format of the *upnp:recordedStartDateTime* property MUST comply with the `date-time` syntax as defined in Annex D, “**(normative)**

EBNF Syntax Definitions”.

Default Value: None.

B.7.11.1 *upnp:recordedStartDateTime@daylightSaving*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:recordedStartDateTime@daylightSaving* property indicates whether the time value used in the *upnp:recordedStartDateTime* property is expressed using as a reference either Daylight Saving Time or Standard Time. This property is only applicable when the time value in the *upnp:recordedStartDateTime* property is expressed in local time. Whenever the time value in the *upnp:recordedStartDateTime* property is expressed in absolute time, the *upnp:recordedStartDateTime@daylightSaving* property MUST not be present on output and MUST be ignored on input.

Default Value: “*UNKNOWN*”.

B.7.11.1.1 allowedValueList for the *upnp:recordedStartDateTime@daylightSaving* Property

See Table B.6 in B.2.1.21.1, “allowedValueList for the *res@daylightSaving* Property”.

B.7.12 *upnp:recordedDuration*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:recordedDuration* property contains the duration of the recorded content.

The format of the *upnp:recordedDuration* property MUST comply with the `duration` syntax as defined in Annex D, "**(normative)**

EBNF Syntax Definitions".

Default Value: None.

B.7.13 *upnp:recordedDayOfWeek*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:recordedDayOfWeek* property contains the day of the week when the recording *started*.

Sorting for this property is based on the order in Table B.12. Ascending: first table entry first.

Default Value: None.

B.7.13.1 allowedValueList for the *upnp:recordedDayOfWeek* Property

Table B.12 — allowedValueList for the upnp:recordedDayOfWeek Property

Value	R/O	Description
"*SUN*"	*R*	
"*MON*"	*R*	
"*TUE*"	*R*	
"*WED*"	*R*	
"*THU*"	*R*	
"*FRI*"	*R*	
"*SAT*"	*R*	

B.7.14 *upnp:srsRecordScheduleID*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read-only* *upnp:srsRecordScheduleID* property contains the value of the *srs:@id* property of the *srs:recordSchedule* object that was used to create this recorded content. Refer to the ScheduledRecording service specification [SRS] for details.

Default Value: None.

B.7.15 *upnp:srsRecordTaskID*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read-only* *upnp:srsRecordTaskID* property contains the value of the *srs:@id* property of the *srs:recordTask* object that was used to create this recorded content. Refer to the ScheduledRecording service specification [SRS] for details.

Default Value: None.

B.7.16 *upnp:recordable*

Namespace: upnp **Property Data Type:** xsd: boolean **Multi-Valued:** *NO*

Description: When the *upnp:recordable* property is set to "*1*", the content represented by this object can potentially be used for recording purposes. If the object is not self-contained (such as an object of class other than "*object.item.epgItem*"), other information may be needed to set up the recording. When set to "*0*", the content represented by this object is not accessible for recording due to various reasons, such as hardware limitations.

Default Value: "*1*".

B.8 Recorded Object-related Properties

Table B.13 — Recorded Object-related Properties Overview

Property Name	NS	Data Type	M-Val	Reference
upnp:programTitle	upnp	xsd:string	*NO*	Subclause B.8.1
upnp:seriesTitle	upnp	xsd:string	*NO*	Subclause B.8.2
upnp:programID	upnp	xsd:string	*NO*	Subclause B.8.3
upnp:programID@type	upnp	xsd:string	*NO*	Subclause B.8.3.1
upnp:seriesID	upnp	xsd:string	*NO*	Subclause B.8.4
upnp:seriesID@type	upnp	xsd:string	*NO*	Subclause B.8.4.1
upnp:channelID	upnp	xsd:string	*YES*	Subclause B.8.5
upnp:channelID@type	upnp	xsd:string	*NO*	Subclause B.8.5.1
upnp:channelID@distriNetworkName	upnp	xsd:string	*NO*	Subclause B.8.5.2
upnp:channelID@distriNetworkID	upnp	xsd:string	*NO*	Subclause B.8.5.3
upnp:episodeCount	upnp	xsd:unsignedInt	*NO*	Subclause B.8.5.2
upnp:episodeNumber	upnp	xsd:unsignedInt	*NO*	Subclause B.8.7
upnp:programCode	upnp	xsd:string	*NO*	Subclause B.8.8
upnp:programCode@type	upnp	xsd:string	*NO*	Subclause B.8.8.1
upnp:rating	upnp	xsd:string	*NO*	Subclause B.8.9
upnp:rating@type	upnp	xsd:string	*NO*	Subclause B.8.9.1
upnp:episodeType	upnp	xsd:string	*NO*	Subclause B.8.10

B.8.1 *upnp:programTitle*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:programTitle* property contains the name of the program. This is most likely obtained from a database that contains program-related information, such as an Electronic Program Guide.

