# ContentDirectory:4 Service

**For UPnP Version 1.0**
**Status: Standardized DCP (SDCP)**
**Date: December 31, 2010**
**Service Template Version 1.01**

| Authors | Company |
|---|---|
| Gary Langille | Echostar |
| Raj Bopardikar | Intel |
| John Ritchie | Intel |
| Jaewook Lee | LGE |
| Anders Klemets | Microsoft |
| Keith Miller (Chair) | Nokia |
| Vlad Stirbu | Nokia |
| Julius Szakolczay | Panasonic |
| Wouter van der Beek | Philips |
| Jeffrey Kang | Philips |
| Geert Knapen | Philips |
| Selvakkumar Palaniyappan | Philips |
| Russell Berkoff | Pioneer |
| Russell Berkoff (Vice-Chair) | Samsung Electronics |
| Wonseok Kwon | Samsung Electronics |

| Authors | Company |
|---|---|
| SJae Oh | Samsung Electronics |
| Mahfuzur Raman | Samsung Electronics |
| Richard Bardini | Sony |

***Note: The UPnP Forum in no way guarantees the accuracy or completeness of this author list and in no way implies any rights for or support from those members listed. This list is not the specifications' contributor list that is kept on the UPnP Forum's website.**

## Contents

## List of Tables

# List of Figures

# 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 Section 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

   1. not escaped in CSV lists
   2. not escaped in search strings when it appears as the start or end delimiter of a property value
   3. represented as "\"" in search strings when it appears as a character that is part of the property value

### 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 section 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 ({a,b,c},{*x*, *y*, *z*}), where a, b, c, *x*, *y* and *z* are the types of the individual values in the subsequence and the subsequences 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 Section 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* supersede 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 ContentDirectory 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 Section 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 Namespace Prefix | Namespace Name | Namespace Description | Normative Definition Document Reference |
|---|---|---|---|
| *AV Working Committee defined namespaces* | | | |
| atrs | urn:schemas-upnp-org:av:AllowedTransformSettings | *AllowedTransformSettings* and *AllowedDefaultTransformSettings* state variables for RenderingControl | [RCS] |
| av | urn:schemas-upnp-org:av:av | Common data types for use in AV schemas | [AV-XSD] |
| avdt | urn:schemas-upnp-org:av:avdt | Datastructure Template | [AVDT] |
| avs | urn:schemas-upnp-org:av:avs | Common structures for use in AV schemas | [AVS-XSD] |
| avt-event | urn:schemas-upnp-org:metadata-1-0/AVT/ | Evented *LastChange* state variable for AVTransport | [AVT] |
| cds-event | urn:schemas-upnp-org:av:cds-event | Evented *LastChange* state variable for ContentDirectory | [CDS] |
| cm-dciu | urn:schemas-upnp-org:av:cm-deviceClockInfoUpdates | Evented *DeviceClockInfoUpdates* state variable for ConnectionManager | [CM] |
| cm-ftrlst | urn:schemas-upnp-org:av:cm-featureList | *FeatureList* state variable for ConnectionManager | [CM] |
| didl-lite | urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/ | Structure and metadata for ContentDirectory | [CDS] |
| dmo | urn:schemas-upnp.org:av:dmo | Evented *DeviceMode* state variable for ContentDirectory | [CDS] |
| dmor | urn:schemas-upnp.org:av:dmor | *A_ARG_TYPE_DeviceModeRequest* state variable for ContentDirectory | [CDS] |
| dmos | urn:schemas-upnp.org:av:dmos | *DeviceModeStatus* state variable for ContentDirectory | [CDS] |
| pi | urn:schemas-upnp.org:av:pi | *PermissionsInfo* state variable for ContentDirectory | [CDS] |
| rcs-event | urn:schemas-upnp-org:metadata-1-0/RCS/ | Evented *LastChange* state variable for RenderingControl | [RCS] |
| rii | urn:schemas-upnp-org:av:rii | *A_ARG_TYPE_RenderingInfoList* state variable for ConnectionManager | [CM] |
| rpl | urn:schemas-upnp-org:av:rpl | *A_ARG_TYPE_PlaylistInfo* state variable for AVTransport | [AVT] |
| 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] |
| trs | urn:schemas-upnp-org:av:TransformSettings | *TransformSettings* and *DefaultTransformSettings* state variables for RenderingControl | [RCS] |
| 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]<br>[XML SCHEMA-2] |
| xsi | http://www.w3.org/2001/XMLSchema-instance | XML Schema Instance Document schema | Sections 2.6 & 3.2.7 of [XML SCHEMA-1] |

| Standard Name-space Prefix | Namespace Name | Namespace Description | Normative Definition Document Reference |
|---|---|---|---|
| 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]<br>• Form 1, Form 2, Form3 | Valid Root Element(s) | Schema Reference |
|---|---|---|---|
| | | *AV Working Committee Defined Namespaces* | |
| atrs | • AllowedTransformSettings-*vn-yyyymmdd*.xsd<br>• AllowedTransformSettings-*vn*.xsd<br>• AllowedTransformSettings.xsd | `<TransformList>` | [ATRS-XSD] |
| av | • av-*vn-yyyymmdd*.xsd<br>• av-*vn*.xsd<br>• av.xsd | *n/a* | [AV-XSD] |
| avdt | • avdt-*vn-yyyymmdd*.xsd<br>• avdt-*vn*.xsd<br>• avdt.xsd | `<AVDT>` | [AVDT] |
| avs | • avs-*vn-yyyymmdd*.xsd<br>• avs-*vn*.xsd<br>• avs.xsd | `<Capabilities>`<br>`<Features>`<br>`<stateVariableValuePairs>` | [AVS-XSD] |
| avt-event | • avt-event-*vn-yyyymmdd*.xsd<br>• avt-event-*vn*.xsd<br>• avt-event.xsd | `<Event>` | [AVT-EVENT-XSD] |
| cds-event | • cds-event-*vn-yyyymmdd*.xsd<br>• cds-event-*vn*.xsd<br>• cds-event.xsd | `<StateEvent>` | [CDS-EVENT-XSD] |
| cm-dciu | • cm-deviceClockInfoUpdates-*vn-yyyymmdd*.xsd<br>• cm-deviceClockInfoUpdates -*vn*.xsd<br>• cm-deviceClockInfoUpdates.xsd | `<DeviceClockInfoUpdates>` | [CM-DCIU-XSD] |
| cm-ftrlst | • cm-featureList-*vn-yyyymmdd*.xsd<br>• cm-featureList-*vn*.xsd<br>• cm-featureList.xsd | `<Features>` | [CM-FTRLST-XSD] |
| didl-lite | • didl-lite-*vn-yyyymmdd*.xsd<br>• didl-lite-*vn*.xsd<br>• didl-lite.xsd | `<DIDL-Lite>` | [DIDL-LITE-XSD] |

| **Standard Name-space Prefix** | **Relative URI and File Name**[1]<br>**• Form 1, Form 2, Form3** | **Valid Root Element(s)** | **Schema Reference** |
|---|---|---|---|
| dmo | • dmo-*vn*-*yyyymmdd*.xsd<br>• dmo-*vn*.xsd<br>• dmo.xsd | `<DeviceMode>` | [DMO-XSD] |
| dmor | • dmor-*vn*-*yyyymmdd*.xsd<br>• dmor-*vn*.xsd<br>• dmor.xsd | `<DeviceModeRequest>` | [DMOR-XSD] |
| dmos | • dmos-*vn*-*yyyymmdd*.xsd<br>• dmos-*vn*.xsd<br>• dmos.xsd | `<DeviceModeStatus>` | [DMOS-XSD] |
| pi | • pi-*vn*-*yyyymmdd*.xsd<br>• pi-*vn*.xsd<br>• pi.xsd | `<PermissionsInfo>` | [PI-XSD] |
| rcs-event | • rcs-event-*vn*-*yyyymmdd*.xsd<br>• rcs-event-*vn*.xsd<br>• rcs-event.xsd | `<Event>` | [RCS-EVENT-XSD] |
| rii | • rii-*vn*-*yyyymmdd*.xsd<br>• rii-*vn*.xsd<br>• rii.xsd | `<rendererInfo>` | [RII-XSD] |
| rpl | • rpl-*vn*-*yyyymmdd*.xsd<br>• rpl-*vn*.xsd<br>• rpl.xsd | `<PlaylistInfo>` | [RPL-XSD] |
| trs | • TransformSettings-*vn*-*yyyymmdd*.xsd<br>• TransformSettings-*vn*.xsd<br>• TransformSettings.xsd | `<TransformSettings>` | [TRS-XSD] |
| srs | • srs-*vn*-*yyyymmdd*.xsd<br>• srs-*vn*.xsd<br>• srs.xsd | `<srs>` | [SRS-XSD] |
| srs-event | • srs-event-*vn*-*yyyymmdd*.xsd<br>• srs-event-*vn*.xsd<br>• srs-event.xsd | `<StateEvent>` | [SRS-EVENT-XSD] |
| upnp | • upnp-*vn*-*yyyymmdd*.xsd<br>• upnp-*vn*.xsd<br>• 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* |
| 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 Section 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.

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

[ATRS-XSD] – *XML Schema for RenderingControl AllowedTransformSettings*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/AllowedTransformSettings-v1-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/AllowedTransformSettings.xsd.

[AVARCH] – *AVArchitecture:2*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v2-20101231.pdf.

Latest version available at: http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v2.pdf.

[AVDT] – *AV Datastructure Template:1*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/specs/av/UPnP-av-AVDataStructureTemplate-v1-20101231.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-AVDataStructureTemplate-v1.pdf.

[AVDT-XSD] – *XML Schema for UPnP AV Datastructure Template*, 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.xsd.

[AV-XSD] – *XML Schema for UPnP AV Common XML Data Types*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/av-v3-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/av.xsd.

[AVS-XSD] – *XML Schema for UPnP AV Common XML Structures*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/avs-v3-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/avs.xsd.

[AVT] – *AVTransport:3*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/specs/av/UPnP-av-AVTransport-v3-Service-20101231.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-AVTransport-v3-Service.pdf.

[AVT-EVENT-XSD] – *XML Schema for AVTransport 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.xsd.

[CDS] – *ContentDirectory:4*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v4-Service-20101231.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v4-Service.pdf.

[CDS-EVENT-XSD] – *XML Schema for ContentDirectory 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.xsd.

[CM] – *ConnectionManager:3*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v3-Service-20101231.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v3-Service.pdf.

[CM-DCIU-XSD] – *XML Schema for ConnectionManager DeviceClockInfoUpdates*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/cm-deviceClockInfoUpdates-v1-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/cm-deviceClockInfoUpdates.xsd.

[CM-FTRLST-XSD] – *XML Schema for ConnectionManager Features*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/cm-featureList-v1-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/cm-featureList.xsd.

[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, October 15, 2008.
Available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20081015.pdf.
Latest version available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.

[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 Structure and Metadata (DIDL-Lite)*, UPnP Forum, December 31, 2010.

Available at: http://www.upnp.org/schemas/av/didl-lite-v3-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/didl-lite.xsd.

[DMO-XSD] – *XML Schema for ContentDirectory DeviceMode*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/dmo-v1-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/dmo.xsd.

[DMOR-XSD] – *XML Schema for ContentDirectory DeviceModeRequest*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/dmor-v1-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/dmor.xsd.

[DMOS-XSD] – *XML Schema for ContentDirectory DeviceModeStatus*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/dmos-v1-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/dmos.xsd.

[DP] – *DeviceProtection:1*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service-20110224.pdf.
Latest version available at: http://www.upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service.pdf.

[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.

[IEEE-802.1AS] – *IEEE P802.1AS™ (Draft 7.0) - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, Institute of Electrical and Electronics Engineers, March 23, 2010.
Available at: http://www.ieee802.org/1/pages/802.1as.html.

[IEEE-1733] – *IEEE-P1733™ (Draft 2.2) – Audio Video Bridge Layer 3 Transport Protocol*, International Institute of Electrical and Electronics Engineers, April 20, 2009.
Available at: http://grouper.ieee.org/groups/1733.

[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:3*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/specs/av/UPnP-av-MediaRenderer-v3-Device-20101231.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-AV-MediaRenderer-v3-Device.pdf.

[MS] – *MediaServer:4*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/specs/av/UPnP-av-MediaServer-v4-Device-20101231.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-AV-MediaServer-v4-Device.pdf.

[PI-XSD] – *XML Schema for ContentDirectory PermissionsInfo*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/pi-v1-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/pi.xsd.

[RCS] – *RenderingControl:3*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v3-Service-20101231.pdf.

Latest version available at: http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v3-Service.pdf.

[RCS-EVENT-XSD] –*XML Schema for RenderingControl LastChange Eventing*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/rcs-event-v3-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/rcs-event.xsd.

[RFC 1305] – *IETF RFC 1305, Network Time Protocol (Version 3) Specification, Implementation and Analysis*, David L. Mills, March 1992.
Available at: http://www.ietf.org/rfc/rfc1305.txt.

[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 2030] – *IETF RFC 2030, Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OS*, D Mills, October 1996.
Available at: http://www.ietf.org/rfc/rfc2030.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.

[RII-XSD] – *XML Schema for ConnectionManager RendererInfo*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/rii-v1-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/rii.xsd.

[RPL-XSD] – *XML Schema for AVTransport PlaylistInfo*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/rpl-v1-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/rpl.xsd.

[RTP] – *IETF RFC 3550, RTP: A Transport Protocol for Real-Time Applications,* H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, July 2003.
Available at: http://www.ietf.org/rfc/rfc3550.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, December 31, 2010.
Available at: http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v2-Service-20101231.pdf.
Latest version available at: http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v2-Service.pdf.

[SRS-XSD] – *XML Schema for ScheduledRecording 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.xsd.

[SRS-EVENT-XSD] – *XML Schema for ScheduledRecording 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.xsd.

[TRS-XSD] – *XML Schema for RenderingControl TransformSettings*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/TransformSettings-v1-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/TransformSettings.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 Metadata*, UPnP Forum, December 31, 2010.
Available at: http://www.upnp.org/schemas/av/upnp-v4-20101231.xsd.
Latest version available at: http://www.upnp.org/schemas/av/upnp.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.

[UUID] – *IETF RFC 4122, A Universally Unique Identifier (UUID) URN Namespace*, P. Leach, Microsoft, M. Mealling, Refactored Networks LLC, R. Salz, DataPower Technology, Inc., July 2005.
Available at: http://www.ietf.org/rfc/rfc4122.txt.

[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:4**

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 Section 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 Section 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 Section 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 Section 2.5.6, "*Browse()*" and Section 2.5.8 "*Search()*" for examples. See Section 2.3.15, "*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 Section 2.2.20, "*CDS Properties*" 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 Section 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 Section 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 Section 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 Section 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 Section 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 section (Section 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 Section 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 Section 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 Appendix 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 Section 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 Section 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 Section 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 Section 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 Section 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 Section 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 Appendix B.19.1, "*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 Section 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.)

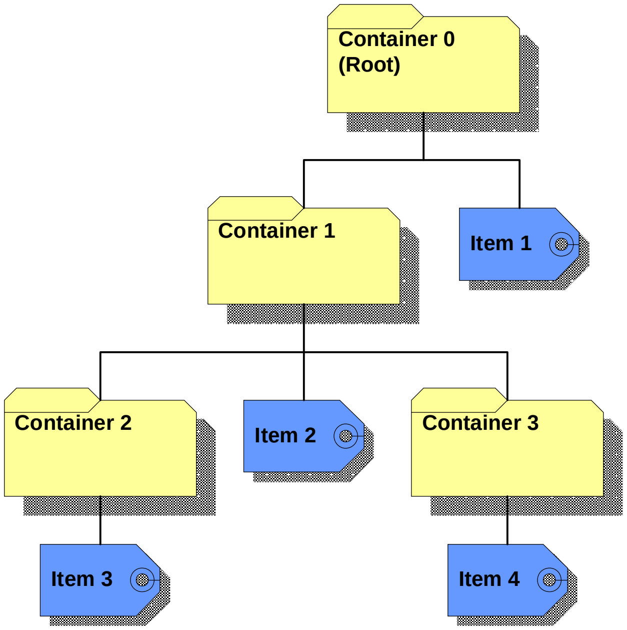


**Figure 1: ContentDirectory Service Object Organization.**

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 Appendix B.1.1, "*@id*" and Appendix 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 Section 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:

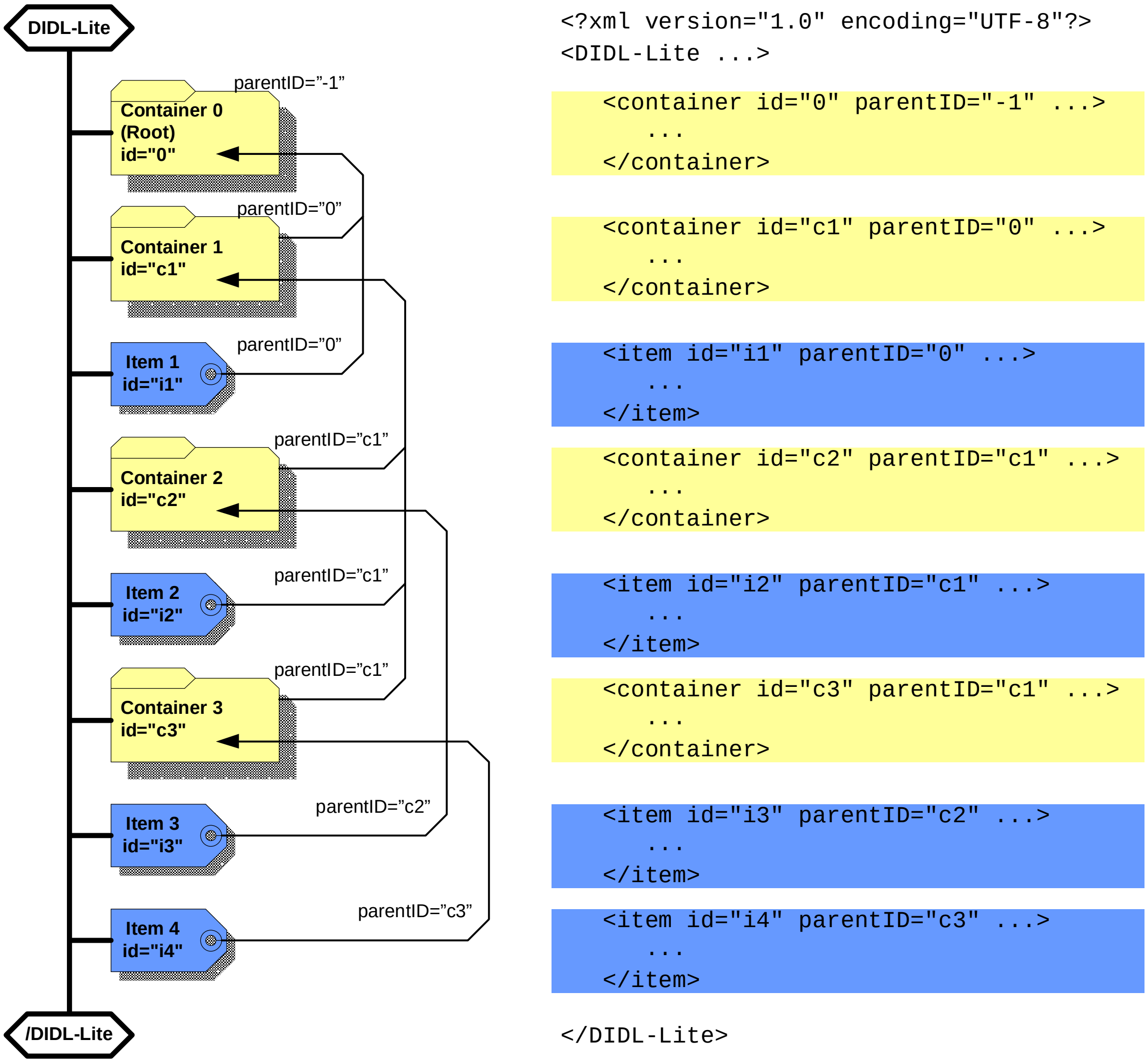
"`<DIDL-Lite …>…</DIDL-Lite>`"
or

"`<?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 *CDS Properties*

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 Appendix B, “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<br>(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 Section 2.2.5 "*Object Modification*" and Section 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 Appendix E, "*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 Section 2.2.20, "*CDS Properties*" 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 Section 2.6.16, "Foreign Metadata", Appendix B.23, "Foreign Metadata-related Properties" and Appendix 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.

### 2.2.25 Device Protection Option

The ContentDirectory service does not include any mechanism to restrict access to service actions/content based on *Roles*, *Control Point Identities* or *User Identities*. Due to this openness, content and services on MediaServers can be accessed by all control points/users in the UPnP network. This leads to violation of content privacy in MediaServers. Implementation of the ContentDirectory service *CONTENT_PROTECTION feature* addresses these limitations.

When a DeviceProtection service [DP] is available on a device implementing a ContentDirectory service then a *CONTENT_PROTECTION feature* (see Appendix E.10 and Appendix F) can also be implemented.When implemented, specific actions and specific content may be restricted from specific control points. In other words, control points which are now identifiable to the service can be assigned specific permissions, that is the right to invoke an action on a per action basis (see *Action level access*) or per object (*container* or *item*) basis (see *Object level access*). In general, control points can be identified on an unrecognized (Public) or recognized (*Control Point Identity* or *User Identity*) basis. In each case, one or more *Role(s)* can be assigned to a *Control Point Identity* or *User Identity*. For control points that do not implement DeviceProtection or remain unregistered, a generic *Role* of (*Public*) is automatically assigned. Additional description of these and related terms can be found in the DeviceProtection service. Familiarization with these terms is recommended.

#### 2.2.25.1 Device Protection Terms

- ***Device Identity***: A *Device Identity* is the identity of a UPnP Device that implements the DeviceProtection service [DP]. A *Device Identity* is a UUID value derived from a hash of the Device's X.509 server peer certificate (not the CA certificate), in accordance with the algorithm given in Section 4.3 of [UUID]. See DeviceProtection specification [DP] for detailed information regarding deriving *Device Identity* UUIDs. The same UUID value MAY be used for both the *Device Identity* and the normal UPnP Device UUID, but this is NOT required.

- ***Control Point Identity***: A *Control Point Identity* (also referred to as its certificate Identity) is a UUID value derived from a hash of the control point's X.509 client peer certificate (not the CA certificate), in accordance with the algorithm given in Section 4.3 of [UUID]. See the DeviceProtection [DP] specification for additional information.

- ***User Identity***: The identity of a human user operating a control point. *User Identities* consist of Username/Password pairs.
- ***Role***: A name used to identify a set of access rights. When a *Role* is assigned to a *Control Point Identity* or *User Identity*, that identity is granted access rights associated with the *Role*. *Role* names defined by UPnP Working Committes MUST be prefixed with the working committee moniker followed by a colon (for example, "AV:"). *Role* names defined by the DeviceProtection service do not include a prefix. Each *Role* name MUST have length no longer than 64 characters, including the prefix (if any). Note the UPnP DeviceProtection service defines three *Roles: Public*, *Basic* and *Admin*.
- ***Permission***: indicates the right to perform an action (*Action level access*) or an action on a specific object (*Object level access*).
- ***Introduction Protocol***: An Introduction Protocol is a protocol designed to support an initial exchange of cryptographic data that can be used subsequently for secure communications.
- ***Restricted DIDL-Lite View***: The Restricted version of the DIDL-Lite View contains only the metadata from the previously defined DIDL-LITE view that would be returned for each object as a response to the *Browse()* action invoked by a control point with a specific *Role(s)* when the *CONTENT_PROTECTION feature* is implemented (See Appendix F.2.3).
- ***Action level access*** indicates that a particular control point has been authorized to invoke a particular action on a device and service implementing the *CONTENT_PROTECTION feature*.
- ***Object level access*** indicates that a particular control point with *Action level access* to an action also has been authorized to invoke the action on a particular object.
- ***Restrictable action*** indicates an action whose invocation can be blocked according to the presence or absence of a specific *Role* attached to a *Control Point Identity* or *User Identity*.
- ***Non-Restrictable action*** indicates an action whose invocation cannot be blocked regardless of the presence or absence of a specific *Role* attached to a *Control Point Identity* or *User Identity*.
- ***Owner/Own***: A *Role* included in the *upnp:objectOwner::role* property of an object is an ***owner*** of that object and is said to ***own*** that object.
- ***Non-Owner***: A *Role* not included in the *upnp:objectOwner::role* property of an object.

### 2.2.26 Device Mode Option

The ContentDirectory service can be configured for special device modes that support certain prioritized modes. The modes currently defined are *ActionBurst mode* and *ExclusiveOwnership mode*.

- ***ActionBurst:*** A burst of action invocations expected in an intense or regular manner exceeding typical usage, such as, those required for synchronizing the content between two devices.
- ***ActionBurst mode***: When a device (ContentDirectory service) announces support for *ActionBurst mode*, it means that the control point can request the device to be ready for an *ActionBurst*. This prior notice helps the device implementation ready itself for an upcoming *ActionBurst* by, for example, reserving resources, pre-allocating memory, or restricting non-related actions, services, or applications. The control point can also request an extension to or cancellation of the *ActionBurst mode*.
- ***ExclusiveOwnership mode***: When a device announces support for *ExclusiveOwnership mode*, it means that the control point can request complete control over the device (ContentDirectory service) for a requested length of time, for example, if a control point wants to make some updates to the ContentDirectory service and doesn't want that process to be interrupted due to action-calls from other control points. The control point can also request extension to or cancellation of the *ExclusiveOwnership mode*.

## 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**)<br>See Section 2.3.2 | | |
| *SortCapabilities* | *R* | **string** | CSV (**string**)<br>See Section 2.3.3 | | |
| *SortExtensionCapabilities* | *CR*[2] | **string** | CSV (**string**)<br>See Section 2.3.4 | | |
| *SystemUpdateID* | *R* | **ui4** | See Section 2.3.5 | | |
| *ContainerUpdateIDs* | *O* | **string** | CSV ({**string**,**ui4**})<br>See Section 2.3.6 | | |
| *ServiceResetToken* | *R* | **string** | See Section 2.3.7 | | |
| *LastChange* | *CR*[2] | string | *LastChange XML Document*<br>See Section 2.3.8 | | |
| *TransferIDs* | *CR*[2] | **string** | CSV (**ui4**)<br>See Section 2.3.9 | | |
| *FeatureList* | *R* | **string** | *Features XML Document*<br>See Section 2.3.10 | | |
| *DeviceMode* | *CR*[2] | **string** | See Section 2.3.11 | | |
| *DeviceModeStatus* | *CR*[2] | **string** | See Section 2.3.12 | | |
| *PermissionsInfo* | *CR*[2] | **string** | See Section 2.3.13 | | |
| *A_ARG_TYPE_ObjectID* | *R* | **string** | See Section 2.3.11 | | |
| *A_ARG_TYPE_Result* | *R* | **string** | See Section 2.3.15 | | |
| *A_ARG_TYPE_SearchCriteria* | *CR*[2] | **string** | See Section 2.3.16 | | |
| *A_ARG_TYPE_BrowseFlag* | *R* | **string** | BrowseMetadata,<br>BrowseDirectChildren | | |
| *A_ARG_TYPE_Filter* | *R* | **string** | CSV (**string**)<br>See Section 2.3.18 | | |
| *A_ARG_TYPE_SortCriteria* | *R* | **string** | CSV (**string**)<br>See Section 2.3.19 | | |

| Variable Name | R/O[1] | Data Type | Allowed Value | Default Value | Eng. Units |
|---|---|---|---|---|---|
| *A_ARG_TYPE_Index* | *R* | **ui4** | See Section 2.3.20 | | |
| *A_ARG_TYPE_Count* | *R* | **ui4** | See Section 2.3.21 | | |
| *A_ARG_TYPE_UpdateID* | *R* | **ui4** | See Section 2.3.22 | | |
| *A_ARG_Type_TransferID* | *CR*[2] | **ui4** | See Section 2.3.23 | | |
| *A_ARG_Type_TransferStatus* | *CR*[2] | **string** | See Section 2.3.24 | | |
| *A_ARG_Type_TransferLength* | *CR*[2] | **string** | See Section 2.3.25 | | |
| *A_ARG_Type_TransferTotal* | *CR*[2] | **string** | See Section 2.3.26 | | |
| *A_ARG_TYPE_TagValueList* | *CR*[2] | **string** | CSV (**string**)<br>See Section 2.3.27 | | |
| *A_ARG_TYPE_URI* | *CR*[2] | **uri** | See Section 2.3.28 | | |
| *A_ARG_TYPE_CDSView* | *CR*[2] | **ui4** | See Section 2.3.29 | | |
| *A_ARG_TYPE_QueryRequest* | *CR*[2] | **string** | See Section 2.3.30 | | |
| *A_ARG_TYPE_QueryResult* | *CR*[2] | **string** | See Section 2.3.31 | | |
| *A_ARG_TYPE_FFQCapabilities* | *CR*[2] | **string** | See Section 2.3.32 | | |
| *A_ARG_TYPE_CPID* | *CR*[2] | **string** | See Section 2.3.33 | | |
| *A_ARG_TYPE_DeviceModeID* | *CR*[2] | **string** | See Section 2.3.34 | | |
| *A_ARG_TYPE_DeviceModeRequest* | *CR*[2] | **String** | See Section 2.3.35 | | |
| *Non-standard state variables implemented by a UPnP vendor go here* | *X* | *TBD* | *TBD* | *TBD* | *TBD* |

[1] For a device this column indicates whether the state variable MUST be implemented or not, where *R* = REQUIRED, *O* = OPTIONAL, *CR* = CONDITIONALLY REQUIRED, *CO* = CONDITIONALLY OPTIONAL, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*).

[2] *CR* = CONDITIONALLY REQUIRED. See referenced section for implementation requirements.

### 2.3.2 *SearchCapabilities*

This REQUIRED 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 Section 1.4, “Management of XML Namespaces in Standardized DCPs” and Appendix B, “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 Section 2.2.20, “*CDS Properties*”. 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 REQUIRED 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 Section 2.2.20, "*CDS Properties*". For example, "`upnp:foreignMetadata::fmBody::fmURI`".

### 2.3.4 *SortExtensionCapabilities*

This CONDITIONALLY REQUIRED state variable MUST be supported if sort modifiers other than "+" and "-" are defined. It 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.

**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.<br>When a ContentDirectory service implements the *SortExtensionCapabilities* state variable, the values "+" and "-" MUST be included. |

| Sort Modifiers | Descriptions |
|---|---|
| *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.<br>1. Object D that has "2005-02-10" in *dc:date*.<br>2. Object A that has "2004-05-08T10:00:00" in *dc:date*.<br>3. Object C that has "2004-05-11T12:00:00" in *dc:date*.<br>4. Object B that has "2003-02-12T18:30:00" in *dc:date*.<br>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*

This REQUIRED state variable 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 Section 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 Section 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 Section 2.3.7, "*ServiceResetToken*" and Section 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 Section 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 Section 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 Appendix B.19.2, "*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 Appendix B.19.2, "*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 Section 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 Section 2.2.9, "*Container Modification*"), its *ContainerUpdateIDValue* indicator changes according to the rules in Section 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 occurried 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**

| | | New value of *ContainerUpdateIDValue* | |
|---|---|---|---|
| **Action** | ***@id*** | **⇓** | **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 value of *ContainerUpdateIDValue*** | |
|---|---|---|---|
| **Action** | ***@id*** | **⇓** | **New value of *ContainerUpdateIDs*** |
| 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 Section 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 Section 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 Appendix B.19.2, "upnp:objectUpdateID".
- All upnp:containerUpdateID properties MUST be reset according to the initialization requirements defined in Appendix B.19.1, "upnp:containerUpdateID".
- All upnp:totalDeletedChildCount properties MUST be reset according to the initialization requirements defined in Appendix B.19.3, "upnp:totalDeletedChildCount".
- All res@updateCount properties MUST be reset according to the initialization requirements defined in Appendix B.19.4, "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

This CONDITIONALLY REQUIRED state variable MUST be supported when the ContentDirectory service implements 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 Section 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 Section 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 Section 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 Section 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 Section 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 Section 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 Section 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 Section 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 Section 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 Section 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 Section 2.4, "Eventing and Moderation" and Section 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 Section 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 Section 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 CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements either the *ImportResource()* or *ExportResource()* actions. The 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 REQUIRED 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 and MUST have a name attribute containing the assigned name of the feature.
- 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 DeviceMode

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *DEVICE_MODE feature*. It is used to inform control points about the current mode of the ContentDirectory service implementation. It is of type string and contains the list of device modes that are currently active. The *DeviceMode* state variable is evented and moderated according to the GENA eventing mechanism as defined by the UPnP Device Architecture Specification [DEVICE].

If the ContentDirectory service does not support the *DEVICE_MODE feature* then the *DeviceMode* state variable MUST NOT be included in the list of ContentDirectory service state variables.

If the ContentDirectory service implementation supports the *DEVICE_MODE feature* then it MUST be in one of the following modes:

- *ActionBurst mode*
- *ExclusiveOwnership mode*
- Normal mode, that is, the absence of either of the above modes.

The following is the XML template for *DeviceMode* state variable.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceMode
 xmlns="urn:schemas-upnp-org:av:dmo"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:dmo
      http://www.upnp.org/schemas/av/dmo.xsd">
   <mode type="Mode is ActionBurst or ExclusiveOwnership"
         CPRequested="Mode requested by control point">
   </mode>
```

```
</DeviceMode>
```

<?xml>

OPTIONAL. Case sensitive.

<DeviceMode>

REQUIRED. MUST include a namespace declaration for the ContentDirectory service [DMO-XSD] Schema (“urn:schemas-upnp-org:av:dmo”). MUST include zero or more of the following elements based on the current mode of the device. If the device is in the normal mode, then all of following elements MUST NOT be present.

<mode>

REQUIRED. xsd:string. Enumerates the mode(s) in which the device is at this instance.

type

REQUIRED. xsd:string. Contains the type of the device mode. The two mode values currently defined are “*ActionBurst*” and “*ExclusiveOwnership*”.

CPRequested

REQUIRED. xsd:boolean. Contains the nature of the device mode request. A value of “*1*” indicates that the mode was requested externally (by a control point), a value of “*0*” indicates that the mode was initiated from an internal trigger, such as a Tuner being allocated.

### 2.3.12 *DeviceModeStatus*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *DEVICE_MODE feature*. It is used to inform control points about the specific details of an active device mode that has been granted to a control point. It is an XML document that MUST conform to the format below.

If the ContentDirectory service does not support the *DEVICE_MODE feature* then the *DeviceModeStatus* state variable MUST NOT be included in the list of ContentDirectory service state variables.

The following is the XML template for *DeviceModeStatus* state variable.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceModeStatus
 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">
   <actionBurstStatus>
      <totalTime>remaining time(millisecs)</totalTime>
      <responseTime>response time(millisecs)</responseTime>
   </actionBurstStatus>
   <exclusiveOwnershipStatus>
      <resourceID type="resource-id"></resourceID>
      <totalTime>remaining time(millisecs)</totalTime>
   </exclusiveOwnershipStatus>
</DeviceModeStatus>
```

<?xml>

OPTIONAL. Case sensitive.

`<DeviceModeStatus>`

REQUIRED. MUST include a namespace declaration for the ContentDirectory service [DMOS-XSD] Schema ("urn:schemas-upnp-org:av:dmos"). MUST include zero or more of the following elements based on the current mode of the device. If the device is in the normal mode, then all of following elements MUST NOT be present.