Example: "Friends Series Finale".

Note: To be precise, this is different from the *dc:title* property which indicates a friendly name for the ContentDirectory service *object*. However, in many cases, the *dc:title* property will be set to the same value as the *upnp:programTitle* property.

Default Value: None.

B.8.2 *upnp:seriesTitle*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:seriesTitle* property contains the name of the series. This is most likely obtained from a database that contains program-related information, such as an Electronic Program Guide.

Default Value: None.

B.8.3 *upnp:programID*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:programID* property contains the unique ID of a program.

When this content is created via a ScheduledRecording service, this is the value of the *srs:matchedID* property of the *recordTask* that generated this content. Otherwise, the *upnp:programID* property value is set by the ContentDirectory service based on some device dependent information, such as an EPG database.

The format and semantics are identical to those of the *srs:matchedID* property, defined in the ScheduledRecording service specification. Refer to the ScheduledRecording service specification [SRS] for details.

Default Value: None.

B.8.3.1 *upnp:programID@type*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:programID@type* property indicates the type of the ID that is contained in the *upnp:programID* property. The format and allowed values are identical to those of the *srs:matchedID@type* property, defined in the ScheduledRecording service specification. Refer to the ScheduledRecording service specification [SRS] for details.

The *upnp:programID@type* property is REQUIRED if the *upnp:programID* property is specified.

Default Value: N/A – The property is REQUIRED when the *upnp:programID* property is present.

B.8.4 *upnp:seriesID*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:seriesID* property contains the unique ID of a series.

When this content is created via a ScheduledRecording service, this is the value of the *srs:matchedID* property of the *recordTask* that generated this content. Otherwise, the *upnp:seriesID* property value is set by the ContentDirectory service based on some device dependent information, such as an EPG database.

The format and semantics are identical to those of the *srs:matchedID* property, defined in the ScheduledRecording service specification. Refer to the ScheduledRecording service specification [SRS] for details.

Default Value: None.

B.8.4.1 *upnp:seriesID@type*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:seriesID@type* property indicates the type of the ID that is contained in the *upnp:seriesID* property. The format and allowed values are identical to those of the *srs:matchedID@type* property, defined in the ScheduledRecording service specification. Refer to the ScheduledRecording service specification [SRS] for details.

The *upnp:seriesID@type* property is REQUIRED if the *upnp:seriesID* property is specified.

Default Value: N/A – The property is REQUIRED when the *upnp:seriesID* property is present.

B.8.5 *upnp:channelID*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: When this content is created via a ScheduledRecording service, the *upnp:channelID* property identifies the channel that was the source. Otherwise, the *upnp:channelID* value indicates the channel that is associated with the content item. For example, when present in an object that represents a tuner channel, it contains the ID of that channel.

The possible formats and the dependency on the *upnp:channelID@type* property are identical to the possible formats of the *srs:scheduledChannelID* and its dependency on the *srs:scheduledChannelID@type* property as described in the ScheduledRecording service [SRS].

The *upnp:channelID* property is multi-valued so that different formats can be used to identify a particular channel. For example, if both the analog channel number and the analog channel frequency are known for the same channel, they can be advertised through the following construct:

```
<upnp:channelID type="ANALOG">5</upnp:channelID>
<upnp:channelID type="FREQUENCY">79000000</upnp:channelID>
```

When multiple instances of the *upnp:channelID* property are included, they MUST refer to the same channel.

Default Value: None.

B.8.5.1 *upnp:channelID@type*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:channelID@type* property determines the format that is used for the *upnp:channelID* property as defined above.

The possible formats and allowed values of the *upnp:channelID@type* property are identical to the possible formats of the *srs:scheduledChannelID@type* property as described in the ScheduledRecording service specification [SRS].

The *upnp:channelID@type* property is REQUIRED if the *upnp:channelID* property is specified.

Default Value: N/A – The property is REQUIRED when the *upnp:channelID* property is present.

B.8.5.2 *upnp:channelID@distriNetworkName*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:channelID@distriNetworkName* property definition is identical to the definition of the *srs:scheduledChannelID@distriNetworkName* property as described in the ScheduledRecording service specification [SRS].

When multiple instances of the *upnp:channelID* property are included, they MUST all either expose the *upnp:channelID@distriNetworkName* property or omit this property. If exposed, all *upnp:channelID@distriNetworkName* properties MUST have the same value.

Default Value: None.

B.8.5.3 *upnp:channelID@distriNetworkID*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:channelID@distriNetworkID* property definition is identical to the definition of the *srs:scheduledChannelID@distriNetworkID* property as described in the ScheduledRecording service specification [SRS].

When multiple instances of the *upnp:channelID* property are included, they MUST all either expose the *upnp:channelID@distriNetworkID* property or omit this property. If exposed, all *upnp:channelID@distriNetworkID* properties MUST have the same value.

Default Value: None.

B.8.6 *upnp:episodeCount*

Namespace: upnp **Property Data Type:** xsd: unsignedInt **Multi-Valued:** *NO*

Description: The *upnp:episodeCount* property contains the total number of episodes in the series to which this content belongs.

Default Value: None.

B.8.7 *upnp:episodeNumber*

Namespace: upnp **Property Data Type:** xsd:unsignedInt **Multi-Valued:** *NO*

Description: The *upnp:episodeCount* property contains the episode number of this recorded content within the series to which this content belongs.

Default Value: None.

B.8.8 *upnp:programCode*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:programCode* property contains a unique program code.

When this content is created via a ScheduledRecording service, this is the value of the *srs:taskProgramCode* property of the *recordTask* that generated this content. Otherwise, the *upnp:programCode* property value is set by the ContentDirectory service based on some device dependent information, such as an EPG database.

The format and semantics are identical to those of the *srs:taskProgramCode* property, defined in the ScheduledRecording service specification. Refer to the ScheduledRecording service specification [SRS] for details.

Default Value: None.

B.8.8.1 *upnp:programCode@type*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:programCode@type* property indicates t indicates the type of the program guide service that defines the program code specified in the *upnp:programCode* property. The format and allowed values are identical to those of the *srs:taskProgramCode@type* property, defined in the ScheduledRecording service specification. See [SRS] for details.

The *upnp:programCode@type* property is REQUIRED if the *upnp:programCode* property is specified.

Default Value: N/A – The property is REQUIRED when the *upnp:programCode* property is present.

B.8.9 *upnp:rating*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *upnp:rating* property contains the viewer rating value of the content of this item expressed in the rating system indicated by the *upnp:rating@type* property. The format and semantics of the *upnp:rating* property are identical to those of the *srs:matchedRating* property, defined in the ScheduledRecording service specification. Refer to the ScheduledRecording service specification [SRS] for details.

Default Value: None.

B.8.9.1 *upnp:rating@type*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:rating@type* property indicates the rating system used in the *upnp:rating* property. The format and allowed values of the *upnp:rating@type* property are identical to those of the *srs:matchedRating@type* property, defined in the ScheduledRecording service specification. Refer to the ScheduledRecording service specification [SRS] for details.

The *upnp:rating@type* property is highly RECOMMENDED if the *upnp:rating* property is specified.

Default Value: N/A.

B.8.10 *upnp:episodeType*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:episodeType* property value indicates the broadcast novelty (for example, "*FIRST-RUN*" or "*REPEAT*") of this content item. The format and allowed values of the *upnp:episodeType* property are identical to those of the *srs:matchedEpisodeType* property, defined in the ScheduledRecording service specification. Refer to the ScheduledRecording service specification [SRS] for details.

Default Value: None.