`<actionBurstStatus>`

OPTIONAL. Indicates that a control point has been granted an *ActionBurst* request. Contains the following attributes and elements:

`<totalTime>`

REQUIRED. xsd:unsignedInt (milliseconds). Contains the most recent measurement of the remaining time the ContentDirectory service implementation has allocated for this *ActionBurst* device mode to remain active. MUST NOT be greater than the previous value, unless a *RequestDeviceMode()* or *ExtendDeviceMode()* action has been successfully invoked. In other words it is a count-down timer with accuracy determined by the ContentDirectory service implementation.

`<responseTime>`

REQUIRED. xsd:unsignedInt (milliseconds), Maximum allowed delay between consecutive invocations of two action requests. MUST be set to the most recently granted value from a *RequestDeviceMode()* or *ExtendDeviceMode()* action. The value MAY be reduced if the `<totalTime>` element value becomes less than the `<requestTime>` element value.

`<exclusiveOwnershipStatus>`

OPTIONAL. Indicates that a control point has been granted *ExclusiveOwnership* of the device for a period of time. Contains all of the following attributes and elements:

`<resourceID>`

REQUIRED. xsd:unsignedInt. Contains the vendor-defined ID of the resource to be locked for exclusive use

`type`

REQUIRED. xsd:string. Contains the type of device resource requested. The one resource currently defined is "*Device*".

`<totalTime>`

REQUIRED. xsd:unsignedInt (milliseconds). Contains the most recent measurement of the remaining time the ContentDirectory service implementation has allocated for this *ExclusiveOwnership* device mode to remain active. MUST NOT be greater than the previous value, unless a *RequestDeviceMode()* or *ExtendDeviceMode()* action has been successfully invoked. In other words it is a count-down timer with accuracy determined by the ContentDirectory service implementation.

Note: Although the time based sub-elements are in units of milliseconds this does not require the ContentDirectory service implementation or control point to implement their internal clocks to a granularity of millisecond accuracy. However, it is RECOMMENDED that the clock accuracy be on the order of the minimum time the ContentDirectory service will grant for the `<responseTime>` element.

An example of the *DeviceModeStatus* state variable while in the *ActionBurst* mode is shown in the following, where an initial time of 100 seconds has been allocated for the *ActionBurst*, and a 2 second maximum response between control point actions.

### 2.3.13 *PermissionsInfo*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *CONTENT_PROTECTION feature* (see Appendix E.10 and Appendix F) and PROHIBITED otherwise. It contains the auxillary information not obtainable through the DeviceProtection service *GetACLData()* and *GetRolesForAction()* actions. The format of the *PermissionsInfo* state variable is an XML document and complies with the PermissionsInfo XML Schema [PI-XSD].

The following is the XML template for the PermissionsInfo state variable.

```
<?xml version="1.0" encoding="UTF-8"?>
<PermissionsInfo
 xmlns="urn:schemas-upnp-org:av:pi"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:pi
      http://www.upnp.org/schemas/av/pi.xsd">
  <nonRestrictable>Name of an action supported by this implementation
  </nonRestrictable>
  <ownAll>Name of a Role supported by this implementation</ownAll>
  <includeAll>
      Name of a Role supported by this implementation
  </includeAll>
</PermissionsInfo>
```

The rules for describing the PermissionsInfo state variable document structure are defined as follows.

`<?xml>`

OPTIONAL. Case sensitive.

`<PermissionsInfo>`

REQUIRED. MUST include a namespace declaration for the ContentDirectory service [PI-XSD] Schema (“urn:schemas-upnp-org:av:pi”). MUST include one or more of each of the following elements. This namespace defines the following elements:

`<nonRestrictable>`

REQUIRED. xsd:string, indicates an always invokable (or *Non-Restrictable*) action. Its allowed value is the name of an action supported by the ContentDirectory service implementation for this device and MUST be identical to an action name exposed by the device for the ContentDirectory service. The element MAY appear more than once but the exact value (action name) MUST NOT be duplicated. All actions that appear in Table F-3, “Pre-defined settings for Restrictable *and Non-Restrictable* AV Actions” with a value of *Non-Restrictable* and that are implemented for a given ContentDirectory service MUST have a `<nonRestrictable>` element in the PermissionsInfo state variable. Actions defined for the ContentDirectory service but not appearing in a `<nonRestrictable>` element are considered to be *Restrictable*.

Note: Actions defined by the AV working committee as *Non-Restrictable* MUST NOT be changed to *Restrictable* by a specific implementation. Likewise, actions defined by the AV working committee as *Non-Restrictable* MUST be included in a `<nonRestrictable>` element.

`<includeAll>`

REQUIRED. xsd:string, indicates that an individual *Role* MUST be implicitly considered included (or present) in the upnp:inclusionControl property of all objects in the ContentDirectory service. Its allowed value MUST be one of the current *Roles* as managed by the DeviceProtection service for the device hosting the ContentDirectory service implementation. A given value MUST occur only once in the PermissionsInfo state variable. At a minimum, an `<includeAll>` element MUST be present for the required AV:SuperReader *Role*.

`<ownAll>`

REQUIRED. xsd:string, indicates that an individual *Role* MUST be implicitly considered included (or present) in the upnp:objectOwner property of all objects in the ContentDirectory service. Its allowed value MUST be one of the current *Roles* as managed by the DeviceProtection service for the device hosting the ContentDirectory service implementation. A given value MUST occur only once in the permissionsInfo state variable. At a minimum, an `<ownAll>` element MUST be present for the required AV:SuperWriter *Role*.

An example of the *PermissionsInfo* state variable along with additional requirements regarding *Roles* for the *CONTENT_PROTECTION feature* can be found in Appendix F.1, “AV *Roles* for Content_Protection”.

### 2.3.14 *A_ARG_TYPE_ObjectID*

This REQUIRED 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.

### 2.3.15 *A_ARG_TYPE_Result*

This REQUIRED 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 Appendix B, “AV Working Committee Properties”.

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

### 2.3.16 *A_ARG_TYPE_SearchCriteria*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *Search()* action. The 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 Section 2.2.20, “*CDS Properties*”. 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.

#### 2.3.16.1 *SearchCriteria* String Syntax

*SearchCriteria* string syntax is described here formally using EBNF as described in Section 1.2.3, "Extended Backus-Naur Form". A ContentDirectory service implementation MUST support the entire syntax as described below.

```
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'|
                  'startsWith'|'derivedFrom'
existsOp      ::= 'exists'
boolVal       ::= 'true'|'false'
quotedVal     ::= dQuote escapedQuote dQuote
wChar         ::= space|hTab|lineFeed|vTab|formFeed|return
property      ::= (* property name as defined in Section 2.2.20 *)
escapedQuote  ::= (* double-quote escaped string as defined in
                  Section 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 *)
```

String operators are case insensitive.

### 2.3.16.2 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**. Relation operators “<”, “>” and comparison operators `contains` , `doesNotContain`, and `startsWith` when applied to strings use case-insensitive comparisons. Overloading of the `derivedFrom` operator for string comparison is allowed but discouraged and implementation dependent. Comparison is done based on lexical ordering. Note that this should not be based on Unicode character values but rather based on localized lexical conventions. For example, the “ö” character in German is ordered between the “n” and “p” characters whereas in Swedish, it s ordered after “z”. See [UNICODE COLLATION].

### 2.3.17 *A_ARG_TYPE_BrowseFlag*

This REQUIRED 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.

### 2.3.18 *A_ARG_TYPE_Filter*

This REQUIRED 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 Section 2.2.20, "*CDS Properties*". 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. An independent property or an independent child property MAY be suffixed by the "#" U+0023 character. When present, this suffix, indicates that the actions associated with the *A_ARG_TYPE_Filter* argument MUST return all child properties descended from the indicated property.

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)
```

**Example 3**: The *Filter* argument in a *Search()* action is specified as `"res#"`, indicating all attributes of this property and all child properties descended from this property are to be returned.

```
Request:
Search("0", "dc:title contains "tenderness"",
     "res#", 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>
      <res  protocolInfo="http-get:*:audio/mpeg:*"
            bitrate=”6553”
            nrAudioChannels=”2”
            duration=”03:12”
            size=”1258291”>
         http://10.0.0.1/audio/O-MP3-211.mp3
      </res>
   </item>
</DIDL-Lite>", 1, 1, 2345)
```

## 2.3.19 *A_ARG_TYPE_SortCriteria*

This REQUIRED 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 Section 2.2.20, “*CDS Properties*”. 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 Section 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.

### 2.3.20 *A_ARG_TYPE_Index*

This REQUIRED 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.

### 2.3.21 *A_ARG_TYPE_Count*

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

### 2.3.22 *A_ARG_TYPE_UpdateID*

This REQUIRED 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 Section 2.3.5, “*SystemUpdateID*”)

### 2.3.23 *A_ARG_TYPE_TransferID*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *ImportResource()* or *ExportResource()* actions.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.

### 2.3.24 *A_ARG_TYPE_TransferStatus*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *GetTransferProgress()* action. 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.

### 2.3.25 *A_ARG_TYPE_TransferLength*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *GetTransferProgress()* action. 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.

### 2.3.26 *A_ARG_TYPE_TransferTotal*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *GetTransferProgress()* action. 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.

### 2.3.27 *A_ARG_TYPE_TagValueList*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *UpdateObject()* action.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 Section 1.3.1, "Comma Separated Value (CSV) Lists".

### 2.3.28 *A_ARG_TYPE_URI*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *ImportResource(), ExportResource() or DeleteResource()* actions. 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).

### 2.3.29 *A_ARG_TYPE_CDSView*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *FreeFormQuery()* action. 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.

### 2.3.30 *A_ARG_TYPE_QueryRequest*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *FreeFormQuery()* action.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 section 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 Section 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 Section 2.5.20, "*GetFreeFormQueryCapabilities()*").All other property names MUST NOT be used in this argument. The content of each `<propertyName>` element (see Section 2.3.32, "*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: Section 2.4 Character Data and Markup) when transmitted in a SOAP message.

### 2.3.31 *A_ARG_TYPE_QueryResult*

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *FreeFormQuery()* action. 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: Section 2.4 Character Data and Markup) when transmitted in a SOAP message.

### 2.3.32 A_ARG_TYPE_FFQCapabilities

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the GetFreeFormQueryCapabilities() action. The 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 Section 2.2.20, "*CDS Properties*"). 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 (Section 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 Section 2.3.30, "*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] Section 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>
```

### 2.3.33 A_ARG_TYPE_CPID

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *DEVICE_MODE feature*. This state variable is introduced to provide type information for the *CPID* input argument of the *RequestDeviceMode()* action. It is of type string. It provides a unique token by which a control point can identify itself to a ContentDirectory service implementation. It is highly RECOMMENDED that the value of this string be a GUID and be persisted for each control point, and if supported, each unique Control Point and User Identity.

### 2.3.34 A_ARG_TYPE_DeviceModeID

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *DEVICE_MODE feature*. The state variable is introduced to provide type information for the *DeviceModeID* input and output arguments. It is of type string. It MUST be unique within the ContentDirectory service implementation and MUST be changed with each granting of a new device mode request.

### 2.3.35 A_ARG_TYPE_DeviceModeRequest

This CONDITIONALLY REQUIRED state variable MUST be supported if the ContentDirectory service implements the *DEVICE_MODE feature*. The state variable is introduced to provide type information for the *DeviceModeRequest* input argument of the *RequestDeviceMode()* and *ExtendDeviceMode()* actions. It is an XML document and of type string and MUST conform to the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceModeRequest
 xmlns="urn:schemas-upnp-org:av:dmor"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:dmor
      http://www.upnp.org/schemas/av/dmor.xsd">
   <actionBurstRequest>
      <totalTime>time(millisecs)</totalTime>
      <responseTime>max action delay</responseTime>
      <label>CP-generated request label<label>
      <description>description text</description>
      <actionName count="invocation count">action name</actionName>
      <actionName count="invocation count">action name</actionName>
      <actionName count="invocation count" size="memory needed">
```

```
         action name
      </actionName>
   </actionBurstRequest>
   <exclusiveOwnershipRequest>
      <resourceID type="Device">ID</resourceID>
      <totalTime>time(millisecs)</TotalTime>
      <label>CP-generated request label<label>
      <description>description text</description>
   </exclusiveOwnershipRequest>
</DeviceModeRequest>
```

<?xml>

OPTIONAL. Case sensitive.

<DeviceModeRequest>

REQUIRED. MUST include a namespace declaration for the ContentDirectory service [DMOR-XSD] Schema (“urn:schemas-upnp-org:av:dmor”). MUST include only one of the following elements.

<actionBurstRequest>

OPTIONAL. Identifies a request for the device to enter the *ActionBurst* mode. Contains all of the following elements:

<totalTime>

REQUIRED. xsd:unsignedInt (milliseconds). Indicates the total amount of time being requested for the *ActionBurst* mode. It SHOULD be based on some previous experience with the ContentDirectory service implementation for similar actions. If left empty the control point is indicating that it wants the ContentDirectory service implementation to assign a default value for totalTime. The value is expressed in units of milliseconds. It has a type of unsigned integer. Contains the most recent measurement of the remaining time the ContentDirectory service implementation has allocated for this *ActionBurst mode* to remain active. MUST NOT be greater than the previous value, unless a *RequestDeviceMode()* or *ExtendDeviceMode()* action has been successfully invoked. In other words it is a count-down timer with accuracy determined by the ContentDirectory service implementation.

<responseTime>

REQUIRED. xsd:unsignedInt (milliseconds). Indicates the requested maximum amount of time to allow between consecutive action invocations. It only applies after the first control point action is invoked. It SHOULD be based on some previous experience or specific knowledge, for example the control point knows it is on a slow link to the ContentDirectory service implementation or that some form of user intervention is involved. Normally it is left empty and the ContentDirectory service implementation will define the expected behavior. The value is expressed in units of milliseconds. It has a type of unsigned integer.

<label>

OPTIONAL. xsd:string. Indicates a label (friendly name) provided with the control point request for performing a specific task, such as “Sync My Music”. It is RECOMMENDED to use the same label to group similar types of *ActionBurst* requests. The <label> element of the <actionBurstRequest> element has a value of type **string**.

<description>

OPTIONAL. xsd:string, Contains end-user displayable description of this *ExclusiveOwnership* request such as “I would like to synchronizing my music?”. The description SHOULD be suitable for passing to an end-user. The <description> element has a value of type **string**.

<actionName>

OPTIONAL. xsd:string. This multi-valued element indicates a particular action that will be invoked during an *ActionBurst* and contains the name of a single action supported by the ContentDirectory service implementation. It is permissible to have multiple <actionName> elements in the input XML document. The order of the <actionName> element does not indicate the order in which the actions will be invoked and in general is not an exhaustive list, therefore as long as any control point is invoking any action within the responseTime time the ContentDirectory service implementation SHOULD continue to honor the *ActionBurst mode* request. However, the ContentDirectory service implementation MAY use the actual behavior of the control point if future similar requests are made in deciding how to grant priority for *ActionBurst*, that is reducing the granted time if a control point consistently underestimates required resources. Also, it is permitted to list each action individually or in any combination equaling the total number of that action to be included in the *ActionBurst*. For example, five *CreateObject()* actions could be indicated by repeating an individual <actionName> element five times or it could be included once with the count attribute of the <actionName> element set to 5

When an actionName element is included, then the ContentDirectory service implementation MAY track the number of times a particular action is invoked during the *ActionBurst* and reduce its expected count and size accordingly. If it does track the *ActionBurst* then the ContentDirectory service MUST set the enforced attribute of the <actionNameProcessing> element of the *DEVICE_MODE feature* value to "*1*"). Unless the ContentDirectory service implementation knows the identity of the requesting control point it SHOULD NOT cancel the granted *ActionBurst mode* and SHOULD NOT reset the *DeviceMode* state variable to the normal mode unless one of the associated timers ( <totalTime> or <responseTime> elements) times out.

Contains the following attributes:

count

OPTIONAL. xsd:unsignedint, If the action is to be requested multiple times during the burst this is indicated by the required count attribute of the <actionName> element. It is of type unsigned integer and MUST have a value greater than zero.

size

OPTIONAL. xsd:unsignedint, If this action, or multiple actions, will require a certain amount of permanent storage (metadata plus uploaded content) on the ContentDirectory service implementation, the amount of storage can be indicated by the optional size attribute of the <actionName> element. The size attribute of the <actionName> element is in units of bytes, is of type unsigned integer, and MUST have a value greater than zero. If an <actionName> element includes both size and count attributes,then the size attribute of the <actionName> element_indicates the net storage requirement for the multiple invocations.

<exclusiveOwnershipRequest>

OPTIONAL. Contains the details of an *ExclusiveOwnership* request or an active *ExclusiveOwnership mode*. Contains at least some of the following child elements:

<resourceID>

REQUIRED. xsd:string. Contains the vendor-defined ID of the resource to be locked for exclusive use.

type

Required. xsd:string. Contains the type of device resource requested. The only resource currently defined is "*Device*".

<totalTime>

REQUIRED. xsd:unsignedInt (milliseconds). Contains the total amount of time requested for the *ExclusiveOwnership mode*. An empty value indicates that the ContentDirectory service implementation MUST determine the amount of time to grant.

<label>

OPTIONAL. xsd:string. Indicates a label (friendly name) provided with the control point request for performing a specific task, such as "Sync My Music". It is RECOMMENDED to use the same label to group similar type *ExclusiveOwnership* requests. The <label> element has a value of type **string**.

`<description>`

OPTIONAL. xsd:string, Contains end-user displayable description of this *ExclusiveOwnership* request such as "Synchronizing Mary's music". The description SHOULD be suitable for passing to an end-user. The `<description>` element has a value of type **string**.

## 2.4 Eventing and Moderation

### Table 2-7: Event moderation

| Variable Name | Evented | Moderated Event | Max Event Rate[1] | Logical Combination | Min Delta per Event[2] |
|---|---|---|---|---|---|
| *TransferIDs* | *YES* | *NO* | | | |
| *A_ARG_TYPE_ObjectID* | *NO* | *NO* | | | |
| *A_ARG_TYPE_Result* | *NO* | *NO* | | | |
| *A_ARG_TYPE_SearchCriteria* | *NO* | *NO* | | | |
| *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* | | | |
| *A_ARG_TYPE_CPID* | *NO* | *NO* | | | |
| *A_ARG_TYPE_DeviceModeID* | *NO* | *NO* | | | |
| *A_ARG_TYPE_DeviceModeRequest* | *NO* | *NO* | | | |
| *SearchCapabilities* | *NO* | *NO* | | | |
| *SortCapabilities* | *NO* | *NO* | | | |
| *SortExtensionCapabilities* | *NO* | *NO* | | | |

| Variable Name | Evented | Moderated Event | Max Event Rate[1] | Logical Combination | Min Delta per Event[2] |
|---|---|---|---|---|---|
| *FeatureList* | *NO* | *NO* | | | |
| *SystemUpdateID* | *YES* | *YES* | 0.2 seconds | | |
| *ContainerUpdateIDs* | *YES* | *YES* | 0.2 seconds | | |
| *ServiceResetToken* | *NO* | *NO* | | | |
| *LastChange* | *YES* | *YES* | 0.2 seconds | | |
| *DeviceMode* | *YES* | *YES* | 0.2 seconds | | |
| *DeviceModeStatus* | *NO* | *NO* | | | |
| *PermissionsInfo* | *NO* | *NO* | | | |
| *Non-standard state variables implemented by a UPnP vendor go here* | *TBD* | *TBD* | *TBD* | *TBD* | *TBD* |

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

[2] (N) * (allowedValueRange Step)

## 2.5 Actions

The following tables and subsections 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[1] | Control Point R/O[2] |
|---|---|---|
| *GetSearchCapabilities()* | *R* | *O* |
| *GetSortCapabilities()* | *R* | *O* |
| *GetSortExtensionCapabilities()* | *CR*[3] | *O* |
| *GetFeatureList()* | *R* | *O* |
| *GetSystemUpdateID()* | *R* | *O* |
| *GetServiceResetToken()* | *R* | *O* |
| *Browse()* | *R* | *R* |
| *Search()* | *O* | *O* |
| *CreateObject()* | *O* | *O* |
| *DestroyObject()* | *O* | *O* |
| *UpdateObject()* | *O* | *O* |
| *MoveObject()* | *O* | *O* |
| *ImportResource()* | *O* | *O* |
| *ExportResource()* | *O* | *O* |

| Name | R/O[1] | Control Point R/O[2] |
|---|---|---|
| *DeleteResource()* | *O* | *O* |
| *StopTransferResource()* | *CO*[4] | *O* |
| *GetTransferProgress()* | *CO*[4] | *O* |
| *CreateReference()* | *O* | *O* |
| *FreeFormQuery()* | *O* | *O* |
| *GetFreeFormQueryCapabilities()* | *CR*[3] | *O* |
| *RequestDeviceMode()* | *CR*[3] | *O* |
| *ExtendDeviceMode()* | *CR*[3] | *O* |
| *CancelDeviceMode()* | *CR*[3] | *O* |
| *GetDeviceMode()* | *CR*[3] | *O* |
| *GetDeviceModeStatus()* | *CR*[3] | *O* |
| *GetPermissionsInfo()* | *CR*[3] | *O* |
| *Non-standard actions implemented by an UPnP vendor go here.* | *X* | *X* |

[1] For a device this column indicates whether the action MUST be implemented or not, where *R* = REQUIRED, *O* = OPTIONAL, *CR* = CONDITIONALLY REQUIRED, *CO* = CONDITIONALLY OPTIONAL, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*).

[2] For a control point this column indicates whether a control point MUST be capable of invoking this action, where *R* = REQUIRED, *O* = OPTIONAL, *CR* = CONDITIONALLY REQUIRED, *CO* = CONDITIONALLY OPTIONAL, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*).

[3] See action description for conditions under which implementation of this action is REQUIRED.

[4] See action description for conditions under which implementation of this action is OPTIONAL. If the condition is not met implementation of this action is PROHIBITED.

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 REQUIRED 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |

### 2.5.2 *GetSortCapabilities()*

This REQUIRED 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |

### 2.5.3 *GetSortExtensionCapabilities()*

This action returns the CSV list of sort modifiers supported by the ContentDirectory service. This CONDITIONALLY REQUIRED 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |

### 2.5.4 *GetFeatureList()*

This REQUIRED 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |

| errorCode | errorDescription | Description |
|---|---|---|
| 600-699 | TBD | See UPnP Device Architecture section on Control. |

### 2.5.5 *GetSystemUpdateID()*

This REQUIRED 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |

### 2.5.6 *GetServiceResetToken()*

This REQUIRED 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 Section 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |

## 2.5.7 *Browse()*

This REQUIRED 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 CONTENT_PROTECTION feature* requirements

The CONDITIONALLY REQUIRED modifications to the *Browse()* action described in Section F.2.2 MUST be implemented when the *CONTENT_PROTECTION feature* is supported.

### 2.5.7.2 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 Section 2.3.17, "*A_ARG_TYPE_BrowseFlag*."
- *Filter*: See Section 2.3.18, "*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 Section 2.3.19, "*A_ARG_TYPE_SortCriteria*."
- *Result*: See Section 2.3.15, "*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* |

### 2.5.7.3 Dependency on State

None.

### 2.5.7.4 Effect on State

None.

### 2.5.7.5 Errors

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

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |
| 701 | No such object | The *Browse()* request failed because the specified *ObjectID* argument is invalid. |

| errorCode | errorDescription | Description |
|---|---|---|
| 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 OPTIONAL 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 CONTENT_PROTECTION feature* requirements

The CONDITIONALLY REQUIRED modifications to the *Search()* action described in Section F.2.4 MUST be implemented when the *CONTENT_PROTECTION feature* is supported.

#### 2.5.8.2 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 Section 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 Section 2.3.16, "*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* |

| Argument | Direction | Related State Variable |
|---|---|---|
| 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 |

#### 2.5.8.3 Dependency on State

None.

#### 2.5.8.4 Effect on State

None.

#### 2.5.8.5 Errors

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

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section 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 OPTIONAL 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 the following ways:

- **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.

  In addition to pre-existing resources, submitted object metadata MAY contain additional properties requiring updates to *res* property values provided by the control point. See Section 2.5.9.5, “Create Segment Items” for additional information.

  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>
…
```

**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:

a. 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.

b. 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 (1. and 2.), 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.

Additional metadata associated with the created *res* property can be supplied by the control point via the *upnp:resExt* property (see Section B.3.1) and relevant child properties thereof. If the *upnp:resExt* property is specified by the control point, then the *upnp:resExt@id* property value MUST match the specified *res@id* property. If no match is found, then the ContentDirectory service MUST generate error code 712 – "Bad metadata".

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".

#### 2.5.9.2 Create Reference Items

*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.

#### 2.5.9.3 Create Bookmark Items

*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 Appendix 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.

Note:

1. 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.
2. 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.

#### 2.5.9.4 Create Multi-component Items

If a ContentDirectory service implementation supports the *MULTI_STREAM feature* (see Section E.6), then the *upnp:resExt::componentInfo* property (see Section B.15) and its child properties are used to provide information about the media components associated with a resource representing a (multiplexed) stream. The control point can provide this metadata during the invocation of the *CreateObject()* action.

#### 2.5.9.4.1 Uploading of content for Multi-component Items

The procedure described in Section 2.5.9.1, "*res* Property Creation" MUST be followed for the creation of the *upnp:resExt::componentInfo::componentGroup::component::compRes::res* property.

### 2.5.9.5 Create Segment Items

If the ContentDirectory service implementation supports the *SEGMENTATION feature*, *CreateObject()* can also be used to create a new segment item. A segment item can be created in any container. When a segment item is created, the associated base content item MUST be updated so that one of its *upnp:segmentID* properties contains the object ID of the newly created segment item. After the segment item is created, it MUST contain the object ID of the base content item in its *upnp:resExt::segmentInfo@baseObjectID* property.

Each pair of *upnp:resExt::segmentInfo@baseObjectID* and *upnp:resExt::segmentInfo@baseResID* properties of the *Elements* input argument MUST uniquely identify a certain *res* property in the base content item. The initial value of the segment *res* property MUST contain the URI value from the base item *res* property indicated by the *upnp:resExt::segmentInfo@baseResID* property. The segment *res* property value SHOULD be updated by the ContentDirectory service if the segment is created successfully. A control point MAY compare the initial *res* property URI values provided and the *res* property values in the *CreateObject()* action *Result* output argument. An unchanged *res* property URI value indicates that the segment item create operation was either unsuccessful or that segmented content creation is not supported by the ContentDirectory service implementation. Since multiple segment *res* properties may be created during a single *CreateObject()* operation, the control point should check that all submitted *res* property URI values were updated. If the submitted *res* URI properties were not updated, then it is recommended that the control point delete the newly created segment item.

When creating a segment *res* property, the required *upnp:resExt::segmentInfo::timeRange* property MUST be provided. In addition, the optional *upnp:resExt::segmentInfo::frameRange* and *upnp:resExt::segmentInfo::byteRange* properties MAY be also required for specific segment base item media formats as indicated by the presence of `<additionalInfoRequired>` elements of the *SEGMENTATION feature* element. See Appendix E.7, "Requirements for the *SEGMENTATION feature*, Version 1" for additional information.

A ContentDirectory service implementation MAY adjust control point specified values for a created segment's time range, byte range and frame range properties to maintain consistency between these properties and to align with playable media boundaries.

### 2.5.9.6 *CONTENT_PROTECTION feature* requirements

The CONDITIONALLY REQUIRED modifications to the *CreateObject()* action described in Section F.2.1 MUST be implemented when the *CONTENT_PROTECTION feature* is supported.

### 2.5.9.7 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* |

| Argument | Direction | Related State Variable |
|---|---|---|
| *Result* | *OUT* | *A_ARG_TYPE_Result* |

### 2.5.9.8 Dependency on State

None.

### 2.5.9.9 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.

### 2.5.9.10 Errors

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

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

### 2.5.10.1 Destroying bookmark items and bookmark containers

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.

### 2.5.10.2 Destroying segment items and base content items

The *DestroyObject()* action can be used to destroy segment items or base content items.

When a segment item is to be destroyed, the ContentDirectory service MUST find the associated base content item using the *upnp:resExt::segmentInfo@baseObjectID* property of the segment item and MUST remove the associated *upnp:segmentID* property from the base content item.

If a base content item containing one or more *upnp:segmentID* properties is destroyed, the ContentDirectory service MUST examine the *res* properties of each segment item identified by the *upnp:segmentID* property value in the item being destroyed. Segment item *res* properties associated with the base content item being destroyed (as indicated by *upnp:resExt::segmentInfo@baseObjectID*) MUST be removed from the segment item. If the updated segment item no longer contains any segment *res* properties, then the segment item MUST also be destroyed.

### *2.5.10.3 CONTENT_PROTECTION feature* requirements

The CONDITIONALLY REQUIRED modifications to the *DestroyObject()* action described in Section F.2.6 MUST be implemented when the *CONTENT_PROTECTION feature* is supported.

### 2.5.10.4 Arguments

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

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

### 2.5.10.5 Dependency on State

None.

### 2.5.10.6 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.

### 2.5.10.7 Errors

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

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section 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 OPTIONAL 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 Appendix B, "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* or *upnp:resExt::componentInfo ::componentGroup::component::compRes::res* properties are to be added, the procedure described in Section 2.5.9.1, "*res* Property Creation" MUST be followed. If there is a *upnp:resExt* property associated with a deleted *res* property, then that *upnp:resExt* property MUST also be deleted.

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 Section 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 Section 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>`<br>`My Favorite Song`<br>`</dc:title>` | `<dc:title>`<br>`My Second Favorite Song`<br>`</dc:title>` | |
| Delete the *dc:date* property | `<dc:date>`<br>`1990-01-01`<br>`</dc:date>` | (Empty entry) | |
| Insert a *upnp:genre* property | (Empty entry) | `<upnp:genre>`<br>`Swing`<br>`</upnp:genre>` | |
| Insert a second value to the multi-value *upnp:genre* property (Option-1) | (Empty entry) | `<upnp:genre>`<br>`Jazz`<br>`</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>`<br>`Swing`<br>`</upnp:genre>` | `<upnp:genre>`<br>`Swing`<br>`</upnp:genre>`<br>`<upnp:genre>`<br>`Jazz`<br>`</upnp:genre>` | Assuming the "Swing" genre already exists, this operation results in two genre properties with a value of "Swing" and "Jazz". |

| Operation | CurrentTagValue | NewTagValue | Notes |
|---|---|---|---|
| Change the upnp:artist property from Singer1 to Singer2 | `<upnp:artist>`<br>`Singer1`<br>`</upnp:artist>` | `<upnp:artist>`<br>`Singer2`<br>`</upnp:artist>` | The entire top-level XML element (that is: `<upnp:artist>`) is included in both the CurrentTagValue and NewTagValue arguments. |
| Change the dc:title property, insert another upnp:genre property, and delete the dc:publisher property | `<dc:title>`<br>`My Favorite Song`<br>`</dc:title>,,`<br>`<dc:publisher>`<br>`Acme Music`<br>`</dc:publisher>` | `<dc:title>`<br>`My Third Favorite Song`<br>`</dc:title>,`<br>`<upnp:genre>Jazz`<br>`</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>`<br>`Somebody`<br>`</upnp:artist>` | `<upnp:artist>`<br>`Somebody else`<br>`</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*. |

| **Operation** | ***CurrentTagValue*** | ***NewTagValue*** | **Notes** |
|---|---|---|---|
| Deleting a modified inherited property, for example, the *upnp:artist* property from above. | `<upnp:artist>`<br>`Somebody else`<br>`</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>`<br>`Somebody`<br>`</upnp:artist>`<br><br>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. |
| Changing the value of the *upnp:desc* property. | `<desc id:"xyz"`<br>`namespace="MyNS">`<br>`  <MyNS:Tag1>`<br>`    value1`<br>`  </MyNS:Tag1>`<br>`  <MyNS:Tag2>`<br>`    old_value`<br>`  </MyNS:Tag2>`<br>`</desc>` | `<desc id:"xyz"`<br>`namespace="MyNS">`<br>`  <MyNS:Tag1>`<br>`    value1`<br>`  </MyNS:Tag1>`<br>`  <MyNS:Tag2>`<br>`    new_value`<br>`  </MyNS:Tag2>`<br>`</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 Updating items containing segment res properties

The UpdateObject() action can be used to add, or delete segment res properties of an existing item. The UpdateObject() action MUST result in an item containing at least one segment res property. As indicated by Table B-1, metadata properties associated with the upnp:resExt::segmentInfo property are designated as "R" (read-only). To update an existing segment res property, the existing res property may be deleted and replaced with a new res property with a corresponding upnp:resExt::segmentInfo property containing updated segment information. The res property removal and addition updates SHOULD be done in the same UpdateObject() action invocation since removal of all segment related res properties in an updated item may trigger the clean-up process described in the following paragraph.

If updates to an item remove all segment res properties referring to a base content item, then the ContentDirectory service MUST update the base content item's upnp:segmentID properties to remove references to the updated segment item. If the UpdateObject() action is used to add a segment to an existing item, then the ContentDirectory service MUST ensure that the base item's properties are updated so that the base item contains at least one upnp:segmentID property identifying the updated segment item.

Updates to a base content media object referenced by a res property URI MUST cause deletion of any segment item res properties that reference the base item res property.

### 2.5.11.3 *CONTENT_PROTECTION feature* requirements

The CONDITIONALLY REQUIRED modifications to the UpdateObject() action described in F.2.2 MUST be implemented when the *CONTENT_PROTECTION feature* is supported.

### 2.5.11.4 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* |

### 2.5.11.5 Dependency on State

None.

### 2.5.11.6 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 ContainerUpdateIDValue indicator. Consequently, the ContainerUpdateIDs state variable, if supported, is updated as well.

### 2.5.11.7 Errors

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

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

| errorCode | errorDescription | Description |
|---|---|---|
| 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. |
| 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 *CONTENT_PROTECTION feature* requirements

The CONDITIONALLY REQUIRED modifications to the *MoveObject()* action described in Section F.2.7 MUST be implemented when the *CONTENT_PROTECTION feature* is supported.

### 2.5.12.2 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* |

### 2.5.12.3 Dependency on State

None.

### 2.5.12.4 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.

### 2.5.12.5 Errors

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

| Error Code | Error Description | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |

| Error Code | Error Description | Description |
|---|---|---|
| 600-699 | TBD | See UPnP Device Architecture section 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 OPTIONAL 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* or *upnp:resExt::componentInfo ::componentGroup::component::compRes::res@importUri* property in the ContentDirectory service implementation. The *res@importUri* or *upnp:resExt::componentInfo::componentGroup::component ::compRes::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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section 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 OPTIONAL 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section 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 OPTIONAL 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 deletes 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 *CONTENT_PROTECTION feature* requirements

The CONDITIONALLY REQUIRED modifications to the *DeleteResource()* action described in Section F.2.8 MUST be implemented when the *CONTENT_PROTECTION feature* is supported.

#### 2.5.15.2 Arguments

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

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

#### 2.5.15.3 Dependency on State

None.

#### 2.5.15.4 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.

#### 2.5.15.5 Errors

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

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section 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 CONDITIONALLY OPTIONAL action MAY only be supported if the ContentDirectory service implements the *ImportResource()* or *ExportResource()* actions. Otherwise, implementation of this action is PROHIBITED. The action stops the file transfer initiated either of these actions. 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section 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 CONDITIONALLY OPTIONAL action MAY only be supported if the ContentDirectory service implements the *ImportResource()* or *ExportResource()* actions. Otherwise, implementation of this action is PROHIBITED. It 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section 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 OPTIONAL 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 Section 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |
| 701 | No such object | *CreateReference()* failed because the specified *ObjectID* argument is invalid. |

| errorCode | errorDescription | Description |
|---|---|---|
| 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 OPTIONAL 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 Section 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 Section 2.6.15.3, “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 CONTENT_PROTECTION feature* requirements

The CONDITIONALLY REQUIRED modifications to the FreeFormQuery() action described in Section F.2.5 MUST be implemented when the *CONTENT_PROTECTION feature* is supported.