B.9 User Channel and EPG Related Properties

Table B.14 — User Channel and EPG Related Properties Overview

Property Name	NS	Data Type	M-Val	Reference
upnp:channelGroupName	upnp	xsd:string	*NO*	Subclause B.9.1
upnp:channelGroupName@id	upnp	xsd:string	*NO*	Subclause B.9.1.1
upnp:callSign	upnp	xsd:string	*NO*	Subclause B.9.2
upnp:networkAffiliation	upnp	xsd:string	*NO*	Subclause B.9.3
upnp:serviceProvider	upnp	xsd:string	*NO*	Subclause B.9.4
upnp:price	upnp	xsd:float	*YES*	Subclause B.9.5
upnp:price@currency	upnp	xsd:string	*NO*	Subclause B.9.5.1
upnp:payPerView	upnp	xsd:boolean	*NO*	Subclause B.9.6
upnp:epgProviderName	upnp	xsd:string	*NO*	Subclause B.9.7
upnp:dateTimeRange	upnp	xsd:string	*NO*	Subclause B.9.8
upnp:dateTimeRange @daylightSaving	upnp	xsd:string	*NO*	Subclause B.9.8.1

B.9.1 *upnp:channelGroupName*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:channelGroupName* property contains the user friendly name of the channelGroup.

Examples: “Digital Terrestrial”, “DirecTV”

A channel group defines a group of channels. A device that has multiple tuners may provide multiple channel groups. Moreover, a physical tuner device may provide multiple channel groups (for example, a set-top-box that contains a single tuner but supports three different input connections: terrestrial, cable, and satellite).

In a channel group, channels may be identified in various ways. For example, *upnp:channelID*, *upnp:channelName*, or *upnp:channelNr* may be used for that purpose.

Default Value: None.

B.9.1.1 *upnp:channelGroupName@id*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:channelGroupName@id* property contains the ID of a channel group to differentiate it from other channel groups implemented in a ContentDirectory service.

The format of the *upnp:channelGroupName@id* property is as follows:

<ICANN registered domain> “_” <channel group id defined in the domain>

Example: “broadcast.com_DigitalSatellite”

The *upnp:channelGroupName@id* property is REQUIRED if the *upnp:channelGroupName* property is specified.

Default Value: N/A – The property is REQUIRED when the *upnpchannelGroupName* property is present.

B.9.2 *upnp:callSign*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:callSign* property contains the broadcast station call sign of the associated broadcast channel. This is typically used for live content or recorded content.

Example: “KGW”.

If the *upnp:callSign* property is supported and *upnp:class* = “*object.item.audioItem.audioBroadcast*” then the *upnp:radioCallSign* property MUST also be supported and MUST be set equal to the value of the *upnp:callSign* property.

Default Value: None.

B.9.3 *upnp:networkAffiliation*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:networkAffiliation* property contains the name of the broadcast network or distribution network associated with this content. This is typically used for live content or recorded content.

Examples: “NBC”, “CBS”, “BBC”.

Default Value: None.

B.9.4 *upnp:serviceProvider*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:serviceProvider* property contains the friendly name of the service provider of this content. This is typically used for live content or recorded content. Note that one service provider may provide multiple channel groups.

Examples: “CANAL+”, “Echostar”, “SkyLife”.

Default Value: None.

B.9.5 *upnp:price*

Namespace: upnp **Property Data Type:** xsd:float **Multi-Valued:** *YES*

Description: The *read-only* *upnp:price* property contains the price for a broadcast, series, program, movie, etc.

Default Value: None.

B.9.5.1 *upnp:price@currency*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read-only* *upnp:price@currency* property indicates the unit of currency used for the *upnp:price* property. The allowed values for this property MUST adhere to ISO 4217, "Type Currency Code List".

The *upnp:price@currency* property is REQUIRED if the *upnp:price* property is specified.

Default Value: N/A – The property is REQUIRED when the *upnp:price* property is present.

B.9.6 *upnp:payPerView*

Namespace: upnp **Property Data Type:** xsd:boolean **Multi-Valued:** *NO*

Description: The *read-only* *upnp:payPerView* property indicates whether the object represents pay-per-view content. When set to "*1*", the object is a pay-per-view object. When set to "*0*", the object is not a pay-per-view object.

Default Value: "*0*".

B.9.7 *upnp:epgProviderName*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:epgProviderName* property indicates the name of the Electronic Program Guide service provider.

Default Value: None.

B.9.8 *upnp:dateTimeRange*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read-only* *upnp:dateTimeRange* property indicates that all EPG items found in this container's subtree exist within this time range. The format of the *upnp:dateTimeRange* property MUST comply with the `date-time-range` syntax as defined in Annex D, "**(normative)**

EBNF Syntax Definitions".

Default Value: None.

B.9.8.1 *upnp:dateTimeRange@daylightSaving*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read-only* *upnp:dateTimeRange@daylightSaving* property indicates whether the time values used in the *upnp:dateTimeRange* property, are expressed using as a reference either Daylight Saving Time or Standard Time. This property is only applicable when the time values in the *upnp:dateTimeRange* property are expressed in local time. Whenever the time values in the *upnp:dateTimeRange* property are expressed in absolute time, the *upnp:dateTimeRange@daylightSaving* property MUST not be present on output and MUST be ignored on input.

B.9.8.1.1 allowedValueList for the *upnp:dateTimeRange@daylightSaving* Property

See Table B.6 in B.2.1.21.1, "allowedValueList for the *res@daylightSaving* Property".

B.10 Radio Broadcast Properties

Table B.15 — Radio Broadcast Properties Overview

Property Name	NS	Data Type	M-Val	Reference
upnp:radioCallSign	upnp	xsd:string	*NO*	Subclause B.10.1
upnp:radioStationID	upnp	xsd:string	*NO*	Subclause B.10.2
upnp:radioBand	upnp	xsd:string	*NO*	Subclause B.10.3

B.10.1 *upnp:radioCallSign*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:radioCallSign* property contains a radio station call sign, for example, “KSJO”.

Default Value: None.

B.10.2 *upnp:radioStationID*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:radioStationID* property contains some identification, for example, “107.7”, broadcast frequency of the radio station.

Default Value: None.

B.10.3 *upnp:radioBand*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:radioBand* property contains the radio station frequency band.

Default Value: None.

B.10.3.1 allowedValueList for the *upnp:radioBand* Property

Table B.16 — allowedValueList for the *upnp:radioBand* Property

Value	R/O	Description
“*AM*”	*O*	
“*FM*”	*O*	
“*Shortwave*”	*O*	
“*Internet*”	*O*	
“*Satellite*”	*O*	
Vendor-defined	*X*	

B.11 Video Broadcast Properties

Table B.17 — Video Broadcast Properties Overview

Property Name	NS	Data Type	M-Val	Reference
upnp:channelNr	upnp	xsd:int	*NO*	Subclause B.11.1
upnp:channelName	upnp	xsd:string	*NO*	Subclause B.11.2

Property Name	NS	Data Type	M-Val	Reference
upnp:scheduledStartTime	upnp	xsd:string	*NO*	Subclause B.11.3
upnp:scheduledStartTime @usage	upnp	xsd:string	*NO*	Subclause B.11.3.1
upnp:scheduledStartTime @daylightSaving	upnp	xsd:string	*NO*	Subclause B.11.3.2
upnp:scheduledEndTime	upnp	xsd:string	*NO*	Subclause B.11.4
upnp:scheduledEndTime @daylightSaving	upnp	xsd:string	*NO*	Subclause B.11.4.1
upnp:scheduledDuration	upnp	xsd:string	*NO*	Subclause B.11.5

B.11.1 *upnp:channelNr*

Namespace: upnp **Property Data Type:** xsd:int **Multi-Valued:** *NO*

Description: The *upnp:channelNr* property contains the number of the associated broadcast channel. This is typically used for live content or recorded content.

If there exists a *upnp:channelID* property with its dependent property *upnp:channelID@type* property set to "*DIGITAL*", then the *upnp:channelNr* property MUST be set equal to the major channel number from that *upnp:channelID* property.

Else, if there exists a *upnp:channelID* property with its dependent *upnp:channelID@type* property set to "*ANALOG*", then the *upnp:channelNr* property MUST be set equal to the value of that *upnp:channelID* property.

Else, the *upnp:channelNr* property MUST NOT exist.

Default Value: None.

B.11.2 *upnp:channelName*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:channelName* property contains the user-friendly name of the associated broadcast channel. This is typically used for live or recorded content.

Default Value: None.