#### 2.5.19.2 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 Section 2.2.19, "*CDS View*" and Section 2.3.29, "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 Section 2.3.29, "A_ARG_TYPE_QueryRequest".
- QueryResult: contains the result generated by processing the instructions, specified in the QueryRequest argument (see Section 2.3.31, "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 Section 2.6.15.1, "Retrieving the title of all music albums") or it could be a valid *DIDL-Lite XML Document* (see example in Section 2.6.15.2, "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 Section 2.5.7, "Browse()").

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

| Argument | Direction | Related State Variable |
|---|---|---|
| ContainerID | IN | A_ARG_TYPE_ObjectID |
| CDSView | IN | A_ARG_TYPE_CDSView |
| QueryRequest | IN | A_ARG_TYPE_QueryRequest |
| QueryResult | OUT | A_ARG_TYPE_QueryResult |
| UpdateID | OUT | A_ARG_TYPE_UpdateID |

### 2.5.19.3 Dependency on State

None.

### 2.5.19.4 Effect on State

None.

### 2.5.19.5 Errors

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

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

| errorCode | errorDescription | Description |
|---|---|---|
| 724 | Unsupported or invalid CDS View | The FreeFormQuery() request failed because the value specified in the CDS View 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 CONDITIONALLY REQUIRED action MUST be supported if the ContentDirectory service implements the FreeFormQuery() action. 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.

### 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 Section 2.3.32, "*A_ARG_TYPE_FFQCapabilities*" and [AVS-XSD] for details.

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

| 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 section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |

## 2.5.21 RequestDeviceMode()

This CONDITIONALLY REQUIRED action MUST be supported if the ContentDirectory service implements the *DEVICE_MODE feature*. It is used to request that the device temporarily enter into one of its special operating modes as specified by the *DeviceModeRequest* input argument.

### 2.5.21.1 Arguments

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

- CPID: This input argument contains an ID that is self-assigned by the control point to uniquely identify the control point (and if supported Control Point or User Identities) requesting the particular device mode (see Section 2.2.3, "Object Identity" for a related discussion) .It is highly RECOMMENDED that the CPID be a GUID and be persisted for each control point. See Section2.3.33, "A_ARG_TYPE_CPID".
- DeviceModeRequest: This input argument identifies the specific details about the operating mode that is being requested. See Section 2.3.35, "A_ARG_TYPE_DeviceModeRequest".
- DeviceModeID: This output argument contains an ID that is assigned by the device to uniquely identify this particular request that has been granted. This ID is used to extend or cancel the granted operating mode. See Section 2.3.34, "A_ARG_TYPE_DeviceModeID".
- DeviceModeStatus: This output parameter contains the specific details about the operating mode that was actually granted by this request. For example, this data structure indicates the amount of time the device is willing to remain in the requested operating mode. See Section 2.3.12, "DeviceModeStatus".

**Table 2-50: Arguments for RequestDeviceMode()**

| Argument | Direction | Related State Variable |
|---|---|---|
| CPID | IN | A_ARG_TYPE_CPID |
| DeviceModeRequest | IN | A_ARG_TYPE_DeviceModeRequest |
| DeviceModeID | OUT | A_ARG_TYPE_DeviceModeID |
| DeviceModeStatus | OUT | DeviceModeStatus |

### 2.5.21.2 Dependency on State

None

### 2.5.21.3 Effect on State

When successful, the DeviceMode and DeviceModeStatus state variables MUST be set to reflect the granted request.

### 2.5.21.4 Errors

**Table 2-51: Error Codes for RequestDeviceMode()**

If a control point requests priority and it is not granted then one of the following appropriate error codes MUST be returned.

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |
| 727 | Request refused | The device refused the requested mode. |
| 728 | Request invalid | The device mode requested is invalid. |

| errorCode | errorDescription | Description |
|---|---|---|
| 729 | Request includes non-supported action | The request failed because the value included in the *DeviceModeRequest* `<actionName>` element is not an action supported by the ContentDirectory service implementation. |
| 730 | Request requires too many resources | The action failed because the ContentDirectory service implementation did not have enough additional permanent storage available for the sum of all the `size` attribute values of the *DeviceModeRequest* `<actionName>` elements requested. |
| 731 | Already in mode | The request failed because the device is already in that mode. |

Error code 729 and 730 MUST NOT be returned if the `support` attribute of the *DEVICE_MODE feature* element has a value of "*0*".

## 2.5.22 *ExtendDeviceMode()*

This CONDITIONALLY REQUIRED action MUST be supported if the ContentDirectory service implements the *DEVICE_MODE feature*. It is used to extend the amount of time the device is willing to stay in the specified operating mode.

### 2.5.22.1 Arguments

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

- *DeviceModeID:* This input argument identifies the previously granted device mode request that is being extended. Its value MUST match the value of the *DeviceModeID* output argument that was returned by an earlier invocation of the *RequestDeviceMode()* action. The referenced device mode request MUST NOT have yet expired. See Section 2.3.34, "*A_ARG_TYPE_DeviceModeID*".
- *DeviceModeRequest*: This input parameter contains the details for extending the specified device mode. In particular, the value of the `<totalTime>` element is the amount of additional time requested for the specified device mode to remain active. The specified value may be more or less than previously requested. The value may depend on the control points observed performance of the device such as the responsiveness of the device, the control points remaining tasks, or modification of the original request. See Section 2.3.35, "*A_ARG_TYPE_DeviceModeRequest*".
- *DeviceModeStatus*: This output argument contains a revised version of the device mode that was granted. See Section 2.3.12, "*DeviceModeStatus*".

**Table 2-52: Arguments for *ExtendDeviceMode()***

| Argument | Direction | Related State Variable |
|---|---|---|
| *DeviceModeID* | *IN* | *A_ARG_TYPE_DeviceModeID* |
| *DeviceModeRequest* | *IN* | *A_ARG_TYPE_DeviceModeRequest* |
| *DeviceModeStatus* | *OUT* | *DeviceModeStatus* |

### 2.5.22.2 Dependency on State

The *DeviceModeID* argument value MUST be equal to the active (unexpired) device mode request that was previously granted by the device otherwise it MUST return error code 707.

### 2.5.22.3 Effect on State

The *DeviceModeStatus* state variable is modified to reflect the extended device mode.

#### 2.5.22.4 Errors

**Table 2-53: Error Codes for *ExtendDeviceMode()***

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |
| 727 | Request refused | The device refused the requested mode. |
| 728 | Request invalid | The device mode requested is invalid. |
| 729 | Request includes non-supported action | The request failed because the value included in the *DeviceModeRequest* `<actionName>` element is not an action supported by the ContentDirectory service implementation. |
| 730 | Request requires too many resources | The request failed because the ContentDirectory service implementation did not have enough additional permanent storage available for the sum of all the `size` attribute values of the *DeviceModeRequest* `<actionName>` elements requested. |
| 732 | Inconsistent `<ActionName>` element usage | The request failed because the `<actionName>` element was included in the original *RequestDeviceMode()* action and not in the requested *ExtendDeviceMode()* action. |
| 733 | Invalid ID | The specified *DeviceModeID* is invalid. |

Error code 729, 730, and 732 MUST NOT be returned if the `enforce` attribute of the *DEVICE_MODE feature* has a value of "*0*".

### 2.5.23 *CancelDeviceMode()*

This CONDITIONALLY REQUIRED action MUST be supported if the ContentDirectory service implements the *DEVICE_MODE feature*. It is used to cancel an existing, active device mode. The value of the *DeviceModeID* input argument MUST match the value of the *DeviceModeID* returned by an earlier invocation of the *RequestDeviceMode()* or *ExtendDeviceMode()* action.

#### 2.5.23.1 Arguments

**Table 2-54: Arguments for *CancelDeviceMode()***

| Argument | Direction | Related State Variable |
|---|---|---|
| *DeviceModeID* | *IN* | *A_ARG_TYPE_DeviceModeID* |

#### 2.5.23.2 Dependency on State

The *DeviceModeID* MUST one of the active, unexpired device mode requests that was previously granted by the device.

#### 2.5.23.3 Effect on State

The *DeviceMode* and *DeviceModeStatus* state variables MUST be updated to reflect the cancellation of the specified device mode.

#### 2.5.23.4 Errors

**Table 2-55: Error Codes for *CancelDeviceMode()***

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section on Control. |
| 733 | Invalid ID | The specified *DeviceModeID* is invalid. |

### 2.5.24 *GetDeviceMode()*

This CONDITIONALLY REQUIRED action MUST be supported if the ContentDirectory service implements the *DEVICE_MODE feature*. It is used to retrieve the current value of the *DeviceMode* state variable.

#### 2.5.24.1 Arguments

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

**Table 2-56: Arguments for *GetDeviceMode()***

| Argument | Direction | Related State Variable |
|---|---|---|
| *DeviceMode* | *OUT* | *DeviceMode* |

#### 2.5.24.2 Dependency on State

None.

#### 2.5.24.3 Effect on State

None.

#### 2.5.24.4 Errors

**Table 2-57: Error Codes for *GetDeviceMode()***

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

### 2.5.25 *GetDeviceModeStatus()*

This CONDITIONALLY REQUIRED action MUST be supported if the ContentDirectory service implements the *DEVICE_MODE feature*. It is used to retrieve the current value of the *DeviceModeStatus* state variable.

#### 2.5.24.1 Arguments

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

**Table 2-58: Arguments for *GetDeviceModeStatus()***

| Argument | Direction | Related State Variable |
|---|---|---|
| *DeviceModeStatus* | *OUT* | *DeviceModeStatus* |

### 2.5.24.2 Dependency on State

None.

### 2.5.24.3 Effect on State

None.

### 2.5.24.4 Errors

**Table 2-59: Error Codes for *GetDeviceModeStatus()***

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

## 2.5.26 *GetPermissionsInfo()*

This CONDITIONALLY REQUIRED action MUST be supported if the ContentDirectory service implements the *CONTENT_PROTECTION feature*.This action returns the current list of *Roles* which have been implicitly added to the *upnp:inclusionControl* (see `<includeAll>` in *PermissionsInfo* state variable) and *upnp:objectOwner* ( see `<ownall>` in *PermissionsInfo* state variable) properties and current list of AV actions which have been declared as *Non-Restrictable*.

### 2.5.26.1 Arguments

**Table 2-60: Arguments for *GetPermissionsInfo()***

| Argument | Direction | Related State Variable |
|---|---|---|
| *PermissionsInfo* | *OUT* | *PermissionsInfo* |

### 2.5.26.2 Dependency on State

None.

### 2.5.26.3 Effect on State

None.

#### 2.5.26.4 Errors

**Table 2-61: Error Codes for *GetPermissionsInfo()***

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

### 2.5.27 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. Section on Description).

### 2.5.28 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 2-62: Common Error Codes**

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 600-699 | TBD | See UPnP Device Architecture section 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. |
| 727 | Request refused | The device refused the requested mode. |
| 728 | Request invalid | The device mode requested is invalid. |
| 729 | Request includes non-supported action | The request failed because the value included in the *DeviceModeRequest* `<actionName>` element is not an action supported by the ContentDirectory service implementation. |

| errorCode | errorDescription | Description |
|---|---|---|
| 730 | Request requires too many resources | The request failed because the ContentDirectory service implementation did not have enough additional permanent storage available for the sum of all the `size` attribute values of the *DeviceModeRequest* `<actionName>` elements requested. |
| 731 | Already in mode | The request failed because the device is already in that mode. |
| 732 | Inconsistent `<ActionName>` element usage | The request failed because the `<actionName>` element was included in the original *RequestDeviceMode()* action and not in the requested *ExtendDeviceMode()* action. |
| 733 | Invalid ID | The specified *DeviceModeID* is invalid. |
| 734 | Invalid *Role* for *upnp:inclusionControl* or *upnp:objectOwner* property | The *upnp:inclusionControl* or *upnp:objectOwner* property contains at least one invalid control point *Role*. |
| 735 | Invalid *Owner* | The *upnp:objectOwner* property does not include a *Role* allowed to modify the property. |
| 736 | Object locked | The *upnp:objectOwner* property or *upnp:inclusionControl* property cannot be modified since they are currently locked. |
| 737 | Input object not authorized | *MoveObject()* failed because the control point does not have *Object level access* to at least one of the objects it is trying to move. |
| 738 | Output object not authorized | *MoveObject()* failed because the control point does not have *Object level access* to the target container. |
| 739 | Source resource access denied | *DeleteResource()* failed because the control point does not have *Role* permissions to invoke this action on at least one of the objects referencing the resource specified by the *ResourceURI* argument. |
| 740 | Object not authorized | The control point does not have *Role* permissions to invoke this action on at least one of the target objects. |

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 section on Control for more details.

## 2.6 Theory of Operation (Informative)

### 2.6.1 Introduction

This section 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 Section 2.2.3, “Object Identity” and Section 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 Section 2.2.3, “Object Identity” and Section 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 Section 2.3.7, “*ServiceResetToken*” and Section 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 Section 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 ContentDirectory 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 Section 2.3.18, "*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 Section 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 Section 2.3.16, "*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 Section 2.3.18, "*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 Section 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 Creating a New MusicTrack

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 Multi-component media representation

Some objects in the ContentDirectory service represent content whose resource(s) point to content that contain a combination of different media types. For instance, an object whose class is "*object.item.videoItem*" may contain a *res* property, which is associated with an MPEG-2 Transport Stream file, which contains a combination of audio and video components. Some objects may contain additional metadata to describe in detail the characteristics of these audio and video components associated with a resource, by means of the *upnp:resExt* property.

An example of a multi-component stream with one video component, four audio components and two closed caption components is given below. The audio components are grouped in two groups. The components in one of these groups have their own resource, of which one also has another object associated with it. All other components are embedded in the same resource as indicated by the *res* property.

```
<item id="100" parentID="200" restricted="0">
   <dc:title>KBS News</dc:title>
   <upnp:class>object.item.videoItem</upnp:class>
   <res id="100-res-1" protocolInfo="http-get:*:video/mpeg:*"
resolution="1920x1080">
      http://10.0.0.1/content/content?id=100-res
   </res>

  <upnp:resExt id="100-res-1">
     <upnp:isSyncAnchor>1</upnp:isSyncAnchor>
```

```
<upnp:componentInfo>
  <upnp:componentGroup groupID="0" required="1">
    <upnp:component componentID="comp_0">
        <upnp:componentClass>Video</upnp:componentClass>
        <upnp:contentType MIMEType="video/MPV" extendedType="*"/>
    </upnp:component>
  </upnp:componentGroup>
  <upnp:componentGroup groupID="1" required="1">
    <upnp:component componentID="comp_1">
       <upnp:componentClass>Audio</upnp:componentClass>
       <upnp:language>en-US</upnp:language>
       <upnp:contentType MIMEType="audio/ac3" extendedType="*"/>

    </upnp:component>
    <upnp:component componentID="comp_2">
       <upnp:componentClass>Audio</upnp:componentClass>
          <upnp:language>fr</upnp:language>
       <upnp:contentType MIMEType="audio/MPA" extendedType="*"/>
    </upnp:component>
 </upnp:componentGroup>
 <upnp:componentGroup groupID="2" required="1">
    <upnp:component componentID="comp_3">
       <upnp:componentClass>Audio</upnp:componentClass>
          <upnp:language>de</upnp:language>
       <upnp:contentType MIMEType="audio/ac3" extendedType="*"/>
       <upnp:compRes>
          <upnp:res protocolInfo="http-get:*:audio/ac3:*">
             http://10.0.0.1/content/content?id=100-res-comp_3
          </upnp:res>
       </upnp:compRes>
    </upnp:component>
    <upnp:component componentID="comp_4">
       <upnp:componentClass>Audio</upnp:componentClass>
          <upnp:language>zh</upnp:language>
       <upnp:contentType MIMEType="audio/MPA" extendedType="*"/>
       <upnp:compRes>
          <upnp:refUDN>
             uuid:420ae355-8566-880d-ea02-51c5e081aa06
          </upnp:refUDN>
          <upnp:refObjectID>101</upnp:refObjectID>
          <upnp:refResID>0</upnp:refResID>
          <upnp:res protocolInfo="http-get:*:audio/MPA:*">
             http://10.0.0.1/content/content?id=101-res-comp_4
          </upnp:res>
       </upnp:compRes>
    </upnp:component>
 </upnp:componentGroup>
 <upnp:componentgroup groupID="3" required="0">
    <upnp:component componentID="comp_5">
       <upnp:componentClass>Caption</upnp:componentClass>
       <upnp:language>nl</upnp:language>
       <upnp:contentType MIMEType="text/srt" extendedType="*"/>
       <upnp:compRes>
          <upnp:res protocolInfo="http-get:*:text/srt:*">
             http://10.0.0.1/content/content?id=100-res-comp5
          </upnp:res>
       </upnp:compRes>
```

```
         </upnp:component>
         <upnp:component componentID="comp_6" supportID="comp_7">
            <upnp:componentClass>Caption</upnp:componentClass>
            <upnp:language>de</upnp:language>
            <upnp:contentType MIMEType="text/sub" extendedType="*"/>
            <upnp:compRes>
               <upnp:res protocolInfo="http-get:*:text/sub:*">
                  http://10.0.0.1/content/content?id=100-res-comp6
               </upnp:res>
            </upnp:compRes>
         </upnp:component>
         <upnp:component componentID="comp_7" supportive="1">
            <upnp:componentClass>Caption</upnp:componentClass>
            <upnp:contentType MIMEType="text/idx" extendedType="*"/>
            <upnp:compRes>
               <upnp:res protocolInfo="http-get:*:text/idx:*">
                  http://10.0.0.1/content/content?id=100-res-comp7
               </upnp:res>
            </upnp:compRes>
         </upnp:component>
       </upnp:componentGroup>
     </upnp:componentInfo>
   </upnp:resExt>
</item>
```

### 2.6.12.1 Creating a multi-component video object

Invoke *CreateObject()* with the *ContainerID* argument set to 11 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.videoItem*". The descriptions of the media components associated with this video object is given by the *upnp:resExt* property and its child properties.

**Request**:

```
CreateObject("11", "
<?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="11" restricted="0">
      <dc:title>New Video</dc:title>
      <upnp:class>
         object.item.videoItem
      </upnp:class>
      <res id="0" protocolInfo="*:*:video/mpeg:*">
      </res>
      <upnp:resExt id="0">
         <upnp:componentInfo>
            <upnp:componentGroup groupID="group_0" required="1">
               <upnp:component componentID="comp_0">
                  <upnp:componentClass>
                     Video
                  </upnp:componentClass>
                  <upnp:contentType MIMEType="video/M2PV"
                  extendedType="*"/>
               </upnp:component>
            </upnp:componentGroup>
            <upnp:componentGroup groupID="group_1" required="1">
               <upnp:component componentID="comp_1">
                  <upnp:componentClass>
                     Audio
                  </upnp:componentClass>
                  <upnp:contentType MIMEType="audio/ac3"
                  extendedType="*"/>
                  <upnp:language>
                     en-US
                  </upnp:language>
               </upnp:component>
            </upnp:componentGroup>
         </upnp:componentInfo>
      </upnp:resExt>
   </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="11" restricted="0">
      <dc:title>New Video</dc:title>
      <upnp:class>
         object.item.videoItem
      </upnp:class>
      <res id="0" importUri="http://10.0.0.1/pc/item?id=12"
       protocolInfo ="*:*:video/mpeg:*">
      </res>
      <upnp:resExt id="0">
         <upnp:componentInfo>
            <upnp:componentGroup groupID="group_0" required="1">
               <upnp:component componentID="comp_0">
                  <upnp:componentClass>
                     Video
                  </upnp:componentClass>
                  <upnp:contentType MIMEType="video/M2PV"
                  extendedType="*"/>
               </upnp:component>
            </upnp:componentGroup>
            <upnp:componentGroup groupID="group_1" required="1">
               <upnp:component componentID="comp_1">
                  <upnp:componentClass>
```

```
                     Audio
                  </upnp:componentClass>
                  <upnp:contentType MIMEType="audio/ac3"
                  extendedType="*"/>
                  <upnp:language>
                     en-US
                  </upnp:language>
               </upnp:component>
            </upnp:componentGroup>
         </upnp:componentInfo>
      </upnp:resExt>
   </item>
</DIDL-Lite>")
```

### 2.6.12.2 Adding a component to a multi-component video object

In this example, an additional audio track is added as a component to the object previously created in Section 2.6.12.1. The audio track's content has been obtained by a control point. The control point has two ways to add this audio track as a component:

- First create a separate audio item for this sound track, then add a component in the video object with a reference to this audio item.
- Do not create a separate item, but rather add the component directly.

The object creation part by means of the *CreateObject()* action for the first approach is illustrated in Section 2.6.12.1. The audio item has *@id*=”20”, and the resource of the audio track resides in the *res* property with *res@id*=”0”. The audio track is added in the component group with *upnp:resExt ::componentInfo::componentGroup@groupID*=”group_1” This audio track component is added to the video object by using the *UpdateObject()* action. The value for the *upnp:resExt::componentInfo ::componentGroup::component::compRes::res* property of the added component is set equal to the value of the *res* property of the corresponding audio item.

**Request:**

```
UpdateObject("12", "
<upnp:resExt id="0">
   <upnp:componentInfo>
      <upnp:componentGroup groupID="group_1" required="1">
         <upnp:component componentID="comp_1">
            <upnp:componentClass>
               Audio
            </upnp:componentClass>
            <upnp:contentType MIMEType="audio/ac3"
            extendedType="*"/>
            <upnp:language>
               en-US
            </upnp:language>
         </upnp:component>
      </upnp:componentGroup>
   </upnp:componentInfo>
</upnp:resExt>
", "
<upnp:resExt id="0">
   <upnp:componentInfo>
      <upnp:componentGroup groupID="group_1" required="1">
         <upnp:component componentID="comp_1">
            <upnp:componentClass>
               Audio
            </upnp:componentClass>
            <upnp:contentType MIMEType="audio/ac3"
            extendedType="*"/>
            <upnp:language>
               en-US
            </upnp:language>
         </upnp:component>
         <upnp:component componentID="comp_2">
            <upnp:componentClass>
               Audio
            </upnp:componentClass>
            <upnp:contentType MIMEType="audio/mp3"
            extendedType="*"/>
            <upnp:language>
               fr
            </upnp:language>
            <upnp:compRes>
               <upnp:res protocolInfo="http-get:*:audio/mp3:*">
                  http://10.0.0.1/content/content?id=20-res-0
               </upnp:res>
               <upnp:refUDN>
                  uuid:420ae355-8566-880d-ea02-51c5e081aa06
               </upnp:refUDN>
               <upnp:refObjectID>
                  20
               </upnp:refObjectID>
               <upnp:refResID>
                  0
               </upnp:refResID>
            </upnp:compRes>
         </upnp:component>
      </upnp:componentGroup>
   </upnp:componentInfo>
</upnp:resExt>
")
```

**Response**:

```
UpdateObject()
```

After *UpdateObject()*, The control point invokes the *Browse()* action on the object again to check for the updated property values. Especially the *upnp:resExt::componentInfo::componentGroup::component::compRes:;res* property value is of interest, since it may have been changed by the ContentDirectory implementation.

In the second approach, where the control point does not first create a new object for the audio track but rather adds the component's resource to the object directly, it invokes the *UpdateObject()* action in the following manner:

**Request**:

```
UpdateObject("12", "
<upnp:resExt id="0">
   <upnp:componentInfo>
      <upnp:componentGroup groupID="group_1" required="1">
         <upnp:component componentID="comp_1">
            <upnp:componentClass>
               Audio
            </upnp:componentClass>
            <upnp:contentType MIMEType="audio/ac3"
            extendedType="*"/>
            <upnp:language>
               en-US
            </upnp:language>
         </upnp:component>
      </upnp:componentGroup>
   </upnp:componentInfo>
</upnp:resExt>
", "
<upnp:resExt id="0">
   <upnp:componentInfo>
      <upnp:componentGroup groupID="group_1" required="1">
         <upnp:component componentID="comp_1">
            <upnp:componentClass>
               Audio
            </upnp:componentClass>
            <upnp:contentType MIMEType="audio/ac3"
            extendedType="*"/>
            <upnp:language>
               en-US
            </upnp:language>
         </upnp:component>
         <upnp:component componentID="comp_2">
            <upnp:componentClass>
               Audio
            </upnp:componentClass>
            <upnp:contentType MIMEType="audio/mp3"
            extendedType="*"/>
            <upnp:language>
               fr
            </upnp:language>
            <upnp:compRes>
               <upnp:res protocolInfo="http-get:*:audio/mp3:*" />
            </upnp:component>
      </upnp:componentGroup>
   </upnp:componentInfo>
</upnp:resExt>
")
```

**Response**:

```
UpdateObject()
```

As a new upnp:resExt::componentInfo::componentGroup::component::compRes::res property is being created in UpdateObject(), the control point invokes the Browse() action on the created object again to obtain the upnp:resExt::componentInfo::componentGroup::component::compRes::res@importUri property in order to perform uploading of the content binary:

**Request**:

```
Browse("12", "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="12" parentID="11" restricted="0">
      <dc:title>New Video</dc:title>
      <upnp:class>
         object.item.videoItem
      </upnp:class>
      <res id="0" importUri="http://10.0.0.1/pc/item?id=12"
       protocolInfo ="*:*:video/mpeg:*">
         http://10.0.0.1/pc/item?id=12
      </res>
      <upnp:resExt id="0">
         <upnp:componentInfo>
            <upnp:componentGroup groupID="group_0" required="1">
               <upnp:component componentID="comp_0">
                  <upnp:componentClass>
                     Video
                  </upnp:componentClass>
                  <upnp:contentType MIMEType="video/M2PV"
                  extendedType="*"/>
               </upnp:component>
            </upnp:componentGroup>
            <upnp:componentGroup groupID="group_1" required="1">
               <upnp:component componentID="comp_1">
                  <upnp:componentClass>
                     Audio
                  </upnp:componentClass>
                  <upnp:contentType MIMEType="audio/ac3"
                  extendedType="*"/>
                  <upnp:language>
                     en-US
                  </upnp:language>
                  </upnp:component>
                  <upnp:component componentID="comp_2">
                  <upnp:componentClass>
                     Audio
                  </upnp:componentClass>
                  <upnp:contentType MIMEType="audio/mp3"
                     extendedType="*"/>
                  <upnp:language>
                     fr
                  </upnp:language>
                  <upnp:compRes>
                     <upnp:res protocolInfo="http-get:*:audio/mp3:*"
            importUri="http://10.0.0.1/pc/item?id=12&compId=comp_2"/>

                  </upnp:compRes>
               </upnp:component>
            </upnp:componentGroup>
         </upnp:componentInfo>
      </upnp:resExt>
   </item>
</DIDL-Lite>", 1, 1, 10)
```

## 2.6.13 Segments Manipulation

### 2.6.13.1 Segment Item Example

The following illustrates a typical example of a segment item. It shows the XML fragments for both the segment item and its base content.

```
<!-- Base Content Item-->
<item id="base-content-1" parentID="container-1" restricted="0">
    <dc:title>KBS News 20080909</dc:title>
    <upnp:class>object.item.videoItem</upnp:class>
    <upnp:icon>http://10.0.0.1/logos/KBS-News.jpg</upnp:icon>
    <upnp:segmentID>segment-item-1</upnp:segmentID>
    <res id="base-res-1" protocolInfo="http-get:*:video/mpeg:*" resolution="1920x1080">
        http://10.0.0.1/video/content?id=1
    </res>
    <res id="base-res-2" protocolInfo="http-get:*:video/mpeg:*" resolution="1280x720">
        http://10.0.0.1/video/content?id=2
    </res>
</item>

<!-- Segment Item -->
<item id="segment-item-1" parentID="segment-container-1" restricted="0">
    <dc:title>KBS News 20080909 Sports Section</dc:title>
    <upnp:class>object.item.videoItem</upnp:class>
    <upnp:icon>http://10.0.0.1/logos/KBS-Sports-News.jpg</upnp:icon>

    <upnp:resExt id="segment-res-1">
        <upnp:segmentInfo baseObjectID="base-content-1" baseResID="base-res-1">
            <upnp:timeRange start="00:05:00" end="00:08:00"/>
            <upnp:byteRange start="500000" end="800000"/>
        </upnp:segmentInfo>
    </upnp:resExt>

    <upnp:resExt id="segment-res-2">
        <upnp:segmentInfo baseObjectID="base-content-1" baseResID="base-res-2">
            <upnp:timeRange start="00:05:00" end="00:08:00"/>
        </upnp:segmentInfo>
    </upnp:resExt>

    <res id="segment-res-1" protocolInfo="http-get:*:video/mpeg:*" resolution="1920x1080">
        http://10.0.0.1/video/content?id=1?start=500;end=800
    </res>
    <res id="segment-res-2" protocolInfo="http-get:*:video/mpeg:*" resolution="1280x720">
        http://10.0.0.1/video/content?id=2?start=500;end=800
    </res>
</item>
```

### 2.6.13.2 Creating, Destroying and Updating Segments

A ContentDirectory service implementation may optionally support the creation of segmented content by a control point.

Support for a control point to create segmented content for a given media format is indicated by the *protocolInfo* attribute of *a <segmentCreate>* element of the *SEGMENTATION feature*.

A ContentDirectory service implementation which specifies a *SEGMENTATION feature* without including any *<segmentCreate>* elements is indicating that the implementation can provide segmented content. However, the installation of this content is implementation dependent.

The *CreateObject()* action is used to create a new segment item and the *UpdateObject()* action is used to add segments to an existing item. When creating a new segment item or adding segments to an existing item, the ContentDirectory service implementation will update the submitted segment item *res* properties to access portions of the indicated base item content binary. The detailed value of the segment item *res* property after updating by the ContentDirectory service is implementation dependent.

When a segment item is created or updated, a control point is required to provide a *res* property containing the URI value of the base content item *res* property and an associated *upnp:resExt::segmentInfo* property.

The *upnp:resExt::segmentInfo* property is required to contain a *upnp:timeRange* property indicating the start and end times within the base content binary. A ContentDirectory service implementation may also require a control point to provide additional *upnp:segmentInfo* properties such as *upnp:byteRange* or *upnp:frameRange* to delineate portions of base content binary to be referenced by the segment item *res* property. The ContentDirectory service metadata requirements to create a segment item *res* property are indicated by the *<additionalInfoRequired>* element of the *SEGMENTATION feature*. The *<additionalInfoRequired>* elements indicate whether *upnp:byteRange* or *upnp:frameRange* properties are also required to be provided to create segment item *res* properties for the base content media format indicated by the *<segmentCreate>* element.

The updated *res* property is included in the *Result* output argument of the *CreateObject()* action.

When a segment is created, the base content item is updated so that one of its *upnp:segmentID* properties contains the object ID of the item which contains the newly created segment.

A base content item may contain multiple *res* properties, each of which points to a different resource. Therefore the control point should be able to specify which specific resource that the segment is associated with. To this end, for a base content item which is referred to by one or more segment descriptors, each of its *res* properties should have an item-wide unique *res@id* attribute. Therefore a control point can refer to a specific *res* property in the base content item by using *upnp:resExt::segmentInfo@baseResID* property.

The following is an example of *CreateObject()* action request and response:

**Request:**

```
CreateObject("SegmentContainer_01", "
<?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="SegmentContainer_01" restricted="0">
        <dc:title>Segment1</dc:title>
        <upnp:class>object.item.videoItem</upnp:class>

        <res id="segment-res">http://192.168.0.1/video/my_movie.mpg</res>

        <upnp:resExt id="segment-res">
            <upnp:segmentInfo baseObjectid="base-content" baseResID="base-res">
                <upnp:timeRange start="00:05:00" end="00:08:00"/>
            </upnp:segmentInfo>
        </upnp:resExt>

    </item>
</DIDL-Lite>")
```

**Response:**

```
CreateObject("Segment_01", "
<?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="Segment_01" parentID="SegmentContainer_01" restricted="0">
        <dc:title>Segment1</dc:title>
        <upnp:class>object.item.videoItem</upnp:class>
        <res id="segment-res" protocolInfo="http-get:*:video/mpeg: *">
            http:// 192.168.0.1/video/my_movie.mpeg?start=00:05:00;end=00:08:00
        </res>

        <upnp:resExt id="segment-res">
            <upnp:segmentInfo baseContentId="base-content" baseResID="base-res">
                <upnp:timeRange start="00:05:00" end="00:08:00"/>
            </upnp:segmentInfo>
        </upnp:resExt>

    </item>
</DIDL-Lite>")
```

In the above example, the ContentDirectory service implementation added parameters to the base content URI to locate start/end points of a segment within the base content-binary. This implies that the ContentDirectory service implementation can detect and interpret these parameters when the segment URI is requested from the device. The choice whether to use parameters to indicate offsets within the original content-binary or to create a new content-binary object with a unique URI value is an implementation choice.

The *DestroyObject()* action is used to destroy objects specified by the *ObjectID* argument. When the object being destroyed represents one or more segments, the ContentDirectory service is required to maintain consistency: that is, when a segment descriptor is destroyed upon the destruction of the object, the ContentDirectory service first finds the base content item using the *upnp:resExt::segmentInfo@baseObjectID* property and it removes the associated *upnp:segmentID* property from the base content item.

Similarly, when a content item that contains one or more *upnp:segmentID* properties is destroyed, the ContentDirectory service finds all associated segment items and removes all segment *res* properties and corresponding *upnp:resExt* properties with a *upnp:resExt:segmentInfo@baseObjectID* value that matches the deleted base object.

The *UpdateObject()* action can be used to add, or delete segment *res* properties of an existing segment item. The *UpdateObject()* action results in a segment item containing at least one segment *res* property. Modification or deletion of metadata properties associated with the *upnp:resExt::segmentInfo* property is prohibited.

Segment related operations that result in internal modifications to base-item properties (for example *upnp:segmentID*) should be captured in the *LastChange* state variable if the ContentDirectory service implementation supports the *Tracking Changes Option*.

### 2.6.13.3 Browse and Search Segment Items

Segments and their base contents are associated to each other through the *upnp:resExt::segmentInfo@baseContentID* property of the segment descriptors, and the *upnp:segmentID* property of base content items. This association enables users to browse and search all the segments related to a certain base content. To browse the segments associated with a certain base content, the control point can first get the *upnp:segmentID* properties of the base content item, and then use the *Browse()* action with each *upnp:segmentID*. The control point can also use the *Search()* action to get all segments, of which the *upnp:resExt::segmentInfo@baseObjectID* property equals the object ID of the base content in concern.

## 2.6.14 Bookmark Manipulation

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

### 2.6.14.1 *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>
```

### 2.6.14.2 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()
```

### 2.6.14.3 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)
```

### 2.6.15 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 sections 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 Appendix H, “Example ContentDirectory Hierarchy”.

#### 2.6.15.1 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)
```

### 2.6.15.2 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)
```

### 2.6.15.3 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)
```

## 2.6.16 Foreign Metadata

### 2.6.16.1 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>")
```

### 2.6.16.2 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)
```

## 2.6.17 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.

### 2.6.17.1 Monitoring Changes while *on-line*

#### 2.6.17.1.1 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.

#### 2.6.17.1.2 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 Section 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.

### 2.6.17.2 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.

#### 2.6.17.2.1 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 Section 2.2.5, "*Object Modification*" and Appendix B.19.2, "*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 Section 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.