B.11.3 *upnp:scheduledStartTime*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:scheduledStartTime* property is used to indicate the start time of a scheduled program, intended for use by tuners. The format MUST be compliant to [ISO 8601] and SHOULD be compliant to [RFC 3339].

Default Value: None.

B.11.3.1 *upnp:scheduledStartTime@usage*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:scheduledStartTime@usage* property is used to indicate whether the *upnp:scheduledStartTime* and *upnp:scheduledEndTime* properties contain the start and end times of a scheduled program event, or contain the start and end times of the time window within which on-demand content is available for consumption.

Default Value: "*SCHEDULED_PROGRAM*".

B.11.3.1.1 allowedValueList for the *upnp:scheduledStartTime@usage* Property

Table B.18 — allowedValueList for the *upnp:scheduledStartTime@usage* Property

Value	R/O	Description
"*SCHEDULED_PROGRAM*"	*O*	the *upnp:scheduledStartTime* and *upnp:scheduledEndTime* properties contain the start and end times of a scheduled program event.
"*ON_DEMAND*"	*O*	the *upnp:scheduledStartTime* and *upnp:scheduledEndTime* properties contain the start and end times of the time window within which on-demand content is available for consumption.

B.11.3.2 *upnp:scheduledStartTime@daylightSaving*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:scheduledStartTime@daylightSaving* property indicates whether the time value used in the *upnp:scheduledStartTime* property is expressed using as a reference either Daylight Saving Time or Standard Time. This property is only applicable when the time value in the *upnp:scheduledStartTime* property is expressed in local time. Whenever the time value in the *upnp:scheduledStartTime* property is expressed in absolute time, the *upnp:scheduledStartTime@daylightSaving* property MUST not be present on output and MUST be ignored on input.

Default Value: "*UNKNOWN*".

B.11.3.2.1 allowedValueList for the *upnp:scheduledStartTime@daylightSaving* Property

See Table B.6 in B.2.1.21.1, "allowedValueList for the *res@daylightSaving* Property".

B.11.4 *upnp:scheduledEndTime*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:scheduledEndTime* property is used to indicate the end time of a scheduled program, intended for use by tuners. The format MUST be compliant to [[ISO 8601] and SHOULD be compliant to [RFC 3339].

Default Value: None.

B.11.4.1 *upnp:scheduledEndTime@daylightSaving*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:scheduledEndTime@daylightSaving* property indicates whether the time value used in the *upnp:scheduledEndTime* property is expressed using as a reference either Daylight Saving Time or Standard Time. This property is only applicable when the time value in the *upnp:scheduledEndTime* property is expressed in local time. Whenever the time value in the *upnp:scheduledEndTime* property is expressed in absolute time, the *upnp:scheduledEndTime@daylightSaving* property MUST not be present on output and MUST be ignored on input.

Default Value: "*UNKNOWN*".

B.11.4.1.1 allowedValueList for the *upnp:scheduledEndTime@daylightSaving* Property

See Table B.6 in B.2.1.21.1, “allowedValueList for the *res@daylightSaving* Property”.

B.11.5 *upnp:scheduledDuration*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:scheduledDuration* property indicates the scheduled duration of a scheduled program. The `duration` format syntax of the *upnp:scheduledDuration* property is defined in Annex D, “**(normative)**

EBNF Syntax Definitions”.

Examples: “P01:30:00” (one hour and thirty minutes), “P2D01:15:00” (two-day and seventy five minutes).

It is highly RECOMMENDED that whenever the *upnp:scheduledDurationTime* property is present, the *upnp:scheduledStartTime* and *upnp:scheduledEndTime* properties are also provided.

Default Value: None.

B.12 Physical Tuner Status-related Properties

Table B.19 — Physical Tuner Status-related Properties Overview

Property Name	NS	Data Type	M-Val	Reference
upnp:signalStrength	upnp	xsd:int	*NO*	Subclause B.12.1
upnp:signalLocked	upnp	xsd:boolean	*NO*	Subclause B.12.2
upnp:tuned	upnp	xsd:boolean	*NO*	Subclause B.12.3

B.12.1 *upnp:signalStrength*

Namespace: upnp **Property Data Type:** xsd:int **Multi-Valued:** *NO*

Description: The *read-only* *upnp:signalStrength* property contains the relative strength of the signal that is used to retrieve the content for the item. A value of 0 indicates “no signal detected”. A value of 100 indicates “best possible” signal strength. A value of -1 indicates that the signal strength is currently unknown. Values less than -1 or greater than 100 are reserved for future use and MUST be treated as -1.

A change in the value of this property does not result in a change in the *SystemUpdateID* state variable or the corresponding *ContainerUpdateIDs* state variable. Therefore, a change to the this property does not constitute an *object modification*.

Default Value: None.

B.12.2 *upnp:signalLocked*

Namespace: upnp **Property Data Type:** xsd:boolean **Multi-Valued:** *NO*

Description: The *read-only* *upnp:signalLocked* property indicates whether the signal strength is sufficiently strong to enable the hardware to lock onto the signal at the current target

frequency. When set to "*1*", the signal strength is high enough for the hardware to lock onto it. When set to "*0*", the signal strength is too low for the hardware to lock onto it.

A change in the value of this property does not result in a change in the *SystemUpdateID* state variable or the corresponding *ContainerUpdateIDs* state variable. Therefore, a change to the this property does not constitute an *object modification*.

Default Value: None.

B.12.3 *upnp:tuned*

Namespace: upnp **Property Data Type:** xsd:boolean **Multi-Valued:** *NO*

Description: The *read-only* *upnp:tuned* property indicates whether a hardware resource is currently tuned to retrieve the content represented by this item. When set to "*1*", there is a hardware resource currently tuned to this item. When set to "*0*", there is no hardware resource currently tuned.

Default Value: None.

B.13 Bookmark-related Properties

Table B.20 — Bookmark-related Properties Overview

Property Name	NS	Data Type	M-Val	Reference
@neverPlayable	DIDL-Lite	xsd:boolean	*NO*	Subclause B.13.1
upnp:bookmarkID	upnp	xsd:string	*YES*	Subclause B.13.2
upnp:bookmarkedObjectID	upnp	xsd:string	*NO*	Subclause B.13.3
upnp:deviceUDN	upnp	xsd:string	*NO*	Subclause B.13.4
upnp:deviceUDN@serviceType	upnp	xsd:string	*NO*	Subclause B.13.4.1
upnp:deviceUDN@serviceId	upnp	xsd:string	*NO*	Subclause B.13.4.2
upnp:stateVariableCollection	upnp	xsd:string	*YES*	Subclause B.13.5
upnp:stateVariableCollection@ serviceName	upnp	xsd:string	*NO*	Subclause B.13.5.1
upnp:stateVariableCollection@ rcsInstanceType	upnp	xsd:string	*NO*	Subclause B.13.5.2

B.13.1 *@neverPlayable*

Namespace: upnp **Property Data Type:** xsd:boolean **Multi-Valued:** *NO*

Description: The *@neverPlayable* property indicates whether an item or container will *ever* have normal playable content. A value of "*1*" indicates that the associated item or container will never have normal playable content. Furthermore, for a container, the complete subtree underneath the container will also never have normal playable content. A value of "*0*" indicates that the item or subtree MAY contain playable content.

The value of this property MUST be static.

Default Value: "*0*".

B.13.2 *upnp:bookmarkID*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *read/write* *upnp:bookmarkID* property contains the object ID of a bookmark item that is associated with this content item and that marks a specific location within its content.

Default Value: None.

B.13.3 *upnp:bookmarkedObjectID*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read/write* *upnp:bookmarkedObjectID* property contains the object ID of the content item that is bookmarked by this bookmark.

Default Value: None.

B.13.4 *upnp:deviceUDN*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read/write* *upnp:deviceUDN* property contains the UDN of the device whose state information is captured in the values of the *upnp:stateVariableCollection* properties within the same bookmark item.

Default Value: None.

B.13.4.1 *upnp:deviceUDN@serviceType*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read/write* *upnp:deviceUDN@serviceType* property contains the service type of the device whose UDN is stored in the associated *upnp:deviceUDN* property. Note that the service type includes the name and version number, such as “AVTransport:1”.

The *upnp:deviceUDN@serviceType* property is REQUIRED if the *upnp:deviceUDN* property is specified.

Default Value: N/A – The property is REQUIRED when the *upnp:deviceUDN* property is present.