#### 2.6.17.2.1.1 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 Section 2.6.17.2.1, "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.

#### 2.6.17.2.1.2 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 (Section 2.6.17.2.1.1, "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.

#### 2.6.17.2.1.3 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 (Section 2.6.17.2.1.1, "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.

#### 2.6.17.2.2 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 (Section 2.6.17.2.1, "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

(Section 2.6.17.2.1.1, “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.

#### 2.6.17.2.3 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 (Section 2.6.17.2.1.1, “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.

### 2.6.18 Browsing preserved transitory content

The following scenarios assume that the ContentDirectory service implementation contains a buffer, which is able to temporarily preserve transitory content, which is available now but may not be available in the future, for example a program that is being distributed using traditional broadcast. Several objects in the ContentDirectory service may expose additional properties to indicate that the transitory content that is associated with these objects are currently being preserved. The control point thus gets an indication that some prior history of the transitory content can be included in a recording or played. In the following subsections, examples are given for browsing tuner objects (that is, objects of class *object.item.audioItem.audioBroadcast* or *object.item.videoItem.videoBroadcast*), and finding out whether the content being broadcast is being preserved.

#### 2.6.18.1 Browsing broadcast items with preserved history

The control point retrieves the children of the Tuner container containing *object.item.videoItem.videoBroadcast* objects. 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("TunerContainerID", "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="32" parentID="TunerContainerID" restricted="1">
      <dc:title>NBC</dc:title>
      <upnp:class>
         object.item.videoItem.videoBroadcast
      </upnp:class>
      <upnp:channelName>NBC</upnp:channelName>
      <upnp:channelNr>2</upnp:channelNr>
      <res protocolInfo="http-get:*:video/mpeg:*">
         http://10.0.0.1/getcontent.asp?id=32
      </res>
   </item>
   <item id="33" parentID="TunerContainerID" restricted="1">
      <dc:title>FOX</dc:title>
      <upnp:class>
         object.item.videoItem.videoBroadcast
      </upnp:class>
      <upnp:channelName>FOX</upnp:channelName>
      <upnp:channelNr>3</upnp:channelNr>
      <upnp:preservedTimeRange startTime="2008-12-06T13:59:50"/>
      <upnp:programList>
         <upnp:program preserved="1">60</upnp:program>
         <upnp:program preserved="0">61</upnp:program>
      </upnp:programList>
      <res protocolInfo="http-get:*:video/mpeg:*">
         http://10.0.0.1/getcontent.asp?id=33
      </res>
   </item>
</DIDL-Lite>", 2, 100, 50)
```

It can be seen that some portion of the content in the past which is being broadcast on the channel with the name “FOX” (item “33”) is being preserved by the ContentDirectory service (starting from 13:59:50 on 6th December 2008 as indicated by the value of the *upnp:preservedTimeRange@startTime* property). It is therefore possible to make a recording of the current channel content including the past portion, starting at any time after the time indicated by this property. Additionally, the *upnp:programList* property identifies a program that is currently being temporarily preserved (EPG object with object ID “60”), and a program that is not preserved. When recording content on the channel with the name “NBC” (item “32”), it is not possible to start recording from a time earlier than the current time, since no properties are present indicating that past content is being preserved.

### 2.6.18.2 Browsing program items indicating preserved history (EPG data available but not exposed to control point)

In this example, the ContentDirectory service has access to EPG data, but does not expose this to the control point via *object.item.epgItem* objects and the *EPG feature*. The ContentDirectory service MAY create special *object.item.epgItem* objects as programs are being preserved. These objects will typically have a minimal amount of metadata present. If the *Track Changes Option* is supported, control points can be notified of these new objects and perform a *Browse()* action on them. In this case, the result of the *Browse()* action may be as below:

**Request**:

```
Browse("50", "BrowseMetadata", "*", 0, 1, "")
```

**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="50" parentID="EPGContainerID" restricted="1">
      <dc:title>CNN News</dc:title>
      <upnp:class>
         object.item.epgItem
      </upnp:class>
      <upnp:channelNr>1</upnp:channelNr>
      <upnp:scheduledStartTime>
         2008-12-06T13:50:00
      </upnp:scheduledStartTime>
      <upnp:scheduledEndTime>
         2008-12-06T14:40:00
      </upnp:scheduledEndTime>
      <upnp:programPreserved startTime="2008-12-06T13:59:50">
         ONGOING
      </upnp:programPreserved>
   </item>
</DIDL-Lite>", 1, 1, 50)
```

It can be seen from the value of the *upnp:programPreserved* property that the program named "CNN News" on channel number 1 is being preserved. From the presence and the value of the *upnp:programPreserved@startTime* property, it is concluded that the beginning portion of the program has not been preserved. The control point can use this EPG object to create a record schedule, but it can expect the result of the recording to be a partial recording.

### 2.6.18.3 Browsing program items for recording (EPG exposed to control point)

In the scenario described in this section, the control point browses a certain EPG container and wants to use an EPG object for creating a scheduled recording. Depending on whether the content associated with a certain EPG object is being/has been preserved, different combinations of values for the *upnp:programPreserved*, *upnp:programPreserved@startTime* and *upnp:programPreserved@endTime* properties may exist. The following example illustrate the possible combinations; it is assumed that the ContentDirectory service implementation supports buffering of transitory data.

**Request**:

```
Browse("EPGContainerID", "BrowseDirectChildren", "*", 0, 10, "")
```

**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="60" parentID="EPGContainerID" restricted="1">
      <dc:title>CNN business news</dc:title>
      <upnp:class>
         object.item.epgItem
      </upnp:class>
      <upnp:channelName>CNN</upnp:channelName>
      <upnp:channelNr>1</upnp:channelNr>
      <upnp:scheduledStartTime>
         2008-12-06T13:50:00
      </upnp:scheduledStartTime>
      <upnp:scheduledEndTime>
         2008-12-06T14:40:00
      </upnp:scheduledEndTime>
      <upnp:programPreserved startTime="2008-12-06T13:59:50">
         ONGOING
      </upnp:programPreserved>
   </item>
   <item id="61" parentID="EPGContainerID" restricted="1">
      <dc:title>NBC business news</dc:title>
      <upnp:class>
         object.item.epgItem
      </upnp:class>
      <upnp:channelName>NBC</upnp:channelName>
      <upnp:channelNr>2</upnp:channelNr>
      <upnp:scheduledStartTime>
         2008-12-06T13:50:00
      </upnp:scheduledStartTime>
      <upnp:scheduledEndTime>
         2008-12-06T14:40:00
      </upnp:scheduledEndTime>
      <upnp:programPreserved>ONGOING</upnp:programPreserved>
   </item>
   <item id="62" parentID="EPGContainerID" restricted="1">
      <dc:title>ABC business news</dc:title>
      <upnp:class>
         object.item.epgItem
      </upnp:class>
      <upnp:channelName>ABC</upnp:channelName>
      <upnp:channelNr>3</upnp:channelNr>
      <upnp:scheduledStartTime>
         2008-12-06T13:50:00
      </upnp:scheduledStartTime>
      <upnp:scheduledEndTime>
         2008-12-06T14:40:00
      </upnp:scheduledEndTime>
      <upnp:programPreserved startTime="2008-12-06T13:59:50">
         COMPLETED
      </upnp:programPreserved>
   </item>
   <item id="63" parentID="EPGContainerID" restricted="1">
      <dc:title>FOX business news</dc:title>
      <upnp:class>
```

```
         object.item.epgItem
      </upnp:class>
      <upnp:channelName>FOX</upnp:channelName>
      <upnp:channelNr>4</upnp:channelNr>
      <upnp:scheduledStartTime>
         2008-12-06T13:50:00
      </upnp:scheduledStartTime>
      <upnp:scheduledEndTime>
         2008-12-06T14:40:00
      </upnp:scheduledEndTime>
      <upnp:programPreserved endTime="2008-12-06T14:30:10">
         COMPLETED
      </upnp:programPreserved>
   </item>
   <item id="64" parentID="EPGContainerID" restricted="1">
      <dc:title>BBC business news</dc:title>
      <upnp:class>
         object.item.epgItem
      </upnp:class>
      <upnp:channelName>BBC</upnp:channelName>
      <upnp:channelNr>5</upnp:channelNr>
      <upnp:scheduledStartTime>
         2008-12-06T13:50:00
      </upnp:scheduledStartTime>
      <upnp:scheduledEndTime>
         2008-12-06T14:40:00
      </upnp:scheduledEndTime>
      <upnp:programPreserved>COMPLETED</upnp:programPreserved>
   </item>
</DIDL-Lite>", 5, 5, 50)
```

**Program preserved, no beginning:**

The program associated with the EPG object with object ID “60” is currently being preserved in the buffer, however the start of the program is not preserved. This can be seen from the presence of the *upnp:programPreserved@startTime* property. The value of the *upnp:programPreserved* property (“*ONGOING*”) indicates that the program is still being broadcast and is still being accumulated by the ContentDirectory service.

**Program preserved, starting from beginning:**

The program associated with the EPG object with object ID “61” is currently being preserved in the buffer, including the start of the program (the *upnp:programPreserved@startTime* property is absent). The value of the *upnp:programPreserved* property is ”*ONGOING*”, which means that the program is still being accumulated.

**Program finished being preserved, no beginning:**

The program associated with the EPG object with object ID “62” has already finished and a portion of it has been preserved, except the start of the program. This can be seen from the value of the *upnp:programPreserved* property (“*COMPLETED*”) and the presence of the *upnp:programPreserved@startTime* property.

**Program finished being preserved, no end:**

The program associated with the EPG object with object ID “63” has been preserved from the beginning. It is still being broadcast, however the ContentDirectory service stopped preserving it prematurely. This can be seen from the value of the *upnp:programPreserved* property (“*COMPLETED*”), and the presence of the *upnp:programPreserved@endTime* property.

**Program finished, preserved entirely:**

The program associated with the EPG object with object ID “64” has finished, and has been completely preserved from beginning to end. This can be seen from the value of the *upnp:programPreserved* property (“*COMPLETED*”), and the absence of the *upnp:programPreserved@startTime* and *upnp:programPreserved@endTime* properties.

## 2.6.19 Object Linking

### 2.6.19.1 Displaying Object Link titles

Object Linked lists can be connected together in a hierarchy of lists. An object representing the collection of lists is pointed to by a *upnp:objectLink::startInfo* property from the head item of any list in the collection. The item (described below) contains a *upnp:objectLinkRef* property that indicates the initial GroupID and Item *@id* of the start of the collection of related lists.

Control points can use the *Search()* action (if implemented) to locate items representing the start of collections of lists. These objects contain a *upnp:objectLink::startObject* property with a value of ”*1*”.

If the *Search()* action is not implemented control points can recognize objects representing collections of lists by the presence of a *upnp:objectLink::startObject* property with a value of ”*1*”.

The following is an example of a *Search()* action to locate starting items:

```
Request:
Search(  "0",
         "upnp:objectLinkRef::startObject = "1"",
         "*",
         0,
         0,
         "+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="start0001" parentID="game0001" restricted="1">
  <dc:title>Super Bowl XLIII - First Quarter</dc:title>
  <upnp:class>object.item.imageItem</upnp:class>
  <upnp:objectLinkRef groupID="IndexGrp" targetGroupID="IndexGrp"
                      targetObjID="idx0001" >
    <upnp:startObject>1</upnp:startObject>
  </upnp:objectLinkRef>
  <res protocolInfo="http-get:*:image/jpeg:*">
       http://192.168.0.10/images/SteelersVsCardinals.jpeg
  </res>
</item>
</DIDL-Lite>" );
```

### 2.6.19.2 Locating the head Object Link property of an Object Linked list

When a control point encounters an item with a upnp:objectLink property it can locate the head upnp:objectLink property for the list of object linked items by the following procedure:

1. Obtain and remember the property value of upnp:objectLink@groupID for the upnp:objectLink property,
2. Browse() the item indicated by the upnp:objectLink@headObjID property.
3. Select the upnp:objectLink property in the item (from Step 2) with a upnp:objectLink@groupID that matches group ID obtained in Step 1.

### 2.6.19.3 Starting an Object Linked presentation

When starting an Object Linked presentation, the upnp:objectLink::mode property of the initial list determines the startup processing. Please see the upnp:objectLink::mode property for a description of types of object linked lists.

### 2.6.19.4 Object Linking Example

This example demonstrates presentation of a football game.

#### 2.6.19.4.1 Starting Item processing

A control point can display a starting item which describes the collection of object linked lists. The control point can display the title and/or image associated with the item on the control point local UI display or on a rendering device the control point selects. This item is identified as a starting item by the presence of a upnp:objectLinkRef::startObject property with value "1"

If the end-user selects this item for playback, the upnp:objectLinkRef@targetGroupID and upnp:objectLinkRef@targetObjID properties of this item identify an initial item and group ID.

In this example the initial item is "indx0001" and the starting group ID is "IndexGrp".

The control point first obtains this item using the Browse() action and locates a upnp:objectLink property with a upnp:objectLink@groupID value of "IndexGrp".

The control point then inspects the upnp:objectLink@headObjID property to determine @id of the head item in the object link list. This may be the initial item or may be a different item.

The control point obtains the head item of the object link list and locates an upnp:objectLink property with a upnp:objectLink@groupID value of "IndexGrp". The upnp:objectLink::mode property in this item identifies the type of list to be started.

In this example, the presentation starts with an Index list consisting of four items representing the four quarters of the football game. The media content of these Index list items provide JPEG thumbnails and brief video segments of the game intended for display on the control point or associated rendering device.

The Index list items link to other items which provide the full video for the game.

The playback items also provide links to lists of images of key plays the end-user may choose to display at some point during the playback of the video of the game.

All the video content is derived from segments of a base video item that contains the entire game as a single video object.

### 2.6.19.4.2 Initial Object Link list processing

In this example, the initial list type is an Index list as indicated by the *upnp:objectLink::mode* property value of "*Index*".

The control point will usually start with an initial item that is not necessarily the head item of the Index list. This item is typically the item that the control point displays for playback if the end-user does not change the playback list selection.

The control point performs the following steps to process the Index list:

1. Obtain the items in the Index list:

   The items in the Index list are determined as follows:

   a. The control point locates the initial item *upnp:objectLink* property with a *upnp:objectLink@groupID* value matching the current group ID value.

   b. The control point obtains the *@id* property of the head item of the list using the value of the *upnp:objectLink@headObjID* property.

   c. The control point issues a *Browse()* action to obtain the head list item metadata.

   d. The control point locates the *upnp:objectLink* property in the head item metadata with a *upnp:objectLink@groupID* value that matches the current group ID.

   e. The control point selects the next Index list item *@id* to browse to the value of the *upnp:objectLink@nextObjID* and follows Step c-d to locate the remaining items in the Index list.

2. Positioning Index list items for display:

   a. The initial Index list item is typically intended to be the item the control point will display for playback.

   b. The control point can choose to highlight this item.

   c. In most cases the control point will choose to center this item on end-user display and can display items prior to and succeeding this initial item.

   d. The control point will typically provide a means to allow the end-user to select a different Index list item for playback.

The control point has considerable flexibility in displaying members of the Index list to the end-user. The control point can:

- List the title(s) (*dc:title* property) of members of the Index list.
- Provide a preview window on the control point's user-interface to display the Image or AV content binaries referenced by the item.
- Display the AV or Image content preview on an associated rendering device.

**Starting item**

```
<item id="start0001" parentID="game0001" restricted="1">
   <dc:title>Super Bowl XLIII - First Quarter</dc:title>
   <upnp:class>object.item.imageItem</upnp:class>
   <upnp:objectLinkRef groupID="IndexGrp" targetGroupID="IndexGrp"
                       targetObjID="idx0001">
      <upnp:startObject>1</upnp:startObject>
   </upnp:objectLinkRef>
   <res protocolInfo="http-get:*:image/jpeg:*">
        http://192.168.0.10/images/SteelersVsCardinals.jpeg
   </res>
</item>
```

**First Index item**

```
<item id="idx0001" parentID="game0001" restricted="1">
   <dc:title>Super Bowl XLIII - First Quarter</dc:title>
   <upnp:class>object.item</upnp:class>
   <upnp:objectLink groupID="IndexGrp" nextObjID="idx0002" prevObjID=""
                    headObjID="idx0001">
      <upnp:title>
          Super Bowl XLIII – Pittsburg Steelers v Arizona Cardinals
      </upnp:title>
      <upnp:mode>Index</upnp:mode>
      <upnp:startInfo targetGroupID="IndexGrp"
                      targetObjID="start0001" />
   </upnp:objectLink>
   <upnp:objectLinkRef groupID="IndexGrp" targetGroupID="PlayGrp"
                       targetObjID="pb0001" />
   <res id="res0001" protocolInfo="http-get:*:video/mpeg:*">
       http://192.168.0.10/video/football.mpg?start=600;end=610
   </res>
   <upnp:resExt id="res0001">
      <upnp:segmentInfo baseObjectID="videobase" baseResID="res0001">
          <upnp:timeRange start="00:10:00" end="00:10:10" />
      </upnp:segmentInfo>
   </upnp:resExt>
  <res id="res0002" protocolInfo="http-get:*:image/jpeg:*">
   http://192.168.0.10/photo/football1Q.jpeg
  </res>
</item>
```

**Second Index item**

```
<item id="idx0002" parentID="game0001" restricted="1">
   <dc:title>Second Quarter</dc:title>
   <upnp:class>object.item</upnp:class>
   <upnp:objectLink groupID="IndexGrp" nextObjID="idx0003"
                    prevObjID="idx0001" headObjID="idx0001" />
   <upnp:objectLinkRef groupID="IndexGr" targetGroupID="PlayGrp"
                       targetObjID="pb0002" />
   <res id="res0001" protocolInfo="http-get:*:video/mpeg:*">
      http://192.168.0.10/video/football.mpg?start=960;end=970
   </res>
   <upnp:resExt id="res0001">
      <upnp:segmentInfo baseObjectID="videobase" baseResID="res0001">
         <upnp:timeRange start="00:16:00" end="00:16:10" />
      </upnp:segmentInfo>
   </upnp:resExt>
   <res id="res0002" protocolInfo="http-get:*:image/jpeg:*">
    http://192.168.0.10/photo/football1Q.jpeg
   </res>
</item>
```

**Third Index item**

```
<item id="idx0003" parentID="game0001" restricted="1">
   <dc:title>Third Quarter</dc:title>
   <upnp:class>object.item</upnp:class>
   <upnp:objectLink groupID="IndexGrp" nextObjID="idx0003"
                    prevObjID="idx0001" headObjID="idx0001" />
   <upnp:objectLinkRef groupID="IndexGrp" targetGroup="PlayGrp"
                       targetObjID="pb0003" />
   <res id="res0001" protocolInfo="http-get:*:video/mpeg:*">
       http://192.168.0.10/video/football.mpg?start=1920;end=1930
   </res>
   <upnp:resExt id="res0001">
      <upnp:segmentInfo baseObjectID="videobase" baseResID="res0001">
        <upnp:timeRange start="00:32:00" end="00:32:10" />
     </upnp:segmentInfo>
   </upnp:resExt>
   <res id="res0002" protocolInfo="http-get:*:image/jpeg:*">
      http://192.168.0.10/photo/football3Q.jpeg
   </res>
</item>
```

**Fourth Index item**

```
<item id="idx0004" parentID="game0001" restricted="1">
   <dc:title>Fourth Quarter</dc:title>
   <upnp:class>object.item</upnp:class>
   <upnp:objectLink groupID="IndexGrp" nextObjID="" prevObjID="idx0003"
                    headObjID="idx0001" />
   <upnp:objectLinkRef groupID="IndexGrp" targetGroup="PlayGrp"
                       targetObjID="pb0004" />
   <res id="res0001" protocolInfo="http-get:*:video/mpeg:*">
       http://192.168.0.10/video/football.mpg?start=2700;end=2710
   </res>
   <upnp:resExt id="res0001">
      <upnp:segmentInfo baseObjectID="videobase" baseResID="res0001">
         <upnp:timeRange start="00:45:00" end="00:45:10" />
      </upnp:segmentInfo>
   </upnp:resExt>
   <res id="res0002" protocolInfo="http-get:*:image/jpeg:*">
      http://192.168.0.10/photo/football3Q.jpeg
   </res>
</item>
```

### 2.6.19.4.3 Selection of an Object Link list Index item

Suppose the end-user selects the 2nd Index list item *@id* ”idx0002” displayed by the control point for playback.

The control point typically performs the following steps:

1. Identify the appropriate *upnp:objectLinkRef* instance with a *upnp:objectLinkRef@groupID* value of “IndexGrp”.

2. Determine new group ID and item *@id* values.

   The control point can obtain new group ID and item *@id* values from the *upnp:objectLinkRef@targetGroupID* property with value “PlayGrp” and the *upnp:objectLinkRef@targetObjID* property with value “pb0002” identified in Step 1.

3. Browse the item identified by the *upnp:objectLinkRef@targetObjID* property with value “pb0002” determined in Step 2.

4. Determine the *@id* value of the head item of the target list to determine the target list type.

The control point inspects the item from Step 3 and locates the *upnp:objectLink* property instance with a *upnp:objectLink@groupID* property matching the group ID value “PlayGrp”. The *upnp:objectLink@headObjID* property with value “pb0001” provides the *@id* value for the item at the head of the target list.

5. *Browse()* the item at the head of the target list with *@id* value “pb0001”.
6. Obtain the list type of the target list.

   The control point inspects the item from Step 5 and locates the *upnp:objectLink* property instance with a *upnp:objectLink@groupID* property value that matches the group ID “PlayGrp”. The type of list is obtained from the value of the *upnp:objectLink::mode* property which is “*Playback*” in this example.

In this example the control point has determined that the target list is a “*Playback*” type list as indicated by the head item object ID “pb0001” of the list. Starting with list item with object ID “pb0002”, the control point would play these items on the rendering device.

### 2.6.19.4.4 Playback with references to other lists.

In the current example, item *@id* “pb0002” contains a *upnp:objectLinkRef* property that is also in the “PlayGrp” group. This is an example of a list reference.

To process list references in playback items, the control point can:

- Provide a means to alert the end-user that references for items are available during playback.
- Provide for the display of the text-based titles of reference lists if the end-user requests reference titles to be displayed.
- Allow the user to select a displayed reference title, causing the referenced list to be played.

However, the specific user interface to convey list references to the end-user is left to the control point implementation.

The control point implementation can choose to:

- Display reference list titles as they become available during playback. This model is useful when the control point display is separate from the rendering device.
- Provide an “unobtrusive” indicator, for example a small icon, on the rendering device display to indicate that the user can request the display of reference titles. This model is useful when the control point display is integrated with rendering device.
- Pause playback when the end-user requests displaying of reference titles.

A list reference is identified by a *upnp:objectLinkRef* property with a *upnp:objectLinkRef@groupID* value matching the group ID “PlayGrp”.

In the current example, item *@id* “pb0002” contains a *upnp:objectLinkRef* property with *upnp:objectLinkRef@groupID* value of “PlayGrp”. The new group ID is provided by the value of the *upnp:objectLinkRef@targetGroupID* property with value “Photos2QGrp” and the new item *@id* is provided by the value of the *upnp:objectLinkRef@targetObjID* property with value of “Photo0002”. In this example the referenced object link list points a Playback list of still photos of the game.

**First Quarter Playback Item**

```
<item id="pb0001" parentID="game0001" restricted="1">
   <dc:title>First Quarter</dc:title>
   <upnp:class>object.item.videoItem</upnp:class>
   <upnp:objectLink groupID="PlayGrp" nextObjID="pb0002" prevObjID=""
                    headObjID="pb0001">
```

```
      <upnp:title>Superbowl Highlights</upnp:title>
      <upnp:mode>Playback</upnp:mode>
      <upnp:startInfo targetGroupID="Index" targetObjID="start0001" />
      <upnp:endAction action="Return" />
   </upnp:objectLink>
   <upnp:objectLinkRef groupID="PlayGrp "
                       return="1"
                       targetGroup="Photos1QGrp"
                       targetObjID="Photo0001" />
   <res id="res0001" protocolInfo="http-get:*:video/mpeg:*">
      http://192.168.0.10/video/football.mpg?start=0;end=960
   </res>
   <upnp:resExt id="res0001">
      <upnp:segmentInfo baseObjectID="videobase" baseResID="res0001">
         <upnp:timeRange start="00:00:00" end="00:16:00" />
      </upnp:segmentInfo>
   </upnp:resExt>
</item>
```

**Second Quarter Playback Item**

```
<item id="pb0002" parentID="game0001" restricted="1">
   <dc:title>Second Quarter</dc:title>
   <upnp:class>object.item.videoItem</upnp:class>
   <upnp:objectLink groupID="PlayGrp" nextObjID="pb0003"
                    prevObjID="pb0001" headObjID="pb0001" />
   <upnp:objectLinkRef groupID="PlayGrp"
                       return="1"
                       targetGroup="Photos2QGrp"
                       targetObjID="Photo0002" />
   <res id="res0001" protocolInfo="http-get:*:video/mpeg:*">
       http://192.168.0.10/video/football.mpg?start=960;end=1920
   </res>
   <upnp:resExt id="res0001">
      <upnp:segmentInfo baseObjectID="videobase" baseResID="res0001">
         <upnp:timeRange start="00:16:00" end="00:32:00" />
      </upnp:segmentInfo>
   </upnp:resExt>
</item>
```

**Third Quarter Playback Item**

```
<item id="pb0003" parentID="game0001" restricted="1">
   <dc:title>Third Quarter</dc:title>
   <upnp:class>object.item.videoItem</upnp:class>
   <upnp:objectLink groupID="PlayGrp" nextObjID="pb0004"
                    prevObjID="pb0002" headObjID="pb0001" />
   <upnp:objectLinkRef groupID="PlayGrp"
                       return="1"
                       targetGroup="Photos3QGrp"
                       targetObjID="Photo0003" />
   <res id="res0001" protocolInfo="http-get:*:video/mpeg:*">
       http://192.168.0.10/video/football.mpg?start=1920;end=2700
   </res>
   <upnp:resExt id="res0001">
      <upnp:segmentInfo baseObjectID="videobase" baseResID="res0001">
         <upnp:timeRange start="00:32:00" end="00:45:00" />
```

```
     </upnp:segmentInfo>
   </upnp:resExt>
</item>
```

### Fourth Quarter Playback Item

```
<item id="pb0004" parentID="game0001" restricted="1">
   <dc:title>Fourth Quarter</dc:title>
   <upnp:class>object.item.videoItem</upnp:class>
   <upnp:objectLink groupID="PlayGrp" nextObjID="" prevObjID="pb0003"
                    headObjID="pb0001" />
   <upnp:objectLinkRef groupID="PlayGrp"
                       return="1"
                       targetGroup="Photos4QGrp"
                       targetObjID="Photo0004" />
   <res id="res0001" protocolInfo="http-get:*:video/mpeg:*">
       http://192.168.0.10/video/football.mpg?start=2700;end=3600
   </res>
   <upnp:resExt id="res0001">
      <upnp:segmentInfo baseObjectID="videobase" baseResID="res0001">
         <upnp:timeRange start="00:45:00" end="01:00:00" />
      </upnp:segmentInfo>
   </upnp:resExt>
</item>
```

### First Quarter Photos

```
<item id="Photo0001" parentID="game0001" restricted="1">
   <dc:title>First Quarter Play</dc:title>
   <upnp:class>object.item.imageItem</upnp:class>
   <upnp:objectLink groupID="Photo1QGrp" nextObjID="" prevObjID=""
                    headObjID="Photo0001">
      <upnp:title>1Q Highlights</upnp:title>
      <upnp:mode>Step</upnp:mode>
      <upnp:startInfo targetGroupID="IndexGrp"
                      targetObjID="start0001" />
      <upnp:endAction action="Return" />
   </upnp:objectLink>
   <res id="res0001" protocolInfo="http-get:*:image/jpeg:*">
       http://192.168.0.10/photos/photo1Q.jpeg
   </res>
</item>
```

### Second Quarter Photos

```
<item id="Photo0002" parentID="game0001" restricted="1">
   <dc:title>First Quarter Play</dc:title>
   <upnp:class>object.item.imageItem</upnp:class>
   <upnp:objectLink groupID="Photo2QGrp" nextObjID="" prevObjID=""
                    headObjID="Photo0002">
      <upnp:title>2Q Highlights</upnp:title>
      <upnp:mode>Step</upnp:mode>
      <upnp:startInfo targetGroupID="IndexGrp"
                      targetObjID="start0001" />
      <upnp:endAction action="Return" />
   </upnp:objectLink>
   <res id="res0001" protocolInfo="http-get:*:image/jpeg:*">
       http://192.168.0.10/photos/photo2Q.jpeg
   </res>
</item>
```

### Third Quarter Photos

```
<item id="Photo0003" parentID="game0001" restricted="1">
   <dc:title>3Q Highlights</dc:title>
   <upnp:class>object.item.imageItem</upnp:class>
   <upnp:objectLink groupID="Photo3QGrp" nextObjID="" prevObjID=""
                    headObjID="Photo0003">
      <upnp:title>3Q Highlights</upnp:title>
      <upnp:mode>Step</upnp:mode>
      <upnp:startInfo targetGroupID="IndexGrp" targetObjID="idx0001" />
      <upnp:endAction action="Return" />
   </upnp:objectLink>
   <res id="res0001" protocolInfo="http-get:*:image/jpeg:*">
       http://192.168.0.10/photos/photo3Q.jpeg
   </res>
</item>
```

### Fourth Quarter  Photos

```
<item id="Photo0004" parentID="game0001" restricted="1">
   <dc:title>First Quarter Play</dc:title>
   <upnp:class>object.item.imageItem</upnp:class>
   <upnp:objectLink groupID="Photo4QGrp" nextObjID="" prevObjID=""
                    headObjID="Photo0004">
      <upnp:title>4Q Highlights</upnp:title>
      <upnp:mode>Step</upnp:mode>
      <upnp:startInfo targetGroupID="IndexGrp" targetObjID="idx0001" />
      <upnp:endAction action="Return" />
   </upnp:objectLink>
   <res id="res0001" protocolInfo="http-get:*:image/jpeg:*">
       http://192.168.0.10/photos/photo4Q.jpeg
   </res>
</item>
```

### Base Video Item

```
<item id="videobase" parentID="game0001" restricted="1">
   <dc:title>Football Game</dc:title>
   <upnp:class>object.item.videoItem</upnp:class>
   <upnp:segmentID>idx0001</upnp:segmentID>
   <upnp:segmentID>idx0002</upnp:segmentID>
   <upnp:segmentID>idx0003</upnp:segmentID>
   <upnp:segmentID>idx0004</upnp:segmentID>
   <upnp:segmentID>pb0001</upnp:segmentID>
   <upnp:segmentID>pb0002</upnp:segmentID>
   <upnp:segmentID>pb0003</upnp:segmentID>
   <upnp:segmentID>pb0004</upnp:segmentID>
   <res id="res0001" protocolInfo="http-get:*:video/mpeg:">
       http://192.168.0.10/video/football.mpg
   </res>
</item>
```

### 2.6.20 *DEVICE_MODE feature*

The *DEVICE_MODE feature* and the associated suite of actions (*RequestDeviceMode()*, *ExtendDeviceMode()*, *CancelDeviceMode()*, *GetDeviceModeStatus()*, and *GetDeviceModeID()*) allow a control point to inform a ContentDirectory service, that the control point would like to place the device in, or remove the device from a special operating mode. The two such modes currently defined are *ActionBurst mode* and *ExclusiveOwnership mode*. When in one of these modes, the ContentDirectory service implementation makes special accommodations, such as, reserving resources, pre-allocating memory, or restricting non-related actions, services, or applications.

Note that while the *DEVICE_MODE feature* is defined within the ContentDirectory service, the support of this feature may impact the entire device including out-of-scope components needed to allocate resources for the device.

### 2.6.20.1 Initiating and Managing *ActionBurst* mode

A typical *ActionBurst* will start with a request to establish the *ActionBurst mode* on the device and then a series of interactions within specific time windows between the ContentDirectory service and the configuring control point. This can include additional extensions or a cancellation of the *ActionBurst mode*. Figure 3 illustrates the handshaking that occurs during the *ActionBurst mode*.

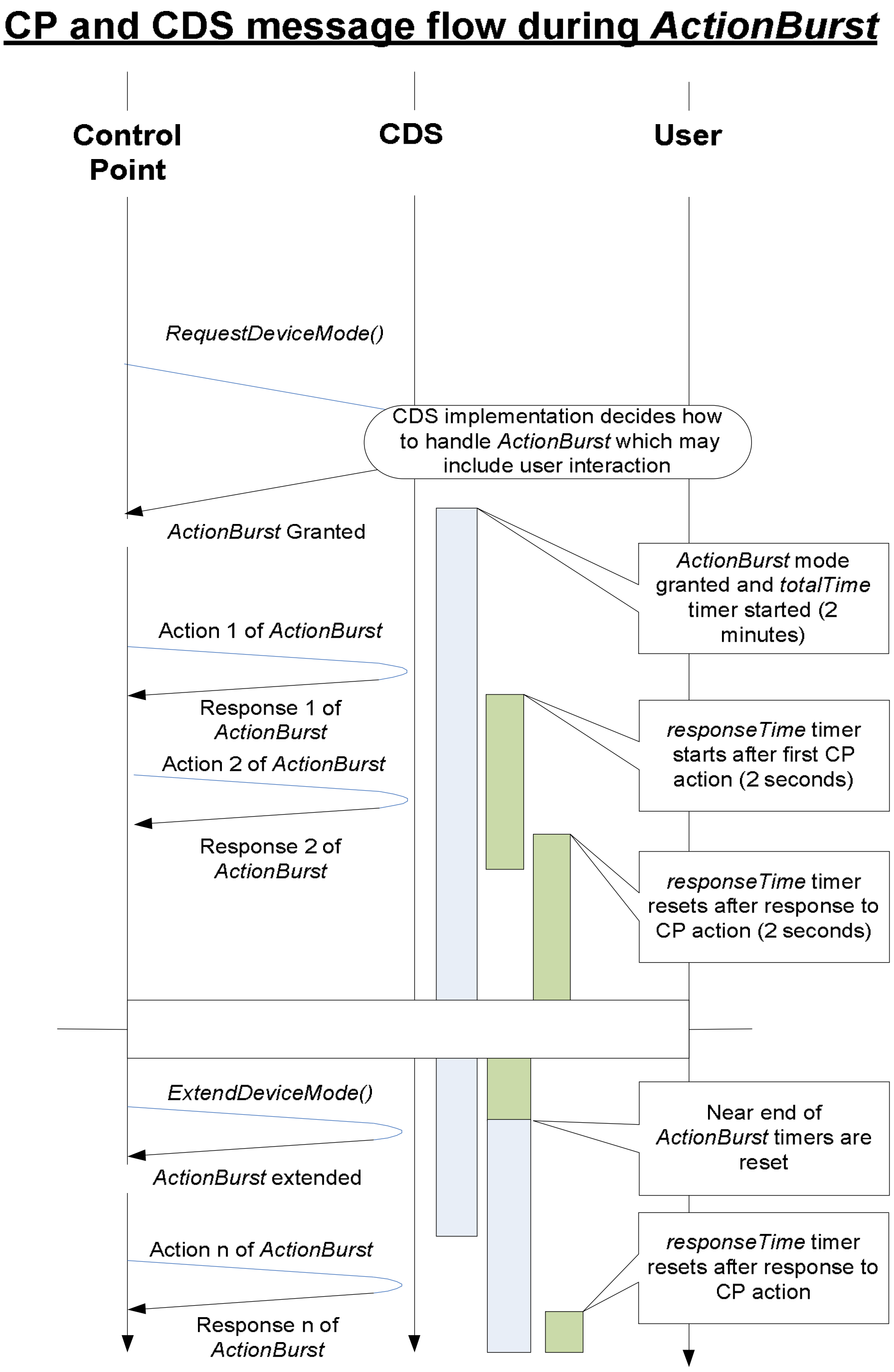

CP and CDS message flow during ActionBurst
Control Point
CDS
User
RequestDeviceMode()
CDS implementation decides how to handle ActionBurst which may include user interaction
ActionBurst Granted
ActionBurst mode granted and totalTime timer started (2 minutes)
Action 1 of ActionBurst
Response 1 of ActionBurst
responseTime timer starts after first CP action (2 seconds)
Action 2 of ActionBurst
Response 2 of ActionBurst
responseTime timer resets after response to CP action (2 seconds)
ExtendDeviceMode()
ActionBurst extended
Near end of ActionBurst timers are reset
Action n of ActionBurst
responseTime timer resets after response to CP action
Response n of ActionBurst

**Figure 3: Example Handshaking for *DEVICE_MODE feature***

In this example, a control point wants to get *ActionBurst* mode for a content transfer task. First, it invokes the GetDeviceMode() action and checks if there is a current *ActionBurst*; being a well behaved control point it does not want to intrude on a current *ActionBurst*. Since there is no current *ActionBurst* it proceeds with its request. In this case the control point is relatively unsophisticated and does not have a good estimate of what kind of time the *ActionBurst* will take and in fact does not know the exact set of actions it will be invoking. Therefore, it leaves the values related to the `totalTime` and `responseTime` requested values empty and includes no <`actionName`> elements. However it does have a descriptive name which could be presented as part of a request to the ContentDirectory service implementation user. In essence the control point is relying on the ContentDirectory service implementation to use some default mechanism for granting *ActionBurst mode*. Thus the control point invokes *RequestDeviceMode()* as follows:

**Request**:

```
RequestDeviceMode(
   "CP-Keith-mobile-DEC-03-2009-10:31:04:06",
   "<?xml version="1.0" encoding="UTF-8"?>
    <DeviceModeRequest
      xmlns="urn:schemas-upnp-org:av:dmor"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
      urn:schemas-upnp-org:av:dmor
      http://www.upnp.org/schemas/av/dmor.xsd">
      <actionBurstRequest>
         <totalTime></totalTime>
         <responseTime></responseTime>
         <label>SyncMusic1</label>
         <description>Keith wants to sync his
         music to your Cell</description>
      </actionBurstRequest>
   </DeviceModeRequest>"
)
```

An alternate request to the above, when the server supports processing of the <`actionName`> element*GetFeatureList()* action, would include a list of the actual actions for the *ActionBurst* (see below).

**Request**:

```
RequestDeviceMode(
   "CP-Keith-mobile-DEC-03-2009-10:31:04:06",
   "<?xml version="1.0" encoding="UTF-8"?>
    <DeviceModeRequest
      xmlns="urn:schemas-upnp-org:av:dmor"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
      urn:schemas-upnp-org:av:dmor
      http://www.upnp.org/schemas/av/dmor.xsd">
      <actionBurstRequest>
         <totalTime></totalTime>
         <responseTime></responseTime>
         <label>SyncMusic1</label>
         <description>Keith wants to sync his
         music to your Cell</description>
         <actionName count="100">CreateObject</actionName>
         <actionName count="100" size="140000000">
            ImportResource
         </actionName>
      </actionBurstRequest>
   </DeviceModeRequest>"
)
```

**Response**:

```
RequestDeviceMode(
   "CPkeithMobile-QRX4-DEC-03-2009-10:31:04:06",
   "<?xml version="1.0" encoding="UTF-8"?>
    <DeviceModeStatus
      xmlns="urn:schemas-upnp-org:av:dmos"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
      urn:schemas-upnp-org:av:dmos
         http://www.upnp.org/schemas/av/dmos.xsd">
      <actionBurstStatus>
         <totalTime>120000</totalTime>
         <responseTime>2000</responseTime>
      </actionBurstStatus>
   </DeviceModeStatus>"
)
```

As a result the ContentDirectory service implementation grants the control point an *ActionBurst mode* for 120 seconds (2 minutes) and expects the control point actions to occur within 2 seconds of responding to the previous request and generates the following responses.

The granted times are returned in the *RequestDeviceMode()* output arguments along with a ContentDirectory service assigned ID valid for this *ActionBurst*.

The *DeviceMode* state variable changes to the *ActionBurst* mode and is evented and the *DeviceModeStatus* state variable is immediately updated and begins counting down the *DeviceModeStatus* state variable *totalTime* value (see below).

**Event**:
*DeviceMode* state variable contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceMode
 xmlns="urn:schemas-upnp-org:av:dmo"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:dmo
      http://www.upnp.org/schemas/av/dmo.xsd">
   <mode type="ActionBurst" CPRequest="1"></mode>
</DeviceMode>
```

**Request**:
```
GetDeviceModeStatus()
```

**Response**:
```
GetDeviceModeStatus("
<?xml version="1.0" encoding="UTF-8"?>
<DeviceModeStatus
   xmlns="urn:schemas-upnp-org:av:dmos"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
   urn:schemas-upnp-org:av:dmos
      http://www.upnp.org/schemas/av/dmos.xsd">
   <actionBurstStatus>
      <totalTime>120000</totalTime>
      <responseTime>2000</responeTime>
   </actionBurstStatus>
</DeviceModeStatus>
);
```

Near the end of the two minutes, when the *ActionBurst* mode is set to expire, suppose 15 seconds, the control point realizes that it will likely not complete its task and requests an extension of its *ActionBurst mode* by invoking the *ExtendDeviceMode()* action with the following input parameter.

```
Request:
ExtendDeviceMode(
   "CPkeithMobile-QRX4-DEC-03-2009-10:31:04:06",
   "<?xml version="1.0" encoding="UTF-8"?>
    <DeviceModeRequest
      xmlns="urn:schemas-upnp-org:av:dmor"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
      urn:schemas-upnp-org:av:dmor
      http://www.upnp.org/schemas/av/dmor.xsd">
      <actionBurstRequest>
         <totalTime>80000</totalTime>
         <responseTime>2000</responseTime>
         <label>SyncMusic1<label>
         <description>Keith wants to sync his music to your
         Cell</description>
      </actionBurstRequest>
   </DeviceModeRequest>"
)

Response:
ExtendDeviceMode("
   <?xml version="1.0" encoding="UTF-8"?>
   <DeviceModeStatus
      xmlns="urn:schemas-upnp-org:av:dmos"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
      urn:schemas-upnp-org:av:dmos
      http://www.upnp.org/schemas/av/dmos.xsd">
      <actionBurstStatus>
         <totalTime>120000</totalTime>
         <responseTime>2000</responseTime>
      </actionBurstStatus>
   </DeviceModeStatus>"
)
```

The initial values are returned in the *ExtendDeviceMode()* output arguments. As a result of the request the ContentDirectory service grants an additional two minutes of *ActionBurst mode*.

The *DeviceModeStatus* state variable is immediately updated as shown below and the ContentDirectory service begins counting down the *DeviceModeStatus* state variable *totalTime* value again.

```
Request:
GetDeviceModeStatus()

Response:
GetDeviceModeStatus("
<?xml version="1.0" encoding="UTF-8"?>
<DeviceModeStatus
   xmlns="urn:schemas-upnp-org:av:dmos"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
   urn:schemas-upnp-org:av:dmos
      http://www.upnp.org/schemas/av/dmos.xsd">
   <actionBurstStatus>
      <totalTime>80000</totalTime>
      <responseTime>2000</responeTime>
   </actionBurstStatus>
</DeviceModeStatus>"
)
```

Before the additional 80 seconds is up the control point finishes its tasks and invokes the CancelDeviceMode() action to return the ContentDirectory service implementation to a normal mode. It is considered preferred behavior to cancel the *ActionBurst mode* when it is no longer needed by the initiating control point. The input parameter for the CancelDeviceMode() action is as follows:

```
Request:
CancelDeviceMode("CPkeithMobile-QRX4-DEC-03-2009-10:31:04:06")

Response:
CancelDeviceMode()
```

As a result the DeviceMode and DeviceModeStatus state variables are immediately updated and returned to a normal mode as shown below. The DeviceMode state variable is also evented.

**Event**:
DeviceMode state variable contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceMode
 xmlns="urn:schemas-upnp-org:av:dmo"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:dmo
      http://www.upnp.org/schemas/av/dmo.xsd">
   <mode></mode>
</DeviceMode>

Request:
GetDeviceModeStatus()

Response:
GetDeviceModeStatus("
<?xml version="1.0" encoding="UTF-8"?>
<DeviceModeStatus
   xmlns="urn:schemas-upnp-org:av:dmos"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
   urn:schemas-upnp-org:av:dmos
      http://www.upnp.org/schemas/av/dmos.xsd">
</DeviceModeStatus>
)
```

### 2.6.20.2 Initiating and Managing *ExclusiveOwnership* mode

In this example the control point wants to request *ExclusiveOwnership* of the device for the next 10 minutes in order to complete some updates on the content directory of the device without any interruptions from other control points.

**Request**:

```
RequestDeviceMode(
   "CP-AnnesCamera-000001",
   "<?xml version="1.0" encoding="UTF-8"?>
    <DeviceModeRequest
      xmlns="urn:schemas-upnp-org:av:dmor"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:schemas-upnp-org:av:dmor
         http://www.upnp.org/schemas/av/dmor.xsd">
      <exclusiveOwnershipRequest>
         <resourceID type="Device"></resourceID>
         <totalTime>600000</totalTime>
         <label>CDS Upload<label>
         <description>Upload my pictures from Texas trip</description>
      </exclusiveOwnershipRequest>
   </DeviceModeRequest>"
)
```

**Response:**

```
RequestDeviceMode(
   "DMODE-id-AACD-XTRK-8FF3-0EFE-5DC7",
   "<?xml version="1.0" encoding="UTF-8"?>
    <DeviceModeStatus
      xmlns="urn:schemas-upnp-org:av:dmos"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:schemas-upnp-org:av:dmos
         http://www.upnp.org/schemas/av/dmos.xsd">
      <exclusiveOwnershipStatus>
         <resourceID type="Device"></resourceID>
         <totalTime>600000</totalTime>
      </exclusiveOwnershipStatus>
   </DeviceModeStatus>"
)
```

The initial values are returned in the *RequestDeviceMode()* output arguments along with a ContentDirectory service assigned ID valid for this *ActionBurst*.

As a result of the request, the device grants the control point *ExclusiveOwnership mode* for 10 minutes.

The *DeviceMode* state variable is immediately modified and evented and the *DeviceModeStatus* state variable is immediately updated as shown below and begins counting down the *DeviceModeStatus* state variable *totalTime* value.

**Event**:
*DeviceMode* state variable contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceMode
   xmlns="urn:schemas-upnp-org:av:dmo"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:av:dmo
      http://www.upnp.org/schemas/av/dmo.xsd">
   <mode type="ExclusiveOwnership" CPRequested="1"></mode>
</DeviceMode>
```

**Request**:

```
GetDeviceModeStatus()
```

**Response**:

```
GetDeviceModeStatus("
<?xml version="1.0" encoding="UTF-8"?>
<DeviceModeStatus
   xmlns="urn:schemas-upnp-org:av:dmos"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:av:dmos
      http://www.upnp.org/schemas/av/dmos.xsd">
   <exclusiveOwnershipStatus>
      <resourceID type="Device"></resourceID>
      <totalTime>600000</totalTime>
   </exclusiveOwnershipStatus>
</DeviceModeStatus>"
)
```

Before the *ExclusiveOwnership mode* granted to the control point expires, suppose in this case with 10 seconds remaining, the control point realizes that it will likely not complete its task and requests an extension of the *ExclusiveOwnership* mode by invoking the *ExtendDeviceMode()* action.

**Request**:

```
ExtendDeviceMode(
   "DMODE-id-AACD-XTRK-8FF3-0EFE-5DC7",
   "<DeviceModeRequest
      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">
      <exclusiveOwnershipRequest>
         <resourceID type="Device"></resourceID>
         <totalTime>180000</totalTime>
         <label>Finish CDS Upload<label>
         <description>
            Continue to Upload my pictures from trip to Texas
         </description>
      </exclusiveOwnershipRequest>
   </DeviceModeRequest>"
)
```

**Response**:

```
ExtendDeviceMode(
   "<?xml version="1.0" encoding="UTF-8"?>
```

```
    <DeviceModeStatus
      xmlns="urn:schemas-upnp-org:av:dmos"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:schemas-upnp-org:av:dmos
         http://www.upnp.org/schemas/av/dmos.xsd">
      <exclusiveOwnershipStatus>
         <resourceID type="Device"></resourceID>
         <totalTime>180000</totalTime>
      </exclusiveOwnershipStatus>
   </DeviceModeStatus>"
)
```

As a result the device grants the control point an additional 3 minutes (180000 milliseconds). Also, the *DeviceModeStatus* state variable is immediately updated as shown below.

**Request**:
```
GetDeviceModeStatus()
```

**Response**:
```
GetDeviceModeStatus("
<?xml version="1.0" encoding="UTF-8"?>
<DeviceModeStatus
   xmlns="urn:schemas-upnp-org:av:dmos"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:av:dmos
      http://www.upnp.org/schemas/av/dmos.xsd">
   <exclusiveOwnershipStatus>
      <resourceID type="Device"></resourceID>
      <totalTime>180000</totalTime>
   </exclusiveOwnershipStatus>
</DeviceModeStatus>"
)
```

Before the additional 3 minutes are up the control point finishes its tasks and invokes the *CancelDeviceMode()* action to return the device to a normal mode.

**Request**:
```
CancelDeviceMode("DMODE-id-AACD-XTRK-8FF3-0EFE-5DC7")
```

**Response**:
```
CancelDeviceMode()
```

As a result the *DeviceMode* and *DeviceModeStatus* state variables are immediately updated and returned to a normal mode as shown below. The *DeviceMode* state variable is also evented.

**Event**:
*DeviceMode* state variable contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceMode
   xmlns="urn:schemas-upnp-org:av:dmo"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:av:dmo
      http://www.upnp.org/schemas/av/dmo.xsd">
   <mode></mode>
```

```
</DeviceMode>
```

**Request**:

```
GetDeviceModeStatus()
```

**Response**:

```
GetDeviceModeStatus("
<?xml version="1.0" encoding="UTF-8"?>
<DeviceModeStatus
   xmlns="urn:schemas-upnp-org:av:dmos"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:av:dmos
      http://www.upnp.org/schemas/av/dmos.xsd">
</DeviceModeStatus>"
)
```

## 2.6.21 Synchronized Playback

### 2.6.21.1 Precision Time Synchronized Playback for RTSP-RTP Transport

For RTSP-RTP transport the control point must determine that all MediaRenderer devices and as well as the MediaServer device share a common precision timebase.

#### 2.6.21.1.1 MediaServer Control Point Operations

The control point can determine overall MediaServer device support for time synchronized media by locating the *CLOCKSYNC feature* in the ContentDirectory service features obtained via the *GetFeatureList()* action. Absence of the *CLOCKSYNC feature* indicates that the MediaServer does not support precision time synchronization. Note, synchronized playback can still be possible using other transports such as HTTP which do not require MediaServer precision timebase support. See Section 2.6.21.2, “Precision Time Synchronized Playback for HTTP Transport” for further details.

The control point also obtains detailed information for supported MediaServer timebase(s) by using the ConnectionManager service [CM] *GetFeatureList()* action on the MediaServer device. (It is possible to determine the presence of the *CLOCKSYNC feature* in the ContentDirectory service by using the ConnectionManager service *GetFeatureList()* action.) The *CLOCKSYNC feature* `<Feature>` element returned by this action result contains a series of `<deviceClockInfo>` elements providing clock synchronization metadata. A simple comparison of device clock information between a MediaServer and MediaRenderer indicates the possibility of synchronized playback between the devices. For detailed information about comparing `<deviceClockInfo>` information between a MediaServer and MediaRenderer, see ConnectionManager service [CM].

Enabling synchronized rtsp-rtp playback between the MediaServer and multiple MediaRenderer devices additionally requires that a MediaServer’s content’s *upnp:resExt::clockSync* metadata references device clock information that matches against each MediaRenderer.

The control point can locate clock synchronization metadata related to a selected ContentDirectory service content binary by matching the content binary’s *upnp:resExt::clockSync@deviceClockInfoID* and *upnp:resExt::clockSync@supportedTimestampsID* property values with corresponding `id=` attributes in the `<deviceClockInfo>` and `<supportedTimestamps>` elements in the ConnectionManager service [CM] *GetFeatureList()* action result obtained above.

**Control point determines if ContentDirectory service supports precision time synchronization:**

**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="CLOCKSYNC" version="1" />
</Features>"
}
```

**Control point obtains content binary metadata:**

```
Browse("50", "BrowseMetadata", "*", 0, 1, "")
```

```
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="50" parentID="5" restricted="1">
      <dc:title>CNN News</dc:title>
      <upnp:class>
         object.item.videoItem
      </upnp:class>

     <res id="50-rtp" protocolInfo="rtsp-rtp-udp:*:video/mpeg:*">
           rtp://10.0.0.1/contentdir?id=50-rtp
     </res>
     <upnp:resExt id="50-rtp">
        <upnp:clockSync deviceClockInfoID="MS-CLOCK#1-802.1AS"
                        supportedTimestampsID="C1-RTP_ALL-IEEE1733" />
     </upnp:resExt>

     <res id="50-http" protocolInfo="http-get:*:video/mpeg:*">
           http://10.0.0.1/contentdir?id=50-http
     </res>

   </item>
</DIDL-Lite>", 1, 1, 50)
```

**Control point obtains supported clock synchronication metadata from MediaServer ConnectionManager service:**

```
Request:
ConnectionManager::GetFeatureList()
```

**Response**:

```
ConnectionManager::GetFeatureList("
<?xml version="1.0" encoding="UTF-8"?>
<Features
 xmlns="urn:schemas-upnp-org:av:cm-featureList"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:cm-featureList
    http://www.upnp.org/schemas/av/cm-featureList.xsd">
   <Feature name="CLOCKSYNC" version="1">
       <deviceClockInfo id="MS-CLOCK#1-802.1AS" updateID="1">
         <syncProtocolID>802.1AS</syncProtocolID>
         <masterClockID>123456FFFE789ABC</masterClockID>
         <accuracy>10</accuracy>
         <supportedTimestamps id="C1-RTP_ALL-IEEE-1733"
                              protocol="rtsp-rtp-udp"
                              format="*">
           RTP+IEEE-1733
         </supportedTimestamps>
      </deviceClockInfo>
   </Feature>
</Features>")
```

### 2.6.21.1.2 MediaRenderer Control Point Operations

The control point then issues a ConnectionManager service [CM] *GetFeatureList()* action on each MediaRenderer device. The *CLOCKSYNC feature* `<feature>` element returned by this action result contains a series of `<deviceClockInfo>` elements providing clock synchronization metadata. The control point then locates the relevant MediaRenderer clock synchronization metadata by searching for a `<deviceClockInfo>` element with `<syncProtocolID>` and `<supportedTimestamp>` element which matches the corresponding clock synchronization elements of the selected content binary

If the control point successfully matches the clock synchronization elements obtained from the MediaServer and MediaRenderer then synchronized playback is possible.

**Note:** No futher processing of clock synchronization metadata is necessary since the MediaRenderer will automatically select the appropriate timebase based on the media format of the content binary. See AVTransport service [AVT] Theory of Operations for more details.

**Obtain supported clock synchronication metadata from MediaRenderer ConnectionManager service:**

**Request**:

```
ConnectionManager::GetFeatureList()
```

**Response**:

```
ConnectionManager::GetFeatureList("
<?xml version="1.0" encoding="UTF-8"?>
<Features
 xmlns="urn:schemas-upnp-org:av:cm-featureList"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:cm-featureList
    http://www.upnp.org/schemas/av/cm-featureList.xsd">
   <Feature name="CLOCKSYNC" version="1">
       <deviceClockInfo id="MR#1-802.1AS" updateID="1">
         <syncProtocolID>802.1AS</syncProtocolID>
         <masterClockID>123456FFFE789ABC</masterClockID>
         <accuracy>50</accuracy>
         <supportedTimestamps id="MR-1733-Video"
                              protocol="rtsp-rtp-udp"
                              format="video/mpeg">
           RTP+IEEE-1733
         </supportedTimestamps>
      </deviceClockInfo>
      <deviceClockInfo id="MR#2-802.1AS" updateID="1">
         <syncProtocolID>802.1AS</syncProtocolID>
         <masterClockID>123456FFFE789ABC</masterClockID>
         <accuracy>50</accuracy>
         <supportedTimestamps id="MR-IDENTITY-Video"
                              protocol="http-get"
                              format="video/mpeg">
           Identity
         </supportedTimestamps>
       </deviceClockInfo>
   </Feature>
</Features>")
```

**Note:** This procedure needs to be done for each MediaRenderer device participating in the synchronized playback operation.

### 2.6.21.2 Precision Time Synchronized Playback for HTTP Transport

For HTTP transport media the control point must determine that all MediaRenderer devices share a common precision timebase which supports the content binary format.

#### 2.6.21.2.1 MediaServer Control Point Operations

**Control point obtains content binary metadata:**

```
Browse("50", "BrowseMetadata", "*", 0, 1, "")
```

**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="50" parentID="5" restricted="1">
      <dc:title>CNN News</dc:title>
      <upnp:class>
         object.item.videoItem
      </upnp:class>

       <res id="50-rtp" protocolInfo="rtsp-rtp-udp:*:video/mpeg:*">
             rtp://10.0.0.1/contentdir?id=50-rtp
       </res>
       <upnp:resExt id="50-rtp">
```

```
          <upnp:clockSync deviceClockInfoID="MS-CLOCK#1-802.1AS"
                          supportedTimestampsID="C1-RTP_ALL-IEEE1733" />
       </upnp:resExt>

       <res id="50-http" protocolInfo="http-get:*:video/mpeg:*">
             http://10.0.0.1/contentdir?id=50-http
       </res>

   </item>
</DIDL-Lite>", 1, 1, 50)
```

#### 2.6.21.2.2 MediaRenderer Control Point Operations

The control point then issues a ConnectionManager service [CM] *GetFeatureList()* action on each MediaRenderer device. The *CLOCKSYNC feature* `<feature>` element returned by this action result contains a series of `<deviceClockInfo>` elements providing clock synchronization metadata.

The control point determines that all MediaRenderer devices to be synchronized share a common synchronization protocol is indicated by the `<syncProtocolID>` element and that a <supportedTimestamps> element is found which matches the transport and format of the content binary.

**Note:** No futher processing of clock synchronization metadata is necessary since the MediaRenderer will automatically select the appropriate timebase based on the media format of the content binary. See AVTransport Theory of Operations for more details.

**Obtain supported clock synchronication metadata from MediaRenderer ConnectionManager service:**

**Request**:

```
ConnectionManager::GetFeatureList()
```

**Response**:

```
 ConnectionManager::GetFeatureList("
<?xml version="1.0" encoding="UTF-8"?>
<Features
 xmlns="urn:schemas-upnp-org:av:cm-featureList"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:av:avs
    http://www.upnp.org/schemas/av/cm-featureList.xsd">
   <Feature name="CLOCKSYNC" version="1">
       <deviceClockInfo id="MR#1-802.1AS" updateID="1">
         <syncProtocolID>802.1AS</syncProtocolID>
         <masterClockID>123456FFFE789ABC</masterClockID>
         <accuracy>50</accuracy>
         <supportedTimestamps id="MR-1733-Video"
                              protocol="rtsp-rtp-udp"
                              format="video/mpeg">
           RTP+IEEE-1733
         </supportedTimestamps>
       </deviceClockInfo>

       <deviceClockInfo id="MR#2-802.1AS" updateID="1">
         <syncProtocolID>802.1AS</syncProtocolID>
         <masterClockID>123456FFFE789ABC</masterClockID>
         <accuracy>50</accuracy>
         <supportedTimestamps id="MR-IDENTITY-Video"
```

```
                              protocol="http-get"
                              format="video/mpeg">
           Identity
         </supportedTimestamps>
       </deviceClockInfo>
   </Feature>
</Features>")
```

# 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>
<action>
   <name>RequestDeviceMode</name>
   <argumentList>
      <argument>
         <name>CPID</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_CPID
         </relatedStateVariable>
      </argument>
      <argument>
         <name>DeviceModeRequest</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_DeviceModeRequest
         </relatedStateVariable>
      </argument>
      <argument>
         <name>DeviceModeID</name>
         <direction>out</direction>
         <relatedStateVariable>
            A_ARG_TYPE_DeviceModeID
         </relatedStateVariable>
      </argument>
      <argument>
         <name>DeviceModeStatus</name>
         <direction>out</direction>
         <relatedStateVariable>
            DeviceModeStatus
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>
<action>
   <name>ExtendDeviceMode</name>
   <argumentList>
      <argument>
         <name>DeviceModeID</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_DeviceModeID
         </relatedStateVariable>
      </argument>
      <argument>
         <name>DeviceModeRequest</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_DeviceModeRequest
         </relatedStateVariable>
      </argument>
      <argument>
         <name>DeviceModeStatus</name>
         <direction>out</direction>
         <relatedStateVariable>
```

```
                  DeviceModeStatus
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>CancelDeviceMode</name>
         <argumentList>
            <argument>
               <name>DeviceModeID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_DeviceModeID
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetDeviceMode</name>
         <argumentList>
            <argument>
               <name>DeviceMode</name>
               <direction>out</direction>
               <relatedStateVariable>
                  DeviceMode
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetDeviceModeStatus</name>
         <argumentList>
            <argument>
               <name>DeviceModeStatus</name>
               <direction>out</direction>
               <relatedStateVariable>
                  DeviceModeStatus
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetPermissionsInfo</name>
         <argumentList>
            <argument>
               <name>PermissionsInfo</name>
               <direction>out</direction>
               <relatedStateVariable>
                  PermissionsInfo
               </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>
      <stateVariable sendEvents="yes">
         <name>DeviceMode</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_CPID</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_DeviceModeRequest</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_DeviceModeID</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>DeviceModeStatus</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>PermissionsInfo</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.

# Appendix A. Schemas

This appendix 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 Section 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 Section 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 Appendix E, “*CDS features*”.

# Appendix B. AV Working Committee Properties

The tables and sections 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 sections 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 Section 2.2.20.1, “Multi-valued property” and Section 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 property description sections following Table B-1, 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* | Appendix B.1.1 |
| 2. | *@parentID* | DIDL-Lite | xsd:string | *NO* | *R* | Appendix B.1.2 |
| 3. | *@refID* | DIDL-Lite | xsd:string | *NO* | *R* | Appendix B.1.3 |
| 4. | *@restricted* | DIDL-Lite | xsd:boolean | *NO* | *R* | Appendix B.1.4 |
| 5. | *@searchable* | DIDL-Lite | xsd:boolean | *NO* | *R* | Appendix B.1.5 |
| 6. | *@childCount* | DIDL-Lite | xsd:unsignedInt | *NO* | *R* | Appendix B.1.6 |
| 7. | *dc:title* | dc | xsd:string | *NO* | *V* | Appendix B.1.7 |
| 8. | *dc:creator* | dc | xsd:string | *NO* | *V* | Appendix B.1.8 |
| 9. | *res* | DIDL-Lite | xsd:anyURI | *YES* | *V* | Appendix B.1.9 |
| 10. | *res@id* | DIDL-Lite | xsd:string | *NO* | *V* | Appendix B.1.9.1 |
| 11. | *upnp:class* | upnp | xsd:string | *NO* | *V* | Appendix B.1.10 |
| 12. | *upnp:class@name* | upnp | xsd:string | *NO* | *V* | Appendix B.1.10.1 |
| 13. | *upnp:searchClass* | upnp | xsd:string | *YES* | *R* | Appendix B.1.11 |
| 14. | *upnp:searchClass@name* | upnp | xsd:string | *NO* | *R* | Appendix B.1.11.1 |
| 15. | *upnp:searchClass@includeDerived* | upnp | xsd:boolean | *NO* | *R* | Appendix B.1.11.2 |
| 16. | *upnp:createClass* | upnp | xsd:string | *YES* | *R* | Appendix B.1.12 |
| 17. | *upnp:createClass@name* | upnp | xsd:string | *NO* | *R* | Appendix B.1.12.1 |
| 18. | *upnp:createClass@includeDerived* | upnp | xsd:boolean | *NO* | *R* | Appendix B.1.12.2 |
| 19. | *upnp:writeStatus* | upnp | xsd:string | *NO* | *R* | Appendix B.1.13 |

| # | Property Name | NS | Data Type | M-Val | R/W | Reference |
|---|---|---|---|---|---|---|
| 20. | res@protocolInfo | DIDL-Lite | xsd:string | NO | V | Appendix B.2.1.1 |
| 21. | res@importUri | DIDL-Lite | xsd:anyURI | NO | R | Appendix B.2.1.2 |
| 22. | res@size | DIDL-Lite | xsd:unsignedLong | NO | V | Appendix B.2.1.3 |
| 23. | res@duration | DIDL-Lite | xsd:string | NO | V | Appendix B.2.1.4 |
| 24. | res@protection | DIDL-Lite | xsd:string | NO | V | Appendix B.2.1.5 |
| 25. | res@bitrate | DIDL-Lite | xsd:unsignedInt | NO | V | Appendix B.2.1.6 |
| 26. | res@bitsPerSample | DIDL-Lite | xsd:unsignedInt | NO | V | Appendix B.2.1.7 |
| 27. | res@sampleFrequency | DIDL-Lite | xsd:unsignedInt | NO | V | Appendix B.2.1.8 |
| 28. | res@nrAudioChannels | DIDL-Lite | xsd:unsignedInt | NO | V | Appendix B.2.1.9 |
| 29. | res@resolution | DIDL-Lite | xsd:string | NO | V | Appendix B.2.1.10 |
| 30. | res@colorDepth | DIDL-Lite | xsd:unsignedInt | NO | V | Appendix B.2.1.11 |
| 31. | res@tspec | DIDL-Lite | xsd:string | NO | V | Appendix B.2.1.12 |
| 32. | res@allowedUse | DIDL-Lite | CSV (xsd:string) | NO | V | Appendix B.2.1.13 |
| 33. | res@validityStart | DIDL-Lite | xsd:string | NO | V | Appendix B.2.1.14 |
| 34. | res@validityEnd | DIDL-Lite | xsd:string | NO | V | Appendix B.2.1.15 |
| 35. | res@remainingTime | DIDL-Lite | xsd:string | NO | V | Appendix B.2.1.16 |
| 36. | res@usageInfo | DIDL-Lite | xsd:string | NO | V | Appendix B.2.1.17 |
| 37. | res@rightsInfoURI | DIDL-Lite | xsd:anyURI | NO | V | Appendix B.2.1.18 |
| 38. | res@contentInfoURI | DIDL-Lite | xsd:anyURI | NO | V | Appendix B.2.1.19 |
| 39. | res@recordQuality | DIDL-Lite | CSV (xsd:string) | NO | V | Appendix B.2.1.20 |
| 40. | res@daylightSaving | DIDL-Lite | xsd:string | NO | V | Appendix B.2.1.21 |
| 41. | upnp:resExt | upnp | <XML> | YES | V | Appendix B.3.1 |
| 42. | upnp:resExt@id | upnp | xsd:string | NO | V | Appendix B.3.1.1 |
| 43. | upnp:artist | upnp | xsd:string | YES | V | Appendix B.4.1 |
| 44. | upnp:artist@role | upnp | xsd:string | NO | V | Appendix B.4.1.1 |
| 45. | upnp:actor | upnp | xsd:string | YES | V | Appendix B.4.2 |
| 46. | upnp:actor@role | upnp | xsd:string | NO | V | Appendix B.4.2.1 |
| 47. | upnp:author | upnp | xsd:string | YES | V | Appendix B.4.3 |
| 48. | upnp:author@role | upnp | xsd:string | NO | V | Appendix B.4.3.1 |
| 49. | upnp:producer | upnp | xsd:string | YES | V | Appendix B.4.4 |
| 50. | upnp:director | upnp | xsd:string | YES | V | Appendix B.4.5 |
| 51. | dc:publisher | dc | xsd:string | YES | V | Appendix B.4.6 |
| 52. | dc:contributor | dc | xsd:string | YES | V | Appendix B.4.7 |
| 53. | upnp:genre | upnp | xsd:string | YES | V | Appendix B.5.1 |
| 54. | upnp:genre@id | upnp | xsd:string | NO | V | Appendix B.5.1.1 |
| 55. | upnp:genre@extended | upnp | CSV (xsd:string) | NO | V | Appendix B.5.1.2 |
| 56. | upnp:album | upnp | xsd:string | YES | V | Appendix B.5.2 |
| 57. | upnp:playlist | upnp | xsd:string | YES | V | Appendix B.5.3 |
| 58. | upnp:albumArtURI | upnp | xsd:anyURI | YES | V | Appendix B.6.1 |

| # | Property Name | NS | Data Type | M-Val | R/W | Reference |
|---|---|---|---|---|---|---|
| 59. | upnp:artistDiscographyURI | upnp | xsd:anyURI | NO | V | Appendix B.6.2 |
| 60. | upnp:lyricsURI | upnp | xsd:anyURI | NO | V | Appendix B.6.3 |
| 61. | dc:relation | dc | xsd:string | YES | V | Appendix B.6.4 |
| 62. | upnp:storageTotal | upnp | xsd:long | NO | R | Appendix B.7.1 |
| 63. | upnp:storageUsed | upnp | xsd:long | NO | R | Appendix B.7.2 |
| 64. | upnp:storageFree | upnp | xsd:long | NO | R | Appendix B.7.3 |
| 65. | upnp:storageMaxPartition | upnp | xsd:long | NO | R | Appendix B.7.4 |
| 66. | upnp:storageMedium | upnp | xsd:string | NO | R | Appendix B.7.5 |
| 67. | dc:description | dc | xsd:string | NO | V | Appendix B.8.1 |
| 68. | upnp:longDescription | upnp | xsd:string | NO | V | Appendix B.8.2 |
| 69. | upnp:icon | upnp | xsd:anyURI | NO | V | Appendix B.8.3 |
| 70. | upnp:region | upnp | xsd:string | NO | V | Appendix B.8.4 |
| 71. | dc:rights | dc | xsd:string | YES | V | Appendix B.8.5 |
| 72. | dc:date | dc | xsd:string | NO | V | Appendix B.8.6 |
| 73. | dc:date@upnp:daylightSaving | upnp | xsd:string | NO | V | Appendix B.8.6.1 |
| 74. | dc:language | dc | xsd:string | YES | V | Appendix B.8.7 |
| 75. | upnp:playbackCount | upnp | xsd:int | NO | R | Appendix B.8.8 |
| 76. | upnp:lastPlaybackTime | upnp | xsd:string | NO | V | Appendix B.8.9 |
| 77. | upnp:lastPlaybackTime @daylightSaving | upnp | xsd:string | NO | V | Appendix B.8.9.1 |
| 78. | upnp:lastPlaybackPosition | upnp | xsd:string | NO | V | Appendix B.8.10 |
| 79. | upnp:recordedStartDateTime | upnp | xsd:string | NO | V | Appendix B.8.11 |
| 80. | upnp:recordedStartDateTime @daylightSaving | upnp | xsd:string | NO | V | Appendix B.8.11.1 |
| 81. | upnp:recordedDuration | upnp | xsd:string | NO | V | Appendix B.8.12 |
| 82. | upnp:recordedDayOfWeek | upnp | xsd:string | NO | V | Appendix B.8.13 |
| 83. | upnp:srsRecordScheduleID | upnp | xsd:string | NO | R | Appendix B.8.14 |
| 84. | upnp:srsRecordTaskID | upnp | xsd:string | NO | R | Appendix B.8.15 |
| 85. | upnp:recordable | upnp | xsd:boolean | NO | V | Appendix B.8.16 |
| 86. | upnp:programTitle | upnp | xsd:string | NO | V | Appendix B.9.1 |
| 87. | upnp:seriesTitle | upnp | xsd:string | NO | V | Appendix B.9.2 |
| 88. | upnp:programID | upnp | xsd:string | NO | V | Appendix B.9.3 |
| 89. | upnp:programID@type | upnp | xsd:string | NO | V | Appendix B.9.3.1 |
| 90. | upnp:seriesID | upnp | xsd:string | NO | V | Appendix B.9.4 |
| 91. | upnp:seriesID@type | upnp | xsd:string | NO | V | Appendix B.9.4.1 |
| 92. | upnp:channelID | upnp | xsd:string | YES | V | Appendix B.9.5 |
| 93. | upnp:channelID@type | upnp | xsd:string | NO | V | Appendix B.9.5.1 |
| 94. | upnp:channelID@distriNetworkName | upnp | xsd:string | NO | V | Appendix B.9.5.2 |
| 95. | upnp:channelID@distriNetworkID | upnp | xsd:string | NO | V | Appendix B.9.5.3 |
| 96. | upnp:episodeCount | upnp | xsd:unsignedInt | NO | V | Appendix B.9.5.2 |

| # | Property Name | NS | Data Type | M-Val | R/W | Reference |
|---|---|---|---|---|---|---|
| 97. | upnp:episodeNumber | upnp | xsd:unsignedInt | NO | V | Appendix B.9.7 |
| 98. | upnp:programCode | upnp | xsd:string | NO | V | Appendix B.9.8 |
| 99. | upnp:programCode@type | upnp | xsd:string | NO | V | Appendix B.9.8.1 |
| 100. | upnp:rating | upnp | xsd:string | NO | V | Appendix B.9.9 |
| 101. | upnp:rating@type | upnp | xsd:string | NO | V | Appendix B.9.9.1 |
| 102. | upnp:episodeType | upnp | xsd:string | NO | V | Appendix B.9.10 |
| 103. | upnp:channelGroupName | upnp | xsd:string | NO | V | Appendix B.10.1 |
| 104. | upnp:channelGroupName@id | upnp | xsd:string | NO | V | Appendix B.10.1.1 |
| 105. | upnp:callSign | upnp | xsd:string | NO | V | Appendix B.10.2 |
| 106. | upnp:networkAffiliation | upnp | xsd:string | NO | V | Appendix B.10.3 |
| 107. | upnp:serviceProvider | upnp | xsd:string | NO | V | Appendix B.10.4 |
| 108. | upnp:price | upnp | xsd:float | YES | R | Appendix B.10.5 |
| 109. | upnp:price@currency | upnp | xsd:string | NO | R | Appendix B.10.5.1 |
| 110. | upnp:payPerView | upnp | xsd:boolean | NO | R | Appendix B.10.6 |
| 111. | upnp:epgProviderName | upnp | xsd:string | NO | V | Appendix B.10.7 |
| 112. | upnp:dateTimeRange | upnp | xsd:string | NO | R | Appendix B.10.8 |
| 113. | upnp:dateTimeRange@daylightSaving | upnp | xsd:string | NO | R | Appendix B.10.8.1 |
| 114. | upnp:programPreserved | upnp | xsd:string | NO | V | Appendix B.11.1 |
| 115. | upnp:programPreserved @startTime | upnp | xsd:string | NO | V | Appendix B.11.1.2 |
| 116. | upnp:programPreserved @startTimeDaylightSaving | upnp | xsd:string | NO | V | Appendix B.11.1.3 |
| 117. | upnp:programPreserved @endTime | upnp | xsd:string | NO | V | Appendix B.11.1.4 |
| 118. | upnp:programPreserved @endTimeDaylightSaving | upnp | xsd:string | NO | V | Appendix B.11.1.5 |
| 119. | upnp:preservedTimeRange | upnp | xsd:string | NO | V | Appendix B.11.2 |
| 120. | upnp:preservedTimeRange @startTime | upnp | xsd:string | NO | V | Appendix B.11.2.1 |
| 121. | upnp:preservedTimeRange @startTimeDaylightSaving | upnp | xsd:string | NO | V | Appendix B.11.2.2 |
| 122. | upnp:preservedTimeRange @endTime | upnp | xsd:string | NO | V | Appendix B.11.2.3 |
| 123. | upnp:preservedTimeRange @endTimeDaylightSaving | upnp | xsd:string | NO | V | Appendix B.11.2.4 |
| 124. | upnp:programList | upnp | <XML> | NO | V | Appendix B.11.3 |
| 125. | upnp:programList::program | upnp | xsd:string | YES | V | Appendix B.11.3.1 |
| 126. | upnp:programList::program @preserved | upnp | xsd:boolean | NO | V | Appendix B.11.3.1.1 |
| 127. | upnp:radioCallSign | upnp | xsd:string | NO | V | Appendix B.12.1 |
| 128. | upnp:radioStationID | upnp | xsd:string | NO | V | Appendix B.12.2 |
| 129. | upnp:radioBand | upnp | xsd:string | NO | V | Appendix B.12.3 |
| 130. | upnp:channelNr | upnp | xsd:int | NO | V | Appendix B.13.1 |

| # | Property Name | NS | Data Type | M-Val | R/W | Reference |
|---|---|---|---|---|---|---|
| 131. | upnp:channelName | upnp | xsd:string | NO | V | Appendix B.13.2 |
| 132. | upnp:scheduledStartTime | upnp | xsd:string | NO | V | Appendix B.13.3 |
| 133. | upnp:scheduledStartTime @usage | upnp | xsd:string | NO | V | Appendix B.13.3.1 |
| 134. | upnp:scheduledStartTime @daylightSaving | upnp | xsd:string | NO | V | Appendix B.13.3.2 |
| 135. | upnp:scheduledEndTime | upnp | xsd:string | NO | V | Appendix B.13.4 |
| 136. | upnp:scheduledEndTime @daylightSaving | upnp | xsd:string | NO | V | Appendix B.13.4.1 |
| 137. | upnp:scheduledDuration | upnp | xsd:string | NO | V | Appendix B.13.5 |
| 138. | upnp:signalStrength | upnp | xsd:int | NO | R | Appendix B.14.1 |
| 139. | upnp:signalLocked | upnp | xsd:boolean | NO | R | Appendix B.14.2 |
| 140. | upnp:tuned | upnp | xsd:boolean | NO | R | Appendix B.14.3 |
| 141. | upnp:resExt::isSyncAnchor | upnp | xsd:string | NO | V | Appendix B.15.1 |
| 142. | upnp:resExt::componentInfo | upnp | xsd:string | YES | V | Appendix B.15.2 |
| 143. | upnp:resExt::componentInfo ::componentGroup | upnp | xsd:string | YES | V | Appendix B.15.2.1 |
| 144. | upnp:resExt::componentInfo ::componentGroup@groupID | upnp | <XML> | NO | V | Appendix B.15.2.1.1 |
| 145. | upnp:resExt::componentInfo ::componentGroup@required | upnp | xsd:anyURI | NO | V | Appendix B.15.2.1.2 |
| 146. | upnp:resExt::componentInfo ::componentGroup::component | upnp | xsd:string | YES | V | Appendix B.15.2.1.3 |
| 147. | upnp:resExt::componentInfo ::componentGroup ::component@componentID | upnp | xsd:anyURI | NO | V | Appendix B.15.2.1.3.1 |
| 148. | upnp:resExt::componentInfo ::componentGroup ::component@supportive | upnp | xsd:boolean | NO | V | Appendix B.15.2.1.3.2 |
| 149. | upnp:resExt::componentInfo ::componentGroup ::component@supportID | upnp | xsd:string | NO | V | Appendix B.15.2.1.3.3 |
| 150. | upnp:resExt::componentInfo ::componentGroup::component ::componentClass | upnp | xsd:string | NO | V | Appendix B.15.2.1.3.4 |
| 151. | upnp:resExt::componentInfo ::componentGroup::component ::contentType | upnp | xsd:string | NO | V | Appendix B.15.2.1.3.5 |
| 152. | upnp:resExt::componentInfo ::componentGroup ::component::contentType@MIMEType | upnp | xsd:string | NO | V | Appendix B.15.2.1.3.5.1 |
| 153. | upnp:resExt::componentInfo ::componentGroup ::component::contentType @extendedType | upnp | xsd:string | NO | V | Appendix B.15.2.1.3.5.2 |
| 154. | upnp:resExt::componentInfo ::componentGroup ::component::language | upnp | xsd:string | NO | V | Appendix B.15.2.1.3.6 |
| 155. | upnp:resExt::componentInfo ::componentGroup | upnp | <XML> | NO | V | Appendix B.15.2.1.3.7 |

| # | Property Name | NS | Data Type | M-Val | R/W | Reference |
|---|---|---|---|---|---|---|
| | ::component::compRes | | | | | |
| 156. | upnp:resExt::componentInfo ::componentGroup::component ::compRes::res | upnp | xsd:anyURI | NO | V | Appendix B.15.2.1.3.7.1 |
| 157. | upnp:resExt::componentInfo ::componentGroup::component ::compRes::res@protocolInfo | upnp | xsd:string | NO | V | Appendix B.15.2.1.3.7.1.1 |
| 158. | upnp:resExt::componentInfo ::componentGroup::component ::compRes::res@importUri | upnp | xsd:anyURI | NO | V | Appendix B.15.2.1.3.7.1.2 |
| 159. | upnp:resExt::componentInfo ::componentGroup::component ::compRes::isSyncAnchor | upnp | xsd:boolean | NO | V | Appendix B.15.2.1.3.7.2 |
| 160. | upnp:resExt::componentInfo ::componentGroup ::component::compRes::refUDN | upnp | xsd:string | NO | V | Appendix B.15.2.1.3.7.3 |
| 161. | upnp:resExt::componentInfo ::componentGroup::component ::compRes::refObjectID | upnp | xsd:string | NO | V | Appendix B.15.2.1.3.7.4 |
| 162. | upnp:resExt::componentInfo ::componentGroup::component ::compRes::refResID | upnp | xsd:string | NO | V | Appendix B.15.2.1.3.7.5 |
| 163. | upnp:segmentID | upnp | xsd:string | YES | R | Appendix B.16.1 |
| 164. | upnp:resExt::segmentInfo | upnp | xsd:string | NO | R/W | Appendix B.16.2 |
| 165. | upnp:resExt::segmentInfo @baseObjectID | upnp | xsd:string | NO | R/W | Appendix B.16.2.1 |
| 166. | upnp:resExt::segmentInfo@baseResID | upnp | xsd:string | NO | R/W | Appendix B.16.2.2 |
| 167. | upnp:resExt::segmentInfo ::timeRange | upnp | xsd:string | NO | R/W | Appendix B.16.2.3 |
| 168. | upnp:resExt::segmentInfo ::timeRange@start | upnp | xsd:string | NO | R/W | Appendix B.16.2.3.1 |
| 169. | upnp:resExt::segmentInfo ::timeRange@end | upnp | xsd:string | NO | R/W | Appendix B.16.2.3.2 |
| 170. | upnp:resExt::segmentInfo::byteRange | upnp | xsd:string | NO | R/W | Appendix B.16.2.4 |
| 171. | upnp:resExt::segmentInfo ::byteRange@start | upnp | xsd:unsignedLong | NO | R/W | Appendix B.16.2.4.1 |
| 172. | upnp:resExt::segmentInfo ::byteRange@end | upnp | xsd:unsignedLong | NO | R/W | Appendix B.16.2.4.2 |
| 173. | upnp:resExt::segmentInfo ::frameRange | upnp | xsd:string | NO | R/W | Appendix B.16.2.5 |
| 174. | upnp:resExt::segmentInfo ::frameRange@start | upnp | xsd:unsignedLong | NO | R/W | Appendix B.16.2.5.1 |
| 175. | upnp:resExt::segmentInfo ::frameRange@end | upnp | xsd:unsignedLong | NO | R/W | Appendix B.16.2.5.2 |
| 176. | @neverPlayable | DIDL-Lite | xsd:boolean | NO | V | Appendix B.17.1 |
| 177. | upnp:bookmarkID | upnp | xsd:string | YES | R/W | Appendix B.17.2 |
| 178. | upnp:bookmarkedObjectID | upnp | xsd:string | NO | R/W | Appendix B.17.3 |
| 179. | upnp:deviceUDN | upnp | xsd:string | NO | R/W | Appendix B.17.4 |
| 180. | upnp:deviceUDN@serviceType | upnp | xsd:string | NO | R/W | Appendix B.17.4.1 |

| # | Property Name | NS | Data Type | M-Val | R/W | Reference |
|---|---|---|---|---|---|---|
| 181. | *upnp:deviceUDN@serviceId* | upnp | xsd:string | *NO* | *R/W* | Appendix B.17.4.2 |
| 182. | *upnp:stateVariableCollection* | upnp | xsd:string | *YES* | *R/W* | Appendix B.17.5 |
| 183. | *upnp:stateVariableCollection@serviceName* | upnp | xsd:string | *NO* | *R/W* | Appendix B.17.5.1 |
| 184. | *upnp:stateVariableCollection@rcsInstanceType* | upnp | xsd:string | *NO* | *R/W* | Appendix B.17.5.2 |
| 185. | *upnp:DVDRegionCode* | upnp | xsd:int | *NO* | *V* | Appendix B.18.1 |
| 186. | *upnp:originalTrackNumber* | upnp | xsd:int | *NO* | *V* | Appendix B.18.2 |
| 187. | *upnp:toc* | upnp | xsd:string | *NO* | *V* | Appendix B.18.3 |
| 188. | *upnp:userAnnotation* | upnp | xsd:string | *YES* | *R/W* | Appendix B.18.4 |
| 189. | *desc* | DIDL-Lite | xsd:string | *YES* | *V* | Appendix B.18.5 |
| 190. | *desc@nameSpace* | DIDL-Lite | xsd:string | *NO* | *V* | Appendix B.18.5.1 |
| 191. | *upnp:containerUpdateID* | upnp | xsd:unsignedInt | *NO* | *R* | Appendix B.19.1 |
| 192. | *upnp:objectUpdateID* | upnp | xsd:unsignedInt | *NO* | *R* | Appendix B.19.2 |
| 193. | *upnp:totalDeletedChildCount* | upnp | xsd:unsignedInt | *NO* | *R* | Appendix B.19.3 |
| 194. | *res@updateCount* | DIDL-Lite | xsd:unsignedInt | *NO* | *R* | Appendix B.19.4 |
| 195. | *upnp:inclusionControl* | upnp | xsd:string | *NO* | *R/W* | Appendix B.20.1 |
| 196. | *upnp:inclusionControl::role* | upnp | xsd:string | *YES* | *R/W* | Appendix B.20.1.1 |
| 197. | *upnp:objectOwner* | upnp | xsd:string | *NO* | *R/W* | Appendix B.21.1 |
| 198. | *upnp:objectOwner@lock* | upnp | xsd:boolean | *NO* | *R/W* | Appendix B.21.1.1 |
| 199. | *upnp:objectOwner::role* | upnp | xsd:string | *YES* | *R/W* | Appendix B.21.1.2 |
| 200. | *upnp:objectLink* | upnp | <XML> | *YES* | *R/W* | Appendix B.22.1. |
| 201. | *upnp:objectLink@groupID* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.1. |
| 202. | *upnp:objectLink@headObjID* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.2 |
| 203. | *upnp:objectLink@nextObjID* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.3 |
| 204. | *upnp:objectLinl@prevObjID* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.4 |
| 205. | *upnp:objectLink::title* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.5 |
| 206. | *upnp:objectLink::startObject* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.6 |
| 207. | *upnp:objectLink::mode* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.7 |
| 208. | *upnp:objectLink::relatedInfo* | upnp | xsd:string | *YES* | *R/W* | Appendix B.22.1.8 |
| 209. | *upnp:objectLink::relatedInfo@role* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.8.1 |
| 210. | *upnp:objectLink::relatedInfo@roleText* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.8.2 |
| 211. | *upnp:objectLink::startInfo* | upnp | <XML> | *YES* | *R/W* | Appendix B.22.1.9 |
| 212. | *upnp:objectLink::startInfo@targetGroupID* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.9.1 |
| 213. | *upnp:objectLink::startInfo@targetObjID* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.9.2 |
| 214. | *upnp:objectLink::endAction* | upnp | <XML> | *NO* | *R/W* | Appendix B.22.1.10 |
| 215. | *upnp:objectLink::endAction@action* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.10.1 |
| 216. | *upnp:objectLink::endAction@targetGroupID* | upnp | xsd:string | *NO* | *R/W* | Appendix B.22.1.10.3 |

| # | Property Name | NS | Data Type | M-Val | R/W | Reference |
|---|---|---|---|---|---|---|
| 217. | upnp:objectLink::endAction @targetObjID | upnp | xsd:string | NO | R/W | Appendix B.22.1.10.2 |
| 218. | upnp:objectLinkRef | upnp | <XML> | YES | R/W | Appendix B.22.2 |
| 219. | upnp:objectLinkRef@groupID | upnp | xsd:string | NO | R/W | Appendix B.22.2.1 |
| 220. | upnp:objectLinkRef@targetGroupID | upnp | xsd:string | NO | R/W | Appendix B.22.2.2 |
| 221. | upnp:objectLinkRef@targetObjID | upnp | xsd:string | NO | R/W | Appendix B.22.2.3 |
| 222. | upnp:objectLinkRef@return | upnp | xsd:string | NO | R/W | Appendix B.22.2.4 |
| 223. | upnp:objectLinkRef::title | upnp | xsd:string | NO | R/W | Appendix B.22.2.5 |
| 224. | upnp:objectLinkRef::startObject | upnp | xsd:string | NO | R/W | Appendix B.22.2.6 |
| 225. | upnp:objectLinkRef::relatedInfo | upnp | xsd:string | YES | R/W | Appendix B.22.2.7 |
| 226. | upnp:objectLinkRef::relatedInfo@role | upnp | xsd:string | NO | R/W | Appendix B.22.2.7.1 |
| 227. | upnp:objectLinkRef::relatedInfo @roleText | upnp | xsd:string | NO | R/W | Appendix B.22.2.7.2 |
| 228. | upnp:foreignMetadata | upnp | <XML> | YES | V | Appendix B.23.1 |
| 229. | upnp:foreignMetadata@type | upnp | xsd:string | NO | V | Appendix B.23.1.1 |
| 230. | upnp:foreignMetadata::fmId | upnp | xsd:string | YES | V | Appendix B.23.1.2 |
| 231. | upnp:foreignMetadata::fmClass | upnp | xsd:string | YES | V | Appendix B.23.1.3 |
| 232. | upnp:foreignMetadata::fmProvider | upnp | xsd:string | NO | V | Appendix B.23.1.4 |
| 233. | upnp:foreignMetadata::fmBody | upnp | <XML> | NO | V | Appendix B.23.1.5 |
| 234. | upnp:foreignMetadata::fmBody @xmlFlag | upnp | xsd:boolean | NO | V | Appendix B.23.1.5.1 |
| 235. | upnp:foreignMetadata::fmBody @mimeType | upnp | xsd:string | NO | V | Appendix B.23.1.5.2 |
| 236. | upnp:foreignMetadata::fmBody ::fmEmbeddedXML | upnp | <XML> | NO | V | Appendix B.23.1.5.3 |
| 237. | upnp:foreignMetadata::fmBody ::fmEmbeddedString | upnp | xsd:string | NO | V | Appendix B.23.1.5.4 |
| 238. | upnp:foreignMetadata::fmBody ::fmURI | upnp | xsd:anyURI | NO | V | Appendix B.23.1.5.5 |
| 239. | upnp:resExt::clockSync | upnp | <XML> | NO | V | Appendix B.24.1 |
| 240. | upnp:resExt::clockSync @deviceClockInfoID | upnp | xsd:string | NO | V | Appendix B.24.1.1 |
| 241. | upnp:resExt::clockSync @supportedTimestampsID | upnp | xsd:string | NO | V | Appendix B.24.1.2 |
| 242. | upnp:resExt::DRMInfo | upnp | <XML> | YES | V | Appendix B.25.1 |
| 243. | upnp:resExt::DRMInfo ::foreignMetadata | upnp | <XML> | NO | V | Appendix B.23.1 |
| 244. | upnp:resExt::DRMInfo ::foreignMetadata@type | upnp | xsd:string | NO | V | Appendix B.23.1.1 |
| 245. | upnp:resExt::DRMInfo ::foreignMetadata::fmId | upnp | xsd:string | YES | V | Appendix B.23.1.2 |
| 246. | upnp:resExt::DRMInfo ::foreignMetadata::fmClass | upnp | xsd:string | YES | V | Appendix B.23.1.3 |
| 247. | upnp:resExt::DRMInfo ::foreignMetadata::fmProvider | upnp | xsd:string | NO | V | Appendix B.23.1.4 |

| # | Property Name | NS | Data Type | M-Val | R/W | Reference |
|---|---|---|---|---|---|---|
| 248. | upnp:resExt::DRMInfo ::foreignMetadata::fmBody | upnp | <XML> | NO | V | Appendix B.23.1.5 |
| 249. | upnp:resExt::DRMInfo ::foreignMetadata::fmBody@xmlFlag | upnp | xsd:boolean | NO | V | Appendix B.23.1.5.1 |
| 250. | upnp:resExt::DRMInfo ::foreignMetadata::fmBody @mimeType | upnp | xsd:string | NO | V | Appendix B.23.1.5.2 |
| 251. | upnp:resExt::DRMInfo ::foreignMetadata::fmBody ::fmEmbeddedXML | upnp | <XML> | NO | V | Appendix B.23.1.5.4 |
| 252. | upnp:resExt::DRMInfo ::foreignMetadata::fmBody ::fmEmbeddedString | upnp | xsd:string | NO | V | Appendix B.23.1.5.3 |
| 253. | upnp:resExt::DRMInfo ::foreignMetadata::fmBody::fmURI | upnp | xsd:anyURI | NO | V | Appendix B.23.1.5.5 |

## B.1 Base Properties

### Table B-2: Base Properties Overview

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| @id | DIDL-Lite | xsd:string | NO | Appendix B.1.1 |
| @parentID | DIDL-Lite | xsd:string | NO | Appendix B.1.2 |
| @refID | DIDL-Lite | xsd:string | NO | Appendix B.1.3 |
| @restricted | DIDL-Lite | xsd:boolean | NO | Appendix B.1.4 |
| @searchable | DIDL-Lite | xsd:boolean | NO | Appendix B.1.5 |
| @childCount | DIDL-Lite | xsd:unsignedInt | NO | Appendix B.1.6 |
| dc:title | dc | xsd:string | NO | Appendix B.1.7 |
| dc:creator | dc | xsd:string | NO | Appendix B.1.8 |
| res | DIDL-Lite | xsd:anyURI | YES | Appendix B.1.9 |
| res@id | DIDL-Lite | xsd:string | NO | Appendix B.1.9.1 |
| upnp:class | upnp | xsd:string | NO | Appendix B.1.10 |
| upnp:class@name | upnp | xsd:string | NO | Appendix B.1.10.1 |
| upnp:searchClass | upnp | xsd:string | YES | Appendix B.1.11 |
| upnp:searchClass@name | upnp | xsd:string | NO | Appendix B.1.11.1 |
| upnp:searchClass@includeDerived | upnp | xsd:boolean | NO | Appendix B.1.11.2 |
| upnp:createClass | upnp | xsd:string | YES | Appendix B.1.12 |
| upnp:createClass@name | upnp | xsd:string | NO | Appendix B.1.12.1 |
| upnp:createClass@includeDerived | upnp | xsd:boolean | NO | Appendix B.1.12.2 |