B.13.4.2 *upnp:deviceUDN@serviceId*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read/write* *upnp:deviceUDN@serviceId* property contains the serviceId of the device whose UDN is stored in the associated *upnp:deviceUDN* property.

The *upnp:deviceUDN@serviceId* property is REQUIRED if the *upnp:deviceUDN* property is specified.

Default Value: N/A – The property is REQUIRED when the *upnp:deviceUDN* property is present.

B.13.5 *upnp:stateVariableCollection*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *read/write* *upnp:stateVariableCollection* property holds a *stateVariableValuePairs XML Document* which encapsulates the collected state variables and their values. See [AVS-XSD].

Example:

The following illustrates a typical example of the *upnp:stateVariableCollection* property content:

```
<?xml version="1.0" encoding="UTF-8"?>
<stateVariableValuePairs
 xmlns="urn:schemas-upnp-org:av:avs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:avs
    http://www.upnp.org/schemas/av/avs.xsd">
   <stateVariable variableName="CurrentPlayMode">
      NORMAL
   </stateVariable>
   <stateVariable variableName="CurrentTrack">
      3
   </stateVariable>
   <!-- More state variable value pairs can be inserted here -->
</stateVariableValuePairs>
```

Default Value: None.

B.13.5.1 *upnp:stateVariableCollection@serviceName*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read/write* *upnp:stateVariableCollection@serviceName* property identifies from which service the state variables were retrieved.

The *upnp:stateVariableCollection@serviceName* property is REQUIRED if the *upnp:stateVariableCollection* property is specified.

Default Value: N/A – The property is REQUIRED when the *upnp:stateVariableCollection* property is present.

B.13.5.2 *upnp:stateVariableCollection@rcsInstanceType*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *read/write* *upnp:stateVariableCollection@rcsInstanceType* property specifies whether the RenderingControl service instance is pre-mix or post-mix. It MUST be specified if the state variable collection originates from a RenderingControl service.

Default Value: N/A – The property is REQUIRED when the collection originates from a RenderingControl service.

B.13.5.2.1 allowedValueList for the *upnp:stateVariableCollection@rcsInstanceType* Property

Table B.21 — allowedValueList for the upnp:stateVariableCollection@rcsInstanceType Property

Value	R/O	Description
"*pre-mix*"	*R*	

Value	R/O	Description
"*post-mix*"	*R*	

B.14 Miscellaneous Properties

Table B.22 — Miscellaneous Properties Overview

Property Name	NS	Data Type	M-Val	Reference
upnp:DVDRegionCode	upnp	xsd:int	*NO*	Subclause B.14.1
upnp:originalTrackNumber	upnp	xsd:int	*NO*	Subclause B.14.2
upnp:toc	upnp	xsd:string	*NO*	Subclause B.14.3
upnp:userAnnotation	upnp	xsd:string	*YES*	Subclause B.14.4
desc	DIDL-Lite	xsd:string	*YES*	Subclause B.14.5
desc@nameSpace	DIDL-Lite	xsd:string	*NO*	Subclause B.14.5.1

B.14.1 *upnp:DVDRegionCode*

Namespace: upnp **Property Data Type:** xsd:int **Multi-Valued:** *NO*

Description: The *upnp:DVDRegionCode* property contains the region code of the DVD disc.

Default Value: None.

B.14.2 *upnp:originalTrackNumber*

Namespace: upnp **Property Data Type:** xsd:int **Multi-Valued:** *NO*

Description: The *upnp:originalTrackNumber* property contains the original track number on an audio CD or other medium.

Default Value: None.

B.14.3 *upnp:toc*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *NO*

Description: The *upnp:toc* property contains the table of contents of the object.

Default Value: None.

B.14.4 *upnp:userAnnotation*

Namespace: upnp **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: The *read/write* *upnp:userAnnotation* property is a general-purpose property where a user can annotate an object with some user-specific information.

Default Value: None.

B.14.5 *desc*

Namespace: DIDL-Lite **Property Data Type:** xsd:string **Multi-Valued:** *YES*

Description: Vendors MAY extend DIDL-Lite metadata by placing blocks of vendor-specific metadata into *desc* properties. The *@nameSpace* property identifies the namespace of the