| upnp:writeStatus | upnp | xsd:string | NO | Appendix 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 Section 2.3.7, "ServiceResetToken" and Section 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 Section 2.3.7, "ServiceResetToken" and Section 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 Section 2.3.7, "ServiceResetToken" and Section 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.1.9.1 *res@id*

**Namespace**: DIDL-Lite **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This property uniquely identifies this instance of the object's *res* property. Its format and value are vendor-defined, but at all times (within a given object), all instances of the *res@id* property

MUST contain a unique value. If the *upnp:resExt* property is present then the *res@id* property is REQUIRED.

**Default Value**: N/A.

### 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 Appendix C.2.1 |
| "*object.item.imageItem*" | *O* | See Appendix C.2.1.1 |
| "*object.item.imageItem.photo*" | *O* | See Appendix C.2.1.1.1 |
| "*object.item.audioItem*" | *O* | See Appendix C.2.1.2 |
| "*object.item.audioItem.musicTrack*" | *O* | See Appendix C.2.1.2.1 |
| "*object.item.audioItem.audioBroadcast*" | *O* | See Appendix C.2.1.2.2 |
| "*object.item.audioItem.audioBook*" | *O* | See Appendix C.2.1.2.3 |
| "*object.item.videoItem*" | *O* | See Appendix C.2.1.3 |
| "*object.item.videoItem.movie*" | *O* | See Appendix C.2.1.3.1 |
| "*object.item.videoItem.videoBroadcast*" | *O* | See Appendix C.2.1.3.2 |
| "*object.item.videoItem.musicVideoClip*" | *O* | See Appendix C.2.1.3.3 |
| "*object.item.playlistItem*" | *O* | See Appendix C.2.1.4 |
| "*object.item.textItem*" | *O* | See Appendix C.2.1.5 |
| "*object.item.bookmarkItem*" | *O* | See Appendix C.2.1.6 |
| "*object.item.epgItem*" | *O* | See Appendix C.2.1.7 |
| "*object.item.epgItem.audioProgram*" | *O* | See Appendix C.2.1.7.1 |
| "*object.item.epgItem.videoProgram*" | *O* | See Appendix C.2.1.7.2 |
| "*object.container.person*" | *O* | See Appendix C.2.2.1 |
| "*object.container.person.musicArtist*" | *O* | See Appendix C.2.2.1.1 |
| "*object.container.playlistContainer*" | *O* | See Appendix C.2.2.2 |
| "*object.container.album*" | *O* | See Appendix C.2.2.3 |
| "*object.container.album.musicAlbum*" | *O* | See Appendix C.2.2.3.1 |
| "*object.container.album.photoAlbum*" | *O* | See Appendix C.2.2.3.2 |
| "*object.container.genre*" | *O* | See Appendix C.2.2.4 |

| Value | R/O | Description |
|---|---|---|
| "object.container.genre.musicGenre" | O | See Appendix C.2.2.4.1 |
| "object.container.genre.movieGenre" | O | See Appendix C.2.2.4.2 |
| "object.container.channelGroup" | O | See Appendix C.2.2.5 |
| "object.container.channelGroup.audioChannelGroup" | O | See Appendix C.2.2.5.1 |
| "object.container.channelGroup.videoChannelGroup" | O | See Appendix C.2.2.5.2 |
| "object.container.epgContainer" | O | See Appendix C.2.2.6 |
| "object.container.storageSystem" | O | See Appendix C.2.2.7 |
| "object.container.storageVolume" | O | See Appendix C.2.2.8 |
| "object.container.storageFolder" | O | See Appendix C.2.2.9 |
| "object.container.bookmarkFolder" | O | See Appendix 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* | Appendix B.2.1 |
| *res@protocolInfo* | DIDL-Lite | xsd:string | *NO* | Appendix B.2.1.1 |
| *res@importUri* | DIDL-Lite | xsd:anyURI | *NO* | Appendix B.2.1.2 |
| *res@size* | DIDL-Lite | xsd:unsignedLong | *NO* | Appendix B.2.1.3 |
| *res@duration* | DIDL-Lite | xsd:string | *NO* | Appendix B.2.1.4 |
| *res@protection* | DIDL-Lite | xsd:string | *NO* | Appendix B.2.1.5 |
| *res@bitrate* | DIDL-Lite | xsd:unsignedInt | *NO* | Appendix B.2.1.6 |
| *res@bitsPerSample* | DIDL-Lite | xsd:unsignedInt | *NO* | Appendix B.2.1.7 |
| *res@sampleFrequency* | DIDL-Lite | xsd:unsignedInt | *NO* | Appendix B.2.1.8 |
| *res@nrAudioChannels* | DIDL-Lite | xsd:unsignedInt | *NO* | Appendix B.2.1.9 |
| *res@resolution* | DIDL-Lite | xsd:string | *NO* | Appendix B.2.1.10 |
| *res@colorDepth* | DIDL-Lite | xsd:unsignedInt | *NO* | Appendix B.2.1.11 |
| *res@tspec* | DIDL-Lite | xsd:string | *NO* | Appendix B.2.1.12 |
| *res@allowedUse* | DIDL-Lite | CSV (xsd:string) | *NO* | Appendix B.2.1.13 |
| *res@validityStart* | DIDL-Lite | xsd:string | *NO* | Appendix B.2.1.14 |

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| res@validityEnd | DIDL-Lite | xsd:string | NO | Appendix B.2.1.15 |
| res@remainingTime | DIDL-Lite | xsd:string | NO | Appendix B.2.1.16 |
| res@usageInfo | DIDL-Lite | xsd:string | NO | Appendix B.2.1.17 |
| res@rightsInfoURI | DIDL-Lite | xsd:anyURI | NO | Appendix B.2.1.18 |
| res@contentInfoURI | DIDL-Lite | xsd:anyURI | NO | Appendix B.2.1.19 |
| res@recordQuality | DIDL-Lite | CSV (xsd:string) | NO | Appendix B.2.1.20 |
| res@daylightSaving | DIDL-Lite | xsd:string | NO | Appendix B.2.1.21 |

## B.2.1 res

See Appendix 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 ConnectionManager service [CM], Section 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 Appendix D, “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 Appendix D, “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@valididtyEnd* 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 Appendix D, “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 Resource Encoding Extension Properties

**Table B-7: Resource Encoding Extension Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:resExt* | upnp | <XML> | *YES* | Appendix B.3 |
| *upnp:resExt@id* | upnp | xsd:string | *NO* | Appendix B.3.1.1 |
| *upnp:resExt::segmentInfo* | upnp | <XML> | *YES* | Appendix B.16 |
| *upnp:resExt::clockSync* | upnp | <XML> | *NO* | Appendix B.24 |
| *upnp:resExt::DRMInfo* | upnp | <XML> | *YES* | Appendix B.25 |
| *upnp:resExt::isSyncAnchor* | upnp | xsd:boolean | *NO* | Appendix B.15 |
| *upnp:resExt::componentInfo* | upnp | <XML> | *YES* | Appendix B.15 |

### B.3.1 *upnp:resExt*

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: *YES*

**Description**: This property is used to "extend" the object's res property whose *res@id* property value exactly equals the *upnp:resExt@id* property. The *upnp:resExt* property contains additional data that embellishes the information held within the dependent properties of the associated *res* property. This additional data might otherwise be added directly to the associated res property as additional dependent properties if it were not for some inherent limitations with the res property. For example, dependent properties of the *res* property are represented as XML attributes which make it cumbersome to hold an XML fragment.

**Default Value**: None – The property is OPTIONAL.

#### B.3.1.1 upnp:resExt@id

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This REQUIRED property identifies the object's *res* property that is extended by this instance of the *upnp:resExt* property. Its format and value are vendor-defined, but at all times (within a given object), all instances of the *upnp:resExt@id* property MUST contain a unique value. Additionally, the value of each *upnp:resExt* property MUST equal the value of exactly one *res@id* property.

**Default Value**: N/A. This property is REQUIRED if the *upnp:resExt* property is present.

## B.4 Contributor-related Properties

**Table B-8: Contributor-related Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:artist* | upnp | xsd:string | *YES* | Appendix B.4.1 |
| *upnp:artist@role* | upnp | xsd:string | *NO* | Appendix B.4.1.1 |
| *upnp:actor* | upnp | xsd:string | *YES* | Appendix B.4.2 |
| *upnp:actor@role* | upnp | xsd:string | *NO* | Appendix B.4.2.1 |
| *upnp:author* | upnp | xsd:string | *YES* | Appendix B.4.3 |
| *upnp:author@role* | upnp | xsd:string | *NO* | Appendix B.4.3.1 |
| *upnp:producer* | upnp | xsd:string | *YES* | Appendix B.4.4 |
| *upnp:director* | upnp | xsd:string | *YES* | Appendix B.4.5 |
| *dc:publisher* | dc | xsd:string | *YES* | Appendix B.4.6 |
| *dc:contributor* | dc | xsd:string | *YES* | Appendix B.4.7 |

### B.4.1 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.4.1.1 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.4.2 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.4.2.1 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.4.3 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.4.3.1 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.4.4 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.4.5 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.4.6 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.4.7 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.5 Affiliation-related Properties

**Table B-9: Affiliation-related Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:genre | upnp | xsd:string | YES | Appendix B.5.1 |
| upnp:genre@id | upnp | xsd:string | NO | Appendix B.5.1.1 |
| upnp:genre@extended | upnp | CSV (xsd:string) | NO | Appendix B.5.1.2 |
| upnp:album | upnp | xsd:string | YES | Appendix B.5.2 |
| upnp:playlist | upnp | xsd:string | YES | Appendix B.5.3 |

### B.5.1 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.5.1.1 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.5.1.2 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.5.2 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.5.3 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.6 Associated Resources Properties

### Table B-10: Associated Resources Properties Overview

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:albumArtURI* | upnp | xsd:anyURI | *YES* | Appendix B.6.1 |
| *upnp:artistDiscographyURI* | upnp | xsd:anyURI | *NO* | Appendix B.6.2 |
| *upnp:lyricsURI* | upnp | xsd:anyURI | *NO* | Appendix B.6.3 |
| *dc:relation* | dc | xsd:string | *YES* | Appendix B.6.4 |

### B.6.1 *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.6.2 *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.6.3 *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.6.4 *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.7 Storage-Related Properties

**Table B-11: Storage-Related Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:storageTotal* | upnp | xsd:long | *NO* | Appendix B.7.1 |
| *upnp:storageUsed* | upnp | xsd:long | *NO* | Appendix B.7.2 |
| *upnp:storageFree* | upnp | xsd:long | *NO* | Appendix B.7.3 |
| *upnp:storageMaxPartition* | upnp | xsd:long | *NO* | Appendix B.7.4 |
| *upnp:storageMedium* | upnp | xsd:string | *NO* | Appendix B.7.5 |

### B.7.1 *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.7.2 *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.7.3 *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.7.4 *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.7.5 *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.7.5.1 allowedValueList for the upnp:storageMedium Property

See Table 2-4, “allowedValueList for PlaybackStorageMedium and RecordStorageMedium” of the AVTransport:1 service specification.

## B.8 General Description (mainly for UI purposes) Properties

Table B-12: General Description (mainly for UI purposes) Properties Overview

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| dc:description | dc | xsd:string | NO | Appendix B.8.1 |
| upnp:longDescription | upnp | xsd:string | NO | Appendix B.8.2 |
| upnp:icon | upnp | xsd:anyURI | NO | Appendix B.8.3 |
| upnp:region | upnp | xsd:string | NO | Appendix B.8.4 |
| dc:rights | dc | xsd:string | YES | Appendix B.8.5 |
| dc:date | dc | xsd:string | NO | Appendix B.8.6 |
| dc:date@ upnp:daylightSaving | dc | xsd:string | NO | Appendix B.8.6.1 |
| dc:language | dc | xsd:string | YES | Appendix B.8.7 |
| upnp:playbackCount | upnp | xsd:int | NO | Appendix B.8.8 |
| upnp:lastPlaybackTime | upnp | xsd:string | NO | Appendix B.8.9 |
| upnp:lastPlaybackTime @daylightSaving | upnp | xsd:string | NO | Appendix B.8.9.1 |
| upnp:lastPlaybackPosition | upnp | xsd:string | NO | Appendix B.8.10 |
| upnp:recordedStartDateTime | upnp | xsd:string | NO | Appendix B.8.11 |
| upnp:recordedStartDateTime @daylightSaving | upnp | xsd:string | NO | Appendix B.8.11.1 |
| upnp:recordedDuration | upnp | xsd:string | NO | Appendix B.8.12 |
| upnp:recordedDayOfWeek | upnp | xsd:string | NO | Appendix B.8.13 |
| upnp:srsRecordScheduleID | upnp | xsd:string | NO | Appendix B.8.14 |
| upnp:srsRecordTaskID | upnp | xsd:string | NO | Appendix B.8.15 |
| upnp:recordable | upnp | xsd:boolean | NO | Appendix B.8.16 |

### B.8.1 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.8.2 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.8.3 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.8.4 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.8.5 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.8.6 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.8.6.1 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.8.6.1.1 allowedValueList for the dc:date@upnp:daylightSaving Property

See Table B-6 in Appendix B.2.1.21.1, “allowedValueList for the *res@daylightSaving* Property”.

### B.8.7 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.8.8 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.8.9 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 Appendix D, “EBNF Syntax Definitions”.

The criteria for determining when the content has been played last, is device dependent.

**Default Value**: None.

#### B.8.9.1 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.8.9.1.1 allowedValueList for the upnp:lastPlaybackTime@daylightSaving Property

See Table B-6 in Appendix B.2.1.21.1, “allowedValueList for the *res@daylightSaving* Property”.

### B.8.10 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 Appendix D, “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.8.11 *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 Appendix D, “EBNF Syntax Definitions”.

**Default Value**: None.

#### B.8.11.1 *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.8.11.1.1 allowedValueList for the *upnp:recordedStartDateTime@daylightSaving* Property

See Table B-6 in Appendix B.2.1.21.1, “allowedValueList for the *res@daylightSaving* Property”.

### B.8.12 *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 Appendix D, “EBNF Syntax Definitions”.

**Default Value**: None.

### B.8.13 *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-13. Ascending: first table entry first.

**Default Value**: None.

#### B.8.13.1 allowedValueList for the upnp:recordedDayOfWeek Property

Table B-13: 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.8.14 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.8.15 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.8.16 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.9 Recorded Object-related Properties

Table B-14: Recorded Object-related Properties Overview

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:programTitle | upnp | xsd:string | NO | Appendix B.9.1 |
| upnp:seriesTitle | upnp | xsd:string | NO | Appendix B.9.2 |
| upnp:programID | upnp | xsd:string | NO | Appendix B.9.3 |

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:programID@type | upnp | xsd:string | NO | Appendix B.9.3.1 |
| upnp:seriesID | upnp | xsd:string | NO | Appendix B.9.4 |
| upnp:seriesID@type | upnp | xsd:string | NO | Appendix B.9.4.1 |
| upnp:channelID | upnp | xsd:string | YES | Appendix B.9.5 |
| upnp:channelID@type | upnp | xsd:string | NO | Appendix B.9.5.1 |
| upnp:channelID@distriNetworkName | upnp | xsd:string | NO | Appendix B.9.5.2 |
| upnp:channelID@distriNetworkID | upnp | xsd:string | NO | Appendix B.9.5.3 |
| upnp:episodeCount | upnp | xsd:unsignedInt | NO | Appendix B.9.5.2 |
| upnp:episodeNumber | upnp | xsd:unsignedInt | NO | Appendix B.9.7 |
| upnp:programCode | upnp | xsd:string | NO | Appendix B.9.8 |
| upnp:programCode@type | upnp | xsd:string | NO | Appendix B.9.8.1 |
| upnp:rating | upnp | xsd:string | NO | Appendix B.9.9 |
| upnp:rating@type | upnp | xsd:string | NO | Appendix B.9.9.1 |
| upnp:episodeType | upnp | xsd:string | NO | Appendix B.9.10 |

### B.9.1 *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.9.2 *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.9.3 *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.9.3.1 *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.9.4 *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.9.4.1 *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.9.5 *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.9.5.1 *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.9.5.2 *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.9.5.3 *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.9.6 *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.9.7 *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.9.8 *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.9.8.1 *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.9.9 *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.9.9.1 *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.9.10 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.10 User Channel and EPG Related Properties

**Table B-15: User Channel and EPG Related Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:channelGroupName | upnp | xsd:string | NO | Appendix B.10.1 |
| upnp:channelGroupName@id | upnp | xsd:string | NO | Appendix B.10.1.1 |
| upnp:callSign | upnp | xsd:string | NO | Appendix B.10.2 |
| upnp:networkAffiliation | upnp | xsd:string | NO | Appendix B.10.3 |
| upnp:serviceProvider | upnp | xsd:string | NO | Appendix B.10.4 |
| upnp:price | upnp | xsd:float | YES | Appendix B.10.5 |
| upnp:price@currency | upnp | xsd:string | NO | Appendix B.10.5.1 |
| upnp:payPerView | upnp | xsd:boolean | NO | Appendix B.10.6 |
| upnp:epgProviderName | upnp | xsd:string | NO | Appendix B.10.7 |
| upnp:dateTimeRange | upnp | xsd:string | NO | Appendix B.10.8 |
| upnp:dateTimeRange @daylightSaving | upnp | xsd:string | NO | Appendix B.10.8.1 |

### B.10.1 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.10.1.1 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.10.2 *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.10.3 *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.10.4 *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.10.5 *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.10.5.1 *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.10.6 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.10.7 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.10.8 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 Appendix D, "EBNF Syntax Definitions".

**Default Value**: None.

#### B.10.8.1 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.10.8.1.1 allowedValueList for the upnp:dateTimeRange@daylightSaving Property

See Table B-6 in Appendix B.2.1.21.1, "allowedValueList for the res@daylightSaving Property".

## B.11 Preserved Program Properties

**Table B-16: Preserved Program Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:programPreserved | upnp | xsd:string | NO | Appendix B.11.1 |
| upnp:programPreserved @startTime | upnp | xsd:string | NO | Appendix B.11.1.2 |
| upnp:programPreserved @startTimeDaylightSaving | upnp | xsd:string | NO | Appendix B.11.1.3 |

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:programPreserved @endTime | upnp | xsd:string | NO | Appendix B.11.1.4 |
| upnp:programPreserved @endTimeDaylightSaving | upnp | xsd:string | NO | Appendix B.11.1.5 |
| upnp:preservedTimeRange | upnp | xsd:string | YES | Appendix B.11.2 |
| upnp:preservedTimeRange @startTime | upnp | xsd:string | NO | Appendix B.11.2.1 |
| upnp:preservedTimeRange @startTimeDaylightSavings | upnp | xsd:string | NO | Appendix B.11.2.2 |
| upnp:preservedTimeRange @endTime | upnp | xsd:string | NO | Appendix B.11.2.3 |
| upnp:preservedTimeRange @endTimeDaylightSavings | upnp | xsd:string | NO | Appendix B.11.2.4 |
| upnp:programList | upnp | <XML> | NO | Appendix B.11.3 |
| upnp:programList::program | upnp | xsd:string | YES | Appendix B.11.3.1 |
| upnp:programList::program @preserved | upnp | xsd:boolean | NO | Appendix B.11.3.1.1 |

## B.11.1 *upnp:programPreserved*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:programPreserved* property, when present, is used to describe transitory (broadcast) content, which remains available beyond its scheduled broadcast time as indicated by the *upnp:scheduledStartTime* property. When the value of this property is "*ONGOING*" or "*COMPLETED*", this indicates that a contiguous portion of the program content is still temporarily preserved, and may be permanently recorded. In addition, if a *res* property is present, then it is also possible to play this preserved content. If this property is not present, this indicates that this object does not have preserved content.

**Default Value**: N/A.

### B.11.1.1.1 allowedValueList for the *upnp:programPreserved* Property

Table B-17: allowedValueList for the *upnp:programPreserved* Property

| Value | R/O | Description |
|---|---|---|
| "*ONGOING*" | *O* | The transitory program content is preserved, and the ContentDirectory service is still accumulating content for this program. |
| "*COMPLETED*" | *O* | The preservation of the content has ended. |

## B.11.1.2 *upnp:programPreserved@startTime*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:programPreserved@startTime* property indicates the time at which the preservation of broadcast program content started. If the value of the *upnp:programPreserved@startTime* property would otherwise be equal to the value of *the upnp:scheduledStartTime* property, then this property MUST be omitted. Presence of this property indicates that preservation of the program started after its scheduled start time. The format of this property is defined in Appendix D.1.

Note that this property is volatile in nature, and may change at any time, for example when the buffer becomes full. It is RECOMMENDED to implement the *Tracking Changes Option* for objects containing this property so that control points can be informed of the change. It is RECOMMENDED to send out object modification events resulting from a modification of this property at a moderate rate (for example, not lower than 15 seconds). When the ContentDirectory service implementation does not have the *Tracking Changes Option* implemented for these objects, the control point should refresh the property value by means of a *Browse()* or *Search()* on that particular object just prior to using this property, for example for creating a record schedule.

**Default Value**: None.

#### B.11.1.3 *upnp:programPreserved@startTimeDaylightSaving*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:programPreserved@startTimeDaylightSaving* property indicates whether the time value used in the *upnp:programPreserved@startTime* 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:programPreserved@startTime* property is expressed in local time. Whenever the time value in the *upnp:programPreserved@startTime* property is expressed in absolute time, the *upnp:programPreserved @startTimeDaylightSaving* property MUST not be present on output and MUST be ignored on input.

**Default Value**: "*UNKNOWN*".

##### B.11.1.3.1 allowedValueList for the *upnp:programPreserved@startTimeDaylight Saving* Property

See Table B-6 in Appendix B.2.1.21.1, "allowedValueList for the *res@daylightSaving* Property".

#### B.11.1.4 *upnp:programPreserved@endTime*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:programPreserved@endTime* property indicates the time at which the preservation of broadcast program content stopped. If the ContentDirectory service is still accumulating content for this program, then this property MUST be omitted. If the value of the *upnp:programPreserved@endTime* property would otherwise be equal to the value of the *upnp:scheduledEndTime* property, then this property MUST be omitted. If the property is omitted, the value of the *upnp:programPreserved* property can be examined to determine whether the preservation is still ongoing or has completed at the scheduled end time. Presence of this property indicates that preservation of the program ended before its scheduled end time. The format of this property is defined in Appendix D.1.

**Default Value**: None.

#### B.11.1.5 *upnp:programPreserved@endTimeDaylightSaving*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:programPreserved@endTimeDaylightSaving* property indicates whether the time value used in the *upnp:programPreserved@endTime* 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:programPreserved@endTime* property is expressed in local time. Whenever the time value in the *upnp:programPreserved@endTime* property is expressed in absolute time, the *upnp:programPreserved@endTimeDaylightSaving* property MUST not be present on output and MUST be ignored on input.

**Default Value**: "*UNKNOWN*".

#### B.11.1.5.1 allowedValueList for the *upnp:programPreserved@endTime DaylightSaving* Property

See Table B-6 in Appendix B.2.1.21.1, "allowedValueList for the *res@daylightSaving* Property".

### B.11.2 *upnp:preservedTimeRange*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *YES*

**Description**: The *upnp:preservedTimeRange* property is used for transitory (broadcast) content, which is available now but may not be available in the future. If this property is present, this indicates that a contiguous portion of the content associated with the object is being temporarily preserved. The contiguous fragment of preserved content is identified by the *upnp:preservedTimeRange@startTime* and *upnp:preservedTimeRange@endTime* properties, which represent the beginning and the end of the fragment. The value of this property is empty.

**Default Value**: None.

#### B.11.2.1 *upnp:preservedTimeRange@startTime*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:preservedTimeRange@startTime* property identifies the beginning of the preserved content fragment, measured in units of time. The format of this property defined in Appendix D.1.

**Default Value**: N/A – The property is REQUIRED when the *upnp:preservedTimeRange* property is present.

#### B.11.2.2 *upnp:preservedTimeRange@startTimeDaylightSaving*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:preservedTimeRange@startTimeDaylightSaving* property indicates whether the time value used in the *upnp:preservedTimeRange@startTime* 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:preservedTimeRange@startTime* property is expressed in local time. Whenever the time value in the *upnp:preservedTimeRange@startTime* property is expressed in absolute time, the *upnp:preservedTimeRange@startTimeDaylightSaving* property MUST not be present on output and MUST be ignored on input.

**Default Value**: "*UNKNOWN*".

#### B.11.2.2.1 allowedValueList for the *upnp:preservedTimeRange@startTime DaylightSaving* Property

See Table B-6 in Appendix B.2.1.21.1, "allowedValueList for the *res@daylightSaving* Property".

#### B.11.2.3 *upnp:preservedTimeRange@endTime*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:preservedTimeRange@endTime* property identifies the end of the preserved content fragment, measured in units of time. If the ContentDirectory service is still accumulating content for the fragment whose beginning is indicated by the *upnp:preservedTimeRange@startTime* property, then this property MUST be omitted. The format of this property defined in Appendix D.1.

**Default Value**: None.

### B.11.2.4 *upnp:preservedTimeRange@endTimeDaylightSaving*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:preservedTimeRange@endTimeDaylightSaving* property indicates whether the time value used in the *upnp:preservedTimeRange@endTime* 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:preservedTimeRange@endTime* property is expressed in local time. Whenever the time value in the *upnp:preservedTimeRange@endTime* property is expressed in absolute time, the *upnp:preservedTimeRange@endTimeDaylightSaving* property MUST not be present on output and MUST be ignored on input.

**Default Value**: "*UNKNOWN*".

#### B.11.2.4.1 allowedValueList for the *upnp:preservedTimeRange@endTimeDaylightSaving* Property

See Table B-6 in Appendix B.2.1.21.1, "allowedValueList for the *res@daylightSaving* Property".

### B.11.3 *upnp:programList*

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: *NO*

**Description**: The *upnp:programList* property identifies a collection of broadcast programs, which are associated with the object, for example, a broadcast item representing a channel may have a list of programs that are broadcast on the channel. The *upnp:programList::program* child property identifies the object ID of an object that represent a program. The program list MUST be in ascending order according to the scheduled start time.

**Example**:

```
<upnp:programList>
   <upnp:program preserved="1">PROG_OBJ_ID_1</upnp:program>
   <upnp:program>PROG_OBJ_ID_2</upnp:program>
</upnp:programList>
```

**Default Value**: None.

### B.11.3.1 *upnp:programList::program*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *YES*

**Description**: The *upnp:programList::program* property contains the object ID of the object representing a program in the list. For an example see above.

**Default Value**: None.

#### B.11.3.1.1 *upnp:programList::program@preserved*

**Namespace**: upnp **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

**Description**: The *upnp:programList::program@preserved* property indicates whether the content of the program represented by the object, whose object ID is identified by the *upnp:programList::program* property, is being temporarily preserved. Additionally, the referenced object will also have its *upnp:programPreserved* property present. For an example see above.

**Default Value**: "*0*".

## B.12 Radio Broadcast Properties

**Table B-18: Radio Broadcast Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:radioCallSign* | upnp | xsd:string | *NO* | Appendix B.12.1 |
| *upnp:radioStationID* | upnp | xsd:string | *NO* | Appendix B.12.2 |
| *upnp:radioBand* | upnp | xsd:string | *NO* | Appendix B.12.3 |

### B.12.1 *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.12.2 *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.12.3 *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.12.3.1 allowedValueList for the *upnp:radioBand* Property

**Table B-19: allowedValueList for the *upnp:radioBand* Property**

| Value | R/O | Description |
|---|---|---|
| “*AM*” | *O* | |
| “*FM*” | *O* | |
| “*Shortwave*” | *O* | |
| “*Internet*” | *O* | |
| “*Satellite*” | *O* | |
| *Vendor-defined* | *X* | |

## B.13 Video Broadcast Properties

**Table B-20: Video Broadcast Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:channelNr* | upnp | xsd:int | *NO* | Appendix B.13.1 |

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:channelName | upnp | xsd:string | NO | Appendix B.13.2 |
| upnp:scheduledStartTime | upnp | xsd:string | NO | Appendix B.13.3 |
| upnp:scheduledStartTime @usage | upnp | xsd:string | NO | Appendix B.13.3.1 |
| upnp:scheduledStartTime @daylightSaving | upnp | xsd:string | NO | Appendix B.13.3.2 |
| upnp:scheduledEndTime | upnp | xsd:string | NO | Appendix B.13.4 |
| upnp:scheduledEndTime @daylightSaving | upnp | xsd:string | NO | Appendix B.13.4.1 |
| upnp:scheduledDuration | upnp | xsd:string | NO | Appendix B.13.5 |

### B.13.1 *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.13.2 *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.13.3 *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.13.3.1 *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.13.3.1.1 allowedValueList for the *upnp:scheduledStartTime@usage* Property

**Table B-21: 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.13.3.2 *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.13.3.2.1 allowedValueList for the *upnp:scheduledStartTime@daylightSaving* Property

See Table B-6 in Appendix B.2.1.21.1, "allowedValueList for the *res@daylightSaving* Property".

### B.13.4 *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.13.4.1 *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.13.4.1.1 allowedValueList for the *upnp:scheduledEndTime@daylightSaving* Property

See Table B-6 in Appendix B.2.1.21.1, "allowedValueList for the *res@daylightSaving* Property".

### B.13.5 *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 Appendix D, "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.14 Physical Tuner Status-related Properties

**Table B-22: Physical Tuner Status-related Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:signalStrength* | upnp | xsd:int | *NO* | Appendix B.14.1 |
| *upnp:signalLocked* | upnp | xsd:boolean | *NO* | Appendix B.14.2 |
| *upnp:tuned* | upnp | xsd:boolean | *NO* | Appendix B.14.3 |

### B.14.1 *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.14.2 *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.14.3 *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.15 MultiStream-related Properties

**Table B-23: MultiStream Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:resExt::isSyncAnchor* | upnp | xsd:boolean | *NO* | Appendix B.15.1 |
| *upnp:resExt::componentInfo* | upnp | <XML> | *YES* | Appendix B.15.2 |
| *upnp:resExt::componentInfo ::componentGroup* | upnp | <XML> | *YES* | Appendix B.15.2.1 |
| *upnp:resExt::componentInfo ::componentGroup@groupID* | upnp | xsd:string | *NO* | Appendix B.15.2.1.1 |
| *upnp:resExt::componentInfo ::componentGroup@required* | upnp | xsd:boolean | *NO* | Appendix B.15.2.1.2 |
| *upnp:resExt::componentInfo ::componentGroup ::component* | upnp | <XML> | *YES* | Appendix B.15.2.1.3 |
| *upnp:resExt::componentInfo ::componentGroup ::component@componentID* | upnp | xsd:string | *NO* | Appendix B.15.2.1.3.1 |
| *upnp:resExt::componentInfo ::componentGroup ::component@supportive* | upnp | xsd:boolean | *NO* | Appendix B.15.2.1.3.2 |
| *upnp:resExt::componentInfo ::componentGroup ::component@supportID* | upnp | xsd:string | *NO* | Appendix B.15.2.1.3.3 |
| *upnp:resExt::componentInfo ::componentGroup ::component ::componentClass* | upnp | xsd:string | *NO* | Appendix B.15.2.1.3.4 |
| *upnp:resExt::componentInfo ::componentGroup ::component::contentType* | upnp | xsd:string | *NO* | Appendix B.15.2.1.3.5 |

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:resExt::componentInfo ::componentGroup ::component::contentType @MIMEType | upnp | xsd:string | NO | Appendix B.15.2.1.3.5.1 |
| upnp:resExt::componentInfo ::componentGroup ::component::contentType @extendedType | upnp | xsd:string | NO | Appendix B.15.2.1.3.5.2 |
| upnp:resExt::componentInfo ::componentGroup ::component::language | upnp | xsd:string | NO | Appendix B.15.2.1.3.6 |
| upnp:resExt::componentInfo ::componentGroup ::component::compRes | upnp | <XML> | NO | Appendix B.15.2.1.3.7 |
| upnp:resExt::componentInfo ::componentGroup ::component::compRes::res | upnp | xsd:anyURI | NO | Appendix B.15.2.1.3.7.1 |
| upnp:resExt::componentInfo ::componentGroup ::component::compRes::res @protocolInfo | upnp | xsd:string | NO | Appendix B.15.2.1.3.7.1.1 |
| upnp:resExt::componentInfo ::componentGroup ::component::compRes::res @importUri | upnp | xsd:anyURI | NO | Appendix B.15.2.1.3.7.1.2 |
| upnp:resExt::componentInfo ::componentGroup ::component::compRes ::isSyncAnchor | upnp | xsd:boolean | NO | Appendix B.15.2.1.3.7.2 |
| upnp:resExt::componentInfo ::componentGroup ::component::compRes ::refUDN | upnp | xsd:string | NO | Appendix B.15.2.1.3.7.3 |
| upnp:resExt::componentInfo ::componentGroup ::component::compRes ::refObjectID | upnp | xsd:string | NO | Appendix B.15.2.1.3.7.4 |
| upnp:resExt::componentInfo ::componentGroup ::component::compRes ::refResID | upnp | xsd:string | NO | Appendix B.15.2.1.3.7.5 |

### B.15.1 upnp:resExt::isSyncAnchor

**Namespace**: upnp **Property Data Type**: xsd:boolean **Multi-Valued**: NO

**Description**: This property, if set to "*1*", indicates that the resource identified by the object's *res* property that is extended by this *upnp:resExt* instance contains time-synchronization information (for example the Program Clock Reference in MPEG-2 Transport Streams), such that another resource that is streamed and played back in conjunction with this resource can be properly synchronized during playback. If set to "*0*", this means that the corresponding resource contains no time-synchronization information, and has to rely on another resource whose *upnp:resExt::isSyncAnchor* or *upnp:resExt::componentInfo::componentGroup ::component::compRes::isSyncAnchor* property value is "*1*" to synchronize to during playback. A resource whose *upnp:resExt::isSyncAnchor* property value is "*1*" is further referred to as a *synchronization anchor*. A completely self-contained stream with no associated resources is by default a synchronization anchor.

**Default Value**: "*1*".

### B.15.2 *upnp:resExt::componentInfo*

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: *YES*

**Description**: This property contains the description of a bundle of different media components that are embedded in the corresponding resource identified by the *res* property. Each bundle is represented by one *upnp:resExt::componentInfo* instance and contains a number of semantically coherent components. The components in a bundle are grouped as indicated by the *upnp:resExt::componentInfo::componentGroup* property, and each component is described by the *upnp:resExt::componentInfo::componentGroup ::component* property. The component's class, as identified by the *upnp:resExt::componentInfo ::componentGroup::component::componentClass* property, indicates its media type (for example audio/video).

Note: Each *upnp:resExt::componentInfo* can contain different combinations of AV streams. However,while *upnp:resExt::componentInfo* properties are related to one another, they can also be rendered independently. To playback different *upnp:resExt::componentInfo* properties concurrently on a media renderer, multiple AVTransport instances can be used.

**Default Value**: None.

#### B.15.2.1 *upnp:resExt::componentInfo::componentGroup*

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: *YES*

**Description**: This property contains the description of a set of media components that are grouped together. Each component group contains one or more components (as described by the *upnp:resExt::componentInfo ::componentGroup::component* property).

**Default Value**: None.

#### B.15.2.1.1 *upnp:resExt::componentInfo::componentGroup@groupID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This REQUIRED property identifies the group that contains one or more components. Its format and value can be any arbitrary string, but at all times, all instances of this property MUST contain a unique value within a given object.

**Default Value**: N/A – This property is REQUIRED when the *upnp:resExt::componentInfo ::componentGroup* property is present.

#### B.15.2.1.2 *upnp:resExt::componentInfo::componentGroup@required*

**Namespace**: upnp **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

**Description**: This REQUIRED property indicates whether the component group is essential for a meaningful user experience during playback. For example, a video component is typically essential for a movie experience, while a subtitle component could be considered non-essential. When set to "*1*", then one or more components of this component group MUST be included in the playback of the containing upnp:resExt::componentInfo property. When set to "0", then this group MAY be left out of the playback.

**Default Value**: N/A – This property is REQUIRED when the upnp:resExt::componentInfo ::componentGroup property is present.

### B.15.2.1.3 upnp:resExt::componentInfo::componentGroup::component

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: YES

**Description**: This property gives a description of a component inside a component group. There MUST be at least one component instance inside a component group.

**Default Value**: None.

### B.15.2.1.3.1 upnp:resExt::componentInfo::componentGroup::component @componentID

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: This REQUIRED property identifies the component inside a set of components within a upnp:resExt instance. Its format and value can be any arbitrary string, but at all times all instances of this property MUST contain a unique value within a given object.

**Default Value**: N/A – This property is REQUIRED when the upnp:resExt::componentInfo ::componentGroup::component property is present.

### B.15.2.1.3.2 upnp:resExt::componentInfo::componentGroup::component @supportive

**Namespace**: upnp **Property Data Type**: xsd:boolean **Multi-Valued**: NO

**Description**: This property indicates whether the component is needed as a supportive component to support correct playback of another component in the group. If set to "1", then this component does not need to be selected during playback, unless it is referred to by a selected component via the upnp:resExt ::componentInfo::componentGroup::component@supportID property.

**Default Value**: "0".

### B.15.2.1.3.3 upnp:resExt::componentInfo::componentGroup::component @supportID

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: This property indicates whether the component needs a supportive component in order to achieve correct playback. An example is a subtitle component, which needs an additional timing file containing information on when to display the subtitle text. The value of this property corresponds to the value of the upnp:resExt::componentInfo::componentGroup::component@componentID property of the corresponding supportive component, which MUST have its upnp:resExt::componentInfo ::componentGroup::component@supportive property value set to "1". The supportive component that is referred to MUST be in the same component group.

**Default Value**: None.

### B.15.2.1.3.4 upnp:resExt::componentInfo::componentGroup::component ::componentClass

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: This REQUIRED property indicates the class of the component.

**Default Value**: N/A – This property is REQUIRED when the *upnp:resExt::componentInfo ::componentGroup::component* property is present.

#### B.15.2.1.3.4.1 allowedValueList for the *upnp:resExt::componentInfo ::componentGroup::component::componentClass* Property

**Table B-24: allowedValueList for the *upnp:resExt::componentInfo::componentGroup ::component::componentClass* Property**

| Value | R/O | Description |
|---|---|---|
| "*Audio*" | *O* | Represents elementary content intended for the audible part of the user experience. |
| "*Video*" | *O* | Represents elementary content intended for the visual part of the user experience. |
| "*Caption*" | *O* | Represents a series of words superimposed on some location of the video frames that communicate dialogue to the hearing-impaired or translate foreign dialogue. Normally used only in North America. |
| "*Subtitle*" | *O* | Represents a series of words superimposed on some location of the video frames that communicate dialogue. |
| "*Unknown*" | *O* | Class of the component is unknown |
| *Vendor-defined* | *X* | |

### B.15.2.1.3.5 *upnp:resExt::componentInfo::componentGroup::component ::contentType*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This REQUIRED property indicates the encoding format of the component.

**Default Value**: N/A – This property is REQUIRED.

#### B.15.2.1.3.5.1 *upnp:resExt::componentInfo::componentGroup::component ::contentType@MIMEType*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This REQUIRED property identifies the encoding format of the component, as described by the MIME specification [RFC 2045]. Components MUST represent elementary media types. Multiplexed media formats such as "video/MP2T" MUST be described in terms of their elementary media type(s) by using multiple *upnp:resExt::componentInfo::componentGroup::component* properties to describe each MIMEType contained within the multiplex item.

**Default Value**: N/A – This property is REQUIRED.

#### B.15.2.1.3.5.2 *upnp:resExt::componentInfo::componentGroup::component ::contentType@extendedType*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This REQUIRED property provides possible additional information needed to set up playback of the associated component. The format and value correspond to the 4th field of the *res@protocolInfo* property (see Section B.2.1.1). See also UPnP A/V ConnectionManager service [CM], Section 2.5.2.

**Default Value**: None.

#### B.15.2.1.3.6 upnp:resExt::componentInfo::componentGroup::component ::language

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: This property indicates the language used in the component. Its format and value are the same as defined for the dc:language property (see Section B.8.7).

**Default Value**: None.

#### B.15.2.1.3.7 upnp:resExt::componentInfo::componentGroup::component ::compRes

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: NO

**Description**: This property describes the characteristics of a resource, which is distinct from the content binary identified by the res property which this upnp:resExt property corresponds to. This resource may be described by a separate ContentDirectory service item or a local URI.

**Default Value**: None.

#### B.15.2.1.3.7.1 upnp:resExt::componentInfo::componentGroup::component ::compRes::res

**Namespace**: upnp **Property Data Type**: xsd:anyURI **Multi-Valued**: NO

**Description**: This property indicates a resource, typically a media file, associated with this component. If the value of this 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].

When an implementation supports the QoS res@tspec property, then the implementation MUST incorporate any additional bandwidth caused by transferring this URI. When additional compRes URIs are selectable, the res@tspec property MUST be increased by the bandwidth of the maximum bandwidth from the selectable components.

**Default Value**: None.

#### B.15.2.1.3.7.1.1 upnp:resExt::componentInfo::componentGroup::component ::compRes::res@protocolInfo

**Namespace**: upnp **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 ConnectionManager service [CM], Section 2.5.2).

**Default Value**: N/A – The property is REQUIRED when the upnp:resExt::componentInfo ::componentGroup::component::compRes::res property is present.

#### B.15.2.1.3.7.1.2 upnp:resExt::componentInfo::componentGroup::component ::compRes::res@importUri

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: This property indicates the URI via which the component resource can be imported to the ContentDirectory service via the ImportResource() action or HTTP POST. The upnp:resExt ::componentInfo::componentGroup::component::compRes::res@importUri property identifies a download portal for the associated upnp:resExt::componentInfo::componentGroup::component::compRes::res property of a specific target component. 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 component by setting the target component's *upnp:resExt::componentInfo::componentGroup::component::compRes::res* property value to a URI for that content, which MAY or MAY NOT be the same URI as the one specified in the *upnp:resExt::componentInfo::componentGroup::component::compRes::res@importUri* property, depending on the ContentDirectory service implementation.

**Default Value**: None.

### B.15.2.1.3.7.2 *upnp:resExt::componentInfo::componentGroup::component ::compRes::isSyncAnchor*

**Namespace**: upnp **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

**Description**: This property, if set to "*1*", indicates that the component resource identified by the *upnp:resExt::componentInfo::componentGroup::component::compRes::res* property contains time-synchronization information. If this property is set to "*0*", the corresponding resource contains no time-synchronization information. See the *upnp:resExt::isSyncAnchor* property in section B.15.1 for futher details on usage of timing information.

**Default Value**: "*0*".

### B.15.2.1.3.7.3 *upnp:resExt::componentInfo::componentGroup::component ::compRes::refUDN*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This property indicates the UDN of the MediaServer device that contains the object which provides additional metadata describing the resource identified by the *upnp:resExt::componentInfo ::componentGroup::component::compRes* property.

Note: A control point can use this value to perform a *Browse()* or a *Search()* to obtain more detailed information on this component resource. The *res* property that is identified by the *upnp:resExt ::componentInfo::componentGroup::component::compRes::refResID* property is located within the object which is identified by the *upnp:resExt::componentInfo::componentGroup::component::compRes ::refObjectID* property.

**Default Value**: None.

### B.15.2.1.3.7.4 *upnp:resExt::componentInfo::componentGroup::component ::compRes::refObjectID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This property indicates the object ID of the object that contains additional metadata describing the resource indicated by the *upnp:resExt::componentInfo::componentGroup::component ::compRes* property.

**Default Value**: None.

### B.15.2.1.3.7.5 *upnp:resExt::componentInfo::componentGroup::component ::compRes::refResID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This property indicates the *res@id* property of the resource within the object, whose object ID is indicated by the *upnp:resExt::componentInfo::componentGroup::component::compRes::refObjectID* property, which contains additional metadata describing the resource indicated by the *upnp:resExt ::componentInfo::componentGroup::component::compRes* property. The value of the *res* property, which is identified by the *upnp:resExt::componentInfo::componentGroup::component::compRes::refResID* property in the referred object, MUST be equal to the value of the *upnp:resExt::componentInfo ::componentGroup::component::compRes:res* property.

**Default Value**: None.

## B.16 Segmentation-related Properties

### Table B-25: Segmentation-related Properties Overview

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:segmentID* | upnp | xsd:string | *YES* | Appendix B.16.1 |
| *upnp:resExt::segmentInfo* | upnp | xsd:string | *NO* | Appendix B.16.2 |
| *upnp:resExt::segmentInfo @baseObjectID* | upnp | xsd:string | *NO* | Appendix B.16.2.1 |
| *upnp:resExt::segmentInfo @baseResID* | upnp | xsd:string | *NO* | Appendix B.16.2.2 |
| *upnp:resExt::segmentInfo ::timeRange* | upnp | xsd:string | *NO* | Appendix B.16.2.3 |
| *upnp:resExt::segmentInfo ::timeRange@start* | upnp | xsd:string | *NO* | Appendix B.16.2.3.1 |
| *upnp:resExt::segmentInfo ::timeRange@end* | upnp | xsd:string | *NO* | Appendix B.16.2.3.2 |
| *upnp:resExt::segmentInfo ::byteRange* | upnp | xsd:string | *NO* | Appendix B.16.2.4 |
| *upnp:resExt::segmentInfo ::byteRange@start* | upnp | xsd:unsignedLong | *NO* | Appendix B.16.2.4.1 |
| *upnp:resExt::segmentInfo ::byteRange@end* | upnp | xsd:unsignedLong | *NO* | Appendix B.16.2.4.2 |
| *upnp:resExt::segmentInfo ::frameRange* | upnp | xsd:string | *NO* | Appendix B.16.2.5 |
| *upnp:resExt::segmentInfo ::frameRange@start* | upnp | xsd:unsignedLong | *NO* | Appendix B.16.2.5.1 |
| *upnp:resExt::segmentInfo ::frameRange@end* | upnp | xsd:unsignedLong | *NO* | Appendix B.16.2.5.2 |

### B.16.1 *upnp:segmentID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *YES*

**Description**: The *upnp:segmentID* property identifies an item representing a segment of the content of this item. The value of this property is the object ID of the segment item. A segment item identified by the *upnp:segmentID* property MUST contain at least one *res* property with a *upnp:resExt::segmentInfo @baseObjectID* property value identifying this item.

**Default Value**: None.

### B.16.2 *upnp:resExt::segmentInfo*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:resExt::segmentInfo* property is a sub-element of *upnp:resExt* property. It contains the segment information specific to a certain *res* property of the containing item.

**Default Value**: None.

### B.16.2.1 *upnp:resExt::segmentInfo@baseObjectID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:resExt::segmentInfo@baseObjectID* property contains the object ID of the base content item that this segment descriptor is associated with. This property is REQUIRED when the *upnp:resExt::segmentInfo* property is present.

**Default Value**: N/A – The property is REQUIRED when the *upnp:resExt::segmentInfo* property is present.

### B.16.2.2 *upnp:resExt::segmentInfo@baseResID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This property identifies the value of the *res@id* property of the base content item with which this segment descriptor is associated. This property is REQUIRED when the *upnp:resExt::segmentInfo* property is present.

**Default Value**: N/A – The property is REQUIRED when the *upnp:resExt::segmentInfo* property is present.

### B.16.2.3 *upnp:resExt::segmentInfo::timeRange*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This property identifies a contiguous fragment of the base content measured in units of time, which starts at *upnp:resExt::segmentInfo::timeRange@start*, and ends at *upnp:resExt::segmentInfo ::timeRange@end*. This property is a REQUIRED child element of the *upnp:resExt::segmentInfo* property. The value of this property is empty.

**Default Value**: N/A – The property is REQUIRED when the *upnp:resExt::segmentInfo* property is present.

#### B.16.2.3.1 *upnp:resExt::segmentInfo::timeRange@start*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This property is a REQUIRED property of the associated *upnp:resExt::segmentInfo::timeRange* property. It indicates the start point of the time range. The format of this property MUST comply with the `time` syntax as defined in Appendix D, "EBNF Syntax Definitions". The *upnp:timeRange* values are measured relative to the base content with 00:00:00 indicating the start of the base content. The *upnp:timeRange* values are subject to the following conditions:

- The *upnp:timeRange* parameters MUST specify a non-NULL fragment of the base content.
- The *upnp:timeRange* parameters MUST comply with:
  *upnp:timeRange@start* < *upnp:timeRange@end*.

**Default Value**: N/A – The property is REQUIRED when the *upnp:resExt::segmentInfo::timeRange* property is present.

#### B.16.2.3.2 *upnp:resExt::segmentInfo::timeRange@end*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This property is a REQUIRED property of the associated *upnp:resExt::segmentInfo::timeRange* property. It indicates the end point of the time range. The format of this property MUST comply with the `time` syntax as defined in Appendix D, "EBNF Syntax Definitions".

**Default Value**: N/A – The property is REQUIRED when the *upnp:resExt::segmentInfo::timeRange* property is present.

### B.16.2.4 upnp:resExt::segmentInfo::byteRange

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This property identifies a contiguous fragment of the base content measured in units of bytes, which starts at *upnp:resExt::segmentInfo::byteRange@start*, and ends at *upnp:resExt::segmentInfo ::byteRange@end*. The value of this property is empty.

The *upnp:byteRange* values are measure relative to the base content with 0 indicating the start of the base content. The *upnp:byteRange* values are subject to the following conditions:

- The *upnp:byteRange* parameters MUST specify a non-NULL fragment of the base content.
- The *upnp:byteRange* parameters MUST comply with:
  *upnp:byteRange@start* < *upnp:byteRange@end*.

**Default Value**: None.

#### B.16.2.4.1 upnp:resExt::segmentInfo::byteRange@start

**Namespace**: upnp **Property Data Type**: xsd:unsignedLong **Multi-Valued**: *NO*

**Description**: This property is a REQUIRED property of the associated *upnp:resExt::segmentInfo ::byteRange* property. It indicates the start point of the byte range. This property is a 64-bit unsigned integer.

**Default Value**: N/A – The property is REQUIRED when the *upnp:resExt::segmentInfo::byteRange* property is present.

#### B.16.2.4.2 upnp:resExt::segmentInfo::byteRange@end

**Namespace**: upnp **Property Data Type**: xsd:unsignedLong **Multi-Valued**: *NO*

**Description**: This property is a REQUIRED property of the associated *upnp:resExt::segmentInfo ::byteRange* property. It indicates the end point of the byte range. This property is a 64-bit unsigned integer.

**Default Value**: N/A – The property is REQUIRED when the *upnp:resExt::segmentInfo::byteRange* property is present.

### B.16.2.5 upnp:resExt::segmentInfo::frameRange

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This property identifies a contiguous fragment measured in units of frames, which starts at *upnp:resExt::segmentInfo::frameRange@start*, and ends at *upnp:resExt::segmentInfo::frameRange@end*. The value of this property is empty.

The *upnp:frameRange* values are measured relative to the base content with 0 indicating the start of the base content. The *upnp:frameRange* values are subject to the following conditions:

- The *upnp:frameRange* parameters MUST specify a non-NULL fragment of the base content.
- The *upnp:frameRange* parameters MUST comply with:
  *upnp:frameRange@start* < *upnp:frameRange@end*.

**Default Value**: None.

#### B.16.2.5.1 upnp:resExt::segmentInfo::frameRange@start

**Namespace**: upnp **Property Data Type**: xsd:unsignedLong **Multi-Valued**: *NO*

**Description**: This property is a REQUIRED property of the associated *upnp:resExt::segmentInfo ::frameRange* property. It indicates the start point of the frame range. This property is a 64-bit unsigned integer.

**Default Value**: N/A – The property is REQUIRED when the *upnp:resExt::segmentInfo::frameRange* property is present.

#### B.16.2.5.2 upnp:resExt::segmentInfo::frameRange@end

**Namespace**: upnp **Property Data Type**: xsd:unsignedLong **Multi-Valued**: *NO*

**Description**: This property is a REQUIRED property of the associated *upnp:resExt::segmentInfo ::frameRange* property. It indicates the end point of the frame range. This property is a 64-bit unsigned integer.

**Default Value**: N/A – The property is REQUIRED when the *upnp:resExt::segmentInfo::frameRange* property is present.

## B.17 Bookmark-related Properties

**Table B-26: Bookmark-related Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *@neverPlayable* | DIDL-Lite | xsd:boolean | *NO* | Appendix B.17.1 |
| *upnp:bookmarkID* | upnp | xsd:string | *YES* | Appendix B.17.2 |
| *upnp:bookmarkedObjectID* | upnp | xsd:string | *NO* | Appendix B.17.3 |
| *upnp:deviceUDN* | upnp | xsd:string | *NO* | Appendix B.17.4 |
| *upnp:deviceUDN@serviceType* | upnp | xsd:string | *NO* | Appendix B.17.4.1 |
| *upnp:deviceUDN@serviceId* | upnp | xsd:string | *NO* | Appendix B.17.4.2 |
| *upnp:stateVariableCollection* | upnp | xsd:string | *YES* | Appendix B.17.5 |
| *upnp:stateVariableCollection@serviceName* | upnp | xsd:string | *NO* | Appendix B.17.5.1 |
| *upnp:stateVariableCollection@rcsInstanceType* | upnp | xsd:string | *NO* | Appendix B.17.5.2 |

### B.17.1 @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.17.2 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.17.3 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.17.4 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.17.4.1 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.17.4.2 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.17.5 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.17.5.1 *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.17.5.2 *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.17.5.2.1 allowedValueList for the *upnp:stateVariableCollection@rcsInstanceType* Property

**Table B-27: allowedValueList for the upnp:stateVariableCollection@rcsInstanceType Property**

| Value | R/O | Description |
|---|---|---|
| "*pre-mix*" | *R* | |
| "*post-mix*" | *R* | |

## B.18 Miscellaneous Properties

**Table B-28: Miscellaneous Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:DVDRegionCode* | upnp | xsd:int | *NO* | Appendix B.18.1 |

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:originalTrackNumber | upnp | xsd:int | NO | Appendix B.18.2 |
| upnp:toc | upnp | xsd:string | NO | Appendix B.18.3 |
| upnp:userAnnotation | upnp | xsd:string | YES | Appendix B.18.4 |
| desc | DIDL-Lite | xsd:string | YES | Appendix B.18.5 |
| desc@nameSpace | DIDL-Lite | xsd:string | NO | Appendix B.18.5.1 |

### B.18.1 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.18.2 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.18.3 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.18.4 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.18.5 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 contained metadata. The contents of each desc property MUST be associated with only one namespace.

The desc property can appear as a `<desc>` element anywhere in a valid *DIDL-Lite XML Document* where an element is allowed to appear.

The desc property is used to associate blocks of other XML-based metadata with a given ContentDirectory service object. Examples of other XML-based metadata include DIG35, MPEG7, RDF, XrML, etc. The desc property could also be used to contain vendor-specific content ratings information, digitally signed rights descriptions, etc.

Allowing the desc property to contain only elements from the namespace defined by the @nameSpace property allows control point vendors to selectively deploy support for a given namespace using parser *plug-in* techniques. desc properties that have an unknown namespace specified in their @nameSpace property should be ignored by the control point.

**Default Value**: None.

### B.18.5.1 *desc@nameSpace*

**Namespace**: DIDL-Lite **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The REQUIRED *@nameSpace* property identifies the namespace of the metadata, contained in the associated independent *desc* property. Since the dependent *@nameSpace* property can only appear once for its associated independent *desc* property, the contents of each *desc* property can be associated with only one namespace.

**Default Value**: None.

## B.19 Object Tracking Properties

The following properties are used in tracking changes on objects; they can be used by control points to determine exactly what object change resulted in an update of the *SystemUpdateID* state variable, even across periods when the control point and/or the ContentDirectory service implementation has been *off-line*. These properties are only allowed on objects if the ContentDirectory service implementation supports the *Tracking Changes Option*. If the ContentDirectory service implementation is tracking changes on a container object, that container MUST have the *upnp:containerUpdateID, upnp:objectUpdateID*, *upnp:totalDeletedChildCount* and *@childCount* properties exposed. If the ContentDirectory service is tracking changes to a non-container object, that object MUST have the *upnp:objectUpdateID* property exposed. If the ContentDirectory service is tracking changes to a particular resource, then that resource MUST have the *res@updateCount* attribute.

**Table B-29: Object Tracking Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:containerUpdateID* | upnp | xsd:unsignedInt | *NO* | Appendix B.19.1 |
| *upnp:objectUpdateID* | upnp | xsd:unsignedInt | *NO* | Appendix B.19.2 |
| *upnp:totalDeletedChildCount* | upnp | xsd:unsignedInt | *NO* | Appendix B.19.3 |
| *res@updateCount* | DIDL-Lite | xsd:unsignedInt | *NO* | Appendix B.19.4 |

### B.19.1 *upnp:containerUpdateID*

**Namespace**: upnp **Property Data Type**: xsd:unsignedInt **Multi-Valued**: *NO*

**Description:** The *read-only upnp:containerUpdateID* property is an OPTIONAL property for all *container* objects (that is: objects that are derived from the *container* class) that contains the value of the *SystemUpdateID* state variable generated by the most recent *Container Modification* for that container. Refer to Section 2.2.9, "*Container Modification*" and Section 2.3.5, "*SystemUpdateID*" for details. If implemented, the value of the *upnp:containerUpdateID* property MUST be preserved even while *off-line* except when the *Service Reset Procedure* is invoked. See Section 2.2.1, "*On-line* and *Off-line* Network States" and Section 2.3.7.1, "*Service Reset Procedure*" for details.

Unlike other ContentDirectory service properties, a modification to the *upnp:containerUpdateID* property MUST NOT be treated as an *Object Modification* and MUST NOT cause the *SystemUpdateID* state variable or the object's *upnp:objectUpdateID* property to be updated. Additionally, a modification of the container's *upnp:containerUpdateID* property MUST NOT be treated as a *Container Modification*, which would otherwise trigger a non-terminating sequence of circular updates of the *upnp:containerUpdateID* property.

When a given instance of the ContentDirectory service is initialized, either at its first instantiation or because of a *Service Reset Procedure* invocation, for all containers that expose the *upnp:containerUpdateID* property, the value of that property MUST be set to the value of that container's *upnp:objectUpdateID* property. The ContentDirectory service implementation MUST first initialize its *upnp:objectUpdateID* property values according to the procedure defined in Appendix B.19.2, "*upnp:objectUpdateID*" prior to initializing the *upnp:containerUpdateID* property values.

**Default Value**: None

### B.19.2 *upnp:objectUpdateID*

**Namespace**: upnp **Property Data Type**: xsd:unsignedInt **Multi-Valued**: *NO*

**Description**: The *read-only upnp:objectUpdateID* property is an OPTIONAL property that contains the value the *SystemUpdateID* state variable that was generated when the object experienced its latest *Object Modification*. In other words, the *upnp:objectUpdateID* property represents a last-modified timestamp for the object relative to the *SystemUpdateID* state variable. If implemented, the *upnp:objectUpdateID* property MUST be preserved even while *off-li*ne except when the *Service Reset Procedure* is invoked. See Section 2.2.1, "*On-line* and *Off-line* Network States" and Section 2.3.7.1, "*Service Reset Procedure*" for details.

By definition, an *Object Modification* occurs if, and only if, one or more of the object's properties is added, deleted, or modified. (See Section 2.2.5, "*Object Modification*" for details.). When an object experiences an *Object Modification*, both the *SystemUpdateID* state variable and the object's *upnp:objectUpdateID* property are updated (see Section 2.3.5, "*SystemUpdateID*"). First, the *SystemUpdateID* is incremented to reflect the *Object Modification* to this object and the resulting value of the *SystemUpdateID* state variable is stored within the object's *upnp:objectUpdateID* property.

Unlike other ContentDirectory service properties, a modification to the *upnp:objectUpdateID* property itself MUST NOT be treated as an *Object Modification* and MUST NOT cause the *SystemUpdateID* state variable or the object's *upnp:objectUpdateID* property to be updated. Otherwise, a non-terminating sequence of circular updates of the *upnp:objectUpdateID* property would result. Additionally, for a container object, a modification of the container's *upnp:objectUpdateID* property MUST NOT be treated as a *Container Modification*, which would otherwise trigger a change to the container's *upnp:containerUpdateID*.

When a given instance of the ContentDirectory service is initialized, either at its first instantiation or because of a *Service Reset Procedure* invocation, for all objects that expose the *upnp:objectUpdateID* property, the value of that property MUST be set to a unique value between (inclusive) one ("1") and the total number of objects, currently in the ContentDirectory service, that expose the *upnp:objectUpdateID* property. Consequently, no two *upnp:objectUpdateID* properties will have the same value.

**Default Value**: None.

### B.19.3 *upnp:totalDeletedChildCount*

**Namespace**: upnp **Property Data Type**: xsd:unsignedInt **Multi-Valued**: *NO*

**Description**: The *read-only upnp:totalDeletedChildCount* property is an OPTIONAL property that contains the total number of child objects that have been deleted from a container object since the last initialization. When a container object is first created, the value of it *upnp:totalDeletedChildCount* property MUST be initialized to zero ("0"). Every time an object is deleted, the *upnp:totalDeletedChildCount* property of the object's parent container MUST be incremented by one ("1"). If implemented, the current value the *upnp:totalDeletedChildCount* property MUST be persisted even while *off-line*. See Section 2.2.1, "*On-line* and *Off-line* Network States".

When a given instance of the ContentDirectory service is initialized, either at its first instantiation or because of a *Service Reset Procedure* invocation, for all objects that expose the *upnp:totalDeletedChildCount* property, the value of that property MUST be set to zero ("0").

Note: Due to the fact that the *SystemUpdateID* state variable must be incremented for every *Object Modification*, it will reach its maximum value and cause a *Service Reset Procedure* prior to the *upnp:totalDeletedChildCount* property reaching its maximum value of $2^{32}$-1.

**Default Value**: None.

### B.19.4 *res@updateCount*

**Namespace**: DIDL-Lite **Property Data Type**: xsd:unsignedInt **Multi-Valued**: *NO*

**Description**:The *read-only* *res@updateCount* property is an OPTIONAL property that contains the number of times the implementation detects that a change was made to the content that is referenced by the *res* property's URI since the last initialization. However, the *res@updateCount* property is not incremented for live content (for example an object whose class is "*object.item.videoItem.videoBroadcast*"). When a *res* property is first created, the value of the *res@updateCount* property MUST be initialized to zero ("0") regardless of whether the *res* property contains an initial URI value or not. When the ContentDirectory service implementation detects that the content referenced by the *res* property's URI has changed, the value of the corresponding *res@updateCount* property MUST be incremented by one ("1"). If implemented, the current value the *res@updateCount* property MUST be persisted even while *off-line*. See Section 2.2.1, "*On-line* and *Off-line* Network States".

When a given instance of the ContentDirectory service is initialized, either at its first instantiation or because of a *Service Reset Procedure* invocation, for all objects that expose the *res@updateCount* property, the value of that property MUST be set to zero ("0").

**Default Value**: None.

## B.20 Permission Properties

**Table B-30: Permission Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:inclusionControl* | upnp | xsd:string | *NO* | Appendix B.20.1 |
| *upnp:inclusionControl::role* | upnp | xsd:string | *YES* | Appendix B.20.1.1 |

### B.20.1 *upnp:inclusionControl*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *Read/Write* *upnp:inclusionControl* property is an OPTIONAL property when the *CONTENT_PROTECTION feature* is implemented on the ContentDirectory service. This property provides a mechanism for indicating *Object level access* to an object. For container objects, the *upnp:inclusionControl* property also applies to a container's direct-child items unless overridden an *upnp:inclusionControl* property one or more child item(s). See Appendix F.1.4 for a detailed description. If the *CONTENT_PROTECTION feature* is not implemented then the *upnp:inclusionControl* property MUST NOT appear on any object in the ContentDirectory service implementation. Also, the *upnp:inclusionControl* property MUST contain one or more valid *upnp:inclusionControl::role* elements (See Section B.20.1.1).

**Default Value**: N/A.

#### B.20.1.1 upnp:inclusionControl::role

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: YES

**Description**: The *Read/Write* upnp:inclusionControl::role property is CONDITIONALLY REQUIRED when the *CONTENT_PROTECTION feature* is implemented on a ContentDirectory service and the upnp:inclusionControl property is present. It identifies the specific *Role(s)* being allowed *Object level access* to the object and possibly its direct-child items, if the object is a container. The value of a upnp:inclusionControl::role property MUST be identical to a *Role* `<name>` element implemented by the DeviceProtection service [DP] (See *A_ARG_TYPE_ACLData* state variable). A specific value MUST appear only once in an individual objects upnp:inclusionControl::role property.

**Default Value**: N/A.

## B.21 Ownership Properties

### Table B-31: Ownership Properties Overview

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:objectOwner | upnp | xsd:string | NO | Appendix B.21.1 |
| upnp:objectOwner@lock | upnp | xsd:boolean | NO | Appendix B.21.1.1 |
| upnp:objectOwner::role | upnp | xsd:string | YES | Appendix B.21.1.2 |

### B.21.1 upnp:objectOwner

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The *Read/Write* upnp:objectOwner property is CONDITIONALLY OPTIONAL when the *CONTENT_PROTECTION feature* is implemented on the ContentDirectory service. Otherwise, this property is PROHIBITED. This property provides a mechanism for controlling *Object level access* to an object's upnp:inclusionControl property and additionally the upnp:objectOwner property itself (See Appendix F.2.1 and F.2.2 for additional details).

**Default Value**: N/A.

#### B.21.1.1 upnp:objectOwner@lock

**Namespace**: upnp **Property Data Type**: xsd:boolean **Multi-Valued**: NO

**Description**: The REQUIRED *Read/Write* upnp:objectOwner@lock property indicates whether the upnp:inclusionControl and upnp::objectOwner properties have been put in the *lock* state by the current *owner(s)* (See Appendix F.2.1 and F.2.2 for additional details).

**Default Value**: N/A. This property is REQUIRED if the upnp:objectOwner property is present.

#### B.21.1.2 upnp:objectOwner::role

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: YES

**Description**: The OPTIONAL *Read/Write* upnp:objectOwner::role property indicates whether the object associated with this property has an *Owner*. The value of this property MUST be identical to a *Role* `<name>` element implemented by the DeviceProtection service [DP] (See *A_ARG_TYPE_ACLData* state variable) and MUST NOT be duplicated in the same upnp:objectOwner property. See Appendix F for additional details.

**Default Value**: N/A.

## B.22 Object Linking Properties

### Table B-32: Object Linking Properties Overview

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:objectLink | upnp | <XML> | YES | Appendix B.22.1 |
| upnp:objectLink@groupID | upnp | xsd:string | NO | Appendix B.22.1.1 |
| upnp:objectLink@headObjID | upnp | xsd:string | NO | Appendix B.22.1.2 |
| upnp:objectLink@nextObjID | upnp | xsd:string | NO | Appendix B.22.1.3 |
| upnp:objectLinl@prevObjID | upnp | xsd:string | NO | Appendix B.22.1.4 |
| upnp:objectLink::title | upnp | xsd:string | NO | Appendix B.22.1.5 |
| upnp:objectLink::startObject | upnp | xsd:string | NO | Appendix B.22.1.6 |
| upnp:objectLink::mode | upnp | xsd:string | NO | Appendix B.22.1.7 |
| upnp:objectLink::relatedInfo | upnp | xsd:string | YES | Appendix B.22.1.8 |
| upnp:objectLink::relatedInfo @role | upnp | xsd:string | NO | Appendix B.22.1.8.1 |
| upnp:objectLink::relatedInfo @roleText | upnp | xsd:string | NO | Appendix B.22.1.8.2 |
| upnp:objectLink::startInfo | upnp | <XML> | YES | Appendix B.22.1.9 |
| upnp:objectLink::startInfo @targetObjID | upnp | xsd:string | NO | Appendix B.22.1.9.1 |
| upnp:objectLink::startInfo @targetGroupID | upnp | xsd:string | NO | Appendix B.22.1.9.2 |
| upnp:objectLink::endAction | upnp | <XML> | NO | Appendix B.22.1.10 |
| upnp:objectLink::endAction @action | upnp | xsd:string | NO | Appendix B.22.1.10.1 |
| upnp:objectLink::endAction @targetObjID | upnp | xsd:string | NO | Appendix B.22.1.10.2 |
| upnp:objectLink::endAction @targetGroupID | upnp | xsd:string | NO | Appendix B.22.1.10.3 |
| upnp:objectLinkRef | upnp | <XML> | YES | Appendix B.22.2 |
| upnp:objectLinkRef @groupID | upnp | xsd:string | NO | Appendix B.22.2.1 |
| upnp:objectLinkRef @targetObjID | upnp | xsd:string | NO | Appendix B.22.2.2 |
| upnp:objectLinkRef @targetGroupID | upnp | xsd:string | NO | Appendix B.22.2.3 |
| upnp:objectLinkRef@return | upnp | xsd:string | NO | Appendix B.22.2.4 |
| upnp:objectLinkRef::title | upnp | xsd:string | NO | Appendix B.22.2.5 |
| upnp:objectLinkRef ::startObject | upnp | xsd:string | NO | Appendix B.22.2.6 |

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:objectLinkRef ::relatedInfo | upnp | xsd:string | YES | Appendix B.22.2.7 |
| upnp:objectLinkRef ::relatedInfo@role | upnp | xsd:string | NO | Appendix B.22.2.7.1 |
| upnp:objectLinkRef ::relatedInfo@roleText | upnp | xsd:string | NO | Appendix B.22.2.7.2 |

### B.22.1 upnp:objectLink

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: YES

**Description**: The upnp:objectLink property describes ordered list relationships (previous object, next object, head object) for this object. An object MAY contain multiple upnp:objectLink properties indicating the object is a member of multiple lists. Each list has a unique identifier as indicated by the upnp:objectLink@groupID property. Child properties of the first object of the list describe the title and intended usage of objects in the ordered list.

**Default Value**: None.

### B.22.1.1 upnp:objectLink@groupID

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The REQUIRED upnp:objectLink@groupID property indicates how upnp:objectLink properties are “grouped”. Multiple upnp:objectLink properties within an object having the same value for their upnp:objectLink@groupID property SHOULD be considered as a single unit when making object-to-object linkage decisions.

Default Value: None.

### B.22.1.2 upnp:objectLink@headObjID

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The REQUIRED upnp:objectLink@headObjID property identifies the target object at the start of a list of related objects. The objectLink at the start of a list will have a upnp:objectLink@headObjID property value equal to the @id property of the containing object. The upnp:objectLink property at the start of the list contains child properties describing the contents of the list.

**Default Value**: None.

### B.22.1.3 upnp:objectLink@nextObjID

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The REQUIRED upnp:objectLink@nextObjID property identifies the target object that logically follows this object. The absence of a successor object is indicated by the empty string as the value for this property.

**Default Value**: None.

### B.22.1.4 upnp:objectLink@prevObjID

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The REQUIRED *upnp:objectLink@prevObjID* property identifies the target object that logically precedes this object. The object indicated by this property MUST contain a *upnp:objectLink@nextObjID* property to this object within the same group. The absence of a predecessor object is indicated by the empty string as the value for this property.

**Default Value**: None.

### B.22.1.5 *upnp:objectLink::title*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The REQUIRED *upnp:objectLink::title* property provides a displayable title for the list of objects participating in this *upnp:objectLink* list. The *upnp:objectLink::title* property MUST only appear within *upnp:objectLink* properties at the start of a list.

**Default Value**: None.

### B.22.1.6 *upnp:objectLink::startObject*

**Namespace**: upnp **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

**Description**: The *upnp:objectLink::startObject* property indicates that this object describes a hierarchy of Object Linked lists. If the value of this property is "*1*" then this item SHOULD be used for representing a collection of Object Linked lists corresponding to a complete work. If this item is selected for playback, then the list identified by the corresponding *upnp:objectLinkRef* properties *upnp:objectLinkRef@targetGroupID* and *upnp:objectLinkRef@targetObjID* SHOULD be played. If no *upnp:objectLinkRef* property is found, then this item SHOULD be treated as the initial *upnp:objectLink* playback item.

**Default Value**: "*0*"

### B.22.1.7 *upnp:objectLink::mode*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The REQUIRED *upnp:objectLink::mode* property defines how this list of objects is intended to be processed. The *upnp:objectLink::mode* property MUST only appear within *upnp:objectLink* properties at the start of a list. The following table provides the allowed values for this property.

**Default Value**: None.

**Table B-33: allowedValueList for the *upnp:objectLink::mode* Property**

| Allowed Value | Description |
|---|---|
| *Playback* | The "*Playback*" setting for the *upnp:objectLink::mode* property indicates that members of this ordered Playback list are intended to be played back with seamless transitions between the end of playback of a list object and the start of playback of next object on the list as identified by the *upnp:objectLink::nextObjID* property. |
| *Step* | The "*Step*" setting for the *upnp:objectLink::mode* property indicates that members of this ordered Step list are intended to be played back with an automatic pause (of an arbitrary duration) at the end of playback of each object prior to playback of the next object on this list as identified by the *upnp:objectLink::nextObjID* property. |
| *Index* | The "*Index"* setting for the *upnp:objectLink::mode* property indicates that the metadata of each member within this ordered Index list is intended to be displayed on the control point user interface as the control point would normally display the results of a *Search()* or *Browse()* action. |

| | |
|---|---|
| | When an Index list item is selected, a *upnp:objectLinkRef* property with a matching group ID value is located; a new group ID and object ID are then selected from the *upnp:objectLinkRef@targetGroupID* and *upnp:objectLinkRef@targetObjID* properties of the selected *upnp:objectLinkRef* property. If a *upnp:objectLinkRef* property with a matching group ID is not found, then the selection of this Index list item has no effect.<br>Processing of the newly selected object (if any) and subsequent objects is defined by the *upnp:objectLink@mode* property of the newly selected list. |

### B.22.1.8 *upnp:objectLink::relatedInfo*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *YES*

**Description**: The *upnp:objectLink::relatedInfo* property provides information about the relationship between the objects in this list. The property value provides the subject of the relationship. The *upnp:objectLink::relatedInfo* property MUST only appear within *upnp:objectLink* properties at the start of a list.

**Default Value**: None.

#### B.22.1.8.1 *upnp:objectLink::relatedInfo@role*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The REQUIRED *upnp:objectLink::relatedInfo@role* property provides the type of relationship. The following table provides the allowed values for this property.

**Default Value**: None.

**Table B-34: allowedValueList for the *upnp:objectLink::relatedInfo@role* Property**

| Allowed Value | Description |
|---|---|
| *Actor* | List objects refer to the indicated actor |
| *Scene* | List objects are related to this scene |
| *Subject* | List objects are related to this subject |
| Vendor defined | List of object are related in a vendor-specified way |

#### B.22.1.8.2 *upnp:objectLink::relatedInfo@roleText*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The REQUIRED *upnp:objectLink::relatedInfo@roleText* property provides the text value of the relationship identified by *upnp:objectLink::relatedInfo@role.*

**Default Value**: None.

### B.22.1.9 *upnp:objectLink::startInfo*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *YES*

**Description**: The *upnp:objectLink::startInfo* property provides information to a starting point within a hierarchy of Object Linked lists. If multiple starting points are possible, then multiple *upnp:objectLink::startInfo* properties MAY be specified. The *upnp:objectLink::startInfo* property MUST only appear within *upnp:objectLink* properties at the start of a list. Usage of the property and its dependent properties are RECOMMENDED.

Default Value: None

#### B.22.1.9.1 upnp:objectLink::startInfo@targetGroupID

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The REQUIRED upnp:objectLink::startInfo@targetGroupID property identifies the starting Group ID to be set when starting processing objects associated by upnp:objectLink properties.

**Default Value**: None.

#### B.22.1.9.2 upnp:objectLink::startInfo@targetObjID

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The REQUIRED upnp:objectLink::startInfo@targetObjID property identifies the starting object ID to be set when starting processing objects associated by upnp:objectLink properties.

**Default Value**: None.

### B.22.1.10 upnp:objectLink::endAction

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The upnp:objectLink::endAction property provides information to what action SHOULD be taken if the end of a list is reached. The upnp:objectLink::endAction property MUST only appear within upnp:objectLink properties at the start of a list. Usage of this property and its related dependent properties are RECOMMENDED.

**Default Value**: None.

#### B.22.1.10.1 upnp:objectLink::endAction@action

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The REQUIRED upnp:objectLink::endAction@action property identifies the action to be taken if the end of a list is reached. The following table provides the allowed values for this property.

**Default Value**: None.

**Table B-35: allowedValueList for the upnp:objectLink::endAction@action Property**

| Allowed Value | Description |
|---|---|
| Return | 1. If the end of a Playback or Step list is reached, the next object and next list are determined by the last saved return.<br>2. If no return point is available, the next object and next list are determined the upnp:objectLink::endAction@targetGroupID and upnp:objectLink::endAction@targetObjID if these properties are specified.<br>3. If neither of the above bullets applies, then the presentation ends. |
| Branch | The next object and next list are determined by the upnp:objectLink::endAction@targetGroupID and upnp:objectLink::endAction@targetObjID. |
| Stop | The presentation ends. |

#### B.22.1.10.2 upnp:objectLink::endAction@targetGroupID

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The REQUIRED *upnp:objectLink::endAction@targetGroupID* property identifies the target group for a "*Return*" or "*Branch*" action.

**Default Value**: None.

### B.22.1.10.3 *upnp:objectLink::endAction@targetObjID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The REQUIRED *upnp:objectLink::endAction@targetObjID* property identifies the target group for a "*Return*" or "*Branch*" action.

**Default Value**: None.

### B.22.2 *upnp:objectLinkRef*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *YES*

**Description**: The *upnp:objectLinkRef* property is a reference to a different list of related objects as indicated by the *upnp:objectLinkRef@targetObjID* and *upnp:objectLinkRef@targetGroupID* properties. The display of list titles of related content SHOULD be done in a manner not disruptive to the current list being processed. For instance, by first displaying a small icon the end-user MAY select to display the alternate title. The *upnp:objectLinkRef::title* and *upnp:objectLinkRef::relatedInfo* properties SHOULD override the title and related information of the target list of objects. If these child properties are not specified on the *upnp:objectLinkRef* property, then the properties from the referenced Object Link list SHOULD be used.

**Default Value**: None.

### B.22.2.1 *upnp:objectLinkRef@groupID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The REQUIRED *upnp:objectLinkRef@groupID* property identifies how *upnp:objectLink*/*upnp:objectLinkRef* properties are "grouped". *upnp:objectLinkRef* properties within an object having the same value for their *upnp:objectLink@groupID* property SHOULD be considered as a single unit when making object-to-object linkage decisions.

**Default Value**: None.

### B.22.2.2 *upnp:objectLinkRef@targetGroupID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The REQUIRED *upnp:objectLinkRef@targetGroupID* property provides the group ID of a new list of related objects.

**Default Value**: None.

### B.22.2.3 *upnp:objectLinkRef@targetObjID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The REQUIRED *upnp:objectLinkRef@targetObjID* property provides the target object of a new list of related objects.

**Note:** The target object MAY not be the first element of the list. Information about the target list MAY be obtained by inspecting the *upnp:objectLink@headObjID* property of the target Object Link list using the group specified by the *upnp:objectLinkRef@targetGroupID* property.

**Default Value**: None.

### B.22.2.4 upnp:objectLinkRef@return

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The upnp:objectLinkRef@return property provides a boolean value. A value of “1” (true) indicates that the current object ID and upnp:objectLinkRef@groupID values represent a pre-selected return point indicated by the content provider.

**Default Value**: “0”

### B.22.2.5 upnp:objectLinkRef::title

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The upnp:objectLinkRef::title property provides a displayable title for the list of objects referred to by this upnp:objectLinkRef property. If this property is present then the list title provided takes precedence over the upnp:objectLink::title property of the target list.

**Default Value**: None.

### B.22.2.6 upnp:objectLinkRef::startObject

**Namespace**: upnp **Property Data Type**: xsd:boolean **Multi-Valued**: NO

**Description**: The upnp:objectLinkRef::startObject property indicates that this object describes a hierarchy of object link lists. If the value of this property is (“1”) then this item SHOULD be used for representing a collection of object link lists corresponding to a complete work. If this item is selected for playback, then the list item identified by the corresponding upnp:objectLinkRef@targetGroupID and upnp:objectLinkRef@targetObjID properties SHOULD be played.

**Default Value**: “0”

### B.22.2.7 upnp:objectLinkRef::relatedInfo

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: YES

**Description**: The upnp:objectLinkRef::relatedInfo property provides information about the relationship of objects on the target list. The property value provides the subject of the relationship. If this property is present in a upnp:objectLinkRef child property the relatedInfo property and its dependent properties takes precedence over the upnp:objectLink::relatedInfo property and all of its dependent properties specified by the target list. See upnp:objectLink::relatedInfo for the definitions of the upnp:objectLinkRef::relatedInfo property and its dependent properties.

**Default Value**: None.

#### B.22.2.7.1 upnp:objectLinkRef::relatedInfo@role

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The REQUIRED upnp:objectLinkRef::relatedInfo@role property provides the type of relationship. The following table provides the allowed values for this property.

**Default Value**: None.

**Table B-36: allowedValueList for the upnp:objectLinkRef::relatedInfo@role Property**

| Allowed Value | Description |
|---|---|
| Actor | List objects refer to the indicated actor |
| Scene | List objects are related to this scene |
| Subject | List objects are related to this subject |
| Vendor defined | List of object are related in a vendor-specified way |

#### B.22.2.7.2 upnp:objectLinkRef::relatedInfo@roleText

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The REQUIRED upnp:objectLinkRef::relatedInfo@roleText property provides the text value of the relationship identified by the upnp:objectLinkRef::relatedInfo@role property.

**Default Value**: None.

## B.23 Foreign Metadata-related Properties

**Table B-37: Foreign Metadata-related Properties Overview**

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| upnp:foreignMetadata | upnp | <XML> | YES | Appendix B.23.1 |
| upnp:foreignMetadata@type | upnp | xsd:string | NO | Appendix B.23.1.1 |
| upnp:foreignMetadata::fmId | upnp | xsd:string | YES | Appendix B.23.1.2 |
| upnp:foreignMetadata::fmClass | upnp | xsd:string | YES | Appendix B.23.1.3 |
| upnp:foreignMetadata::fmProvider | upnp | xsd:string | NO | Appendix B.23.1.4 |
| upnp:foreignMetadata::fmBody | upnp | <XML> | NO | Appendix B.23.1.5 |
| upnp:foreignMetadata::fmBody @xmlFlag | upnp | xsd:boolean | NO | Appendix B.23.1.5.1 |
| upnp:foreignMetadata::fmBody @mimeType | upnp | xsd:string | NO | Appendix B.23.1.5.2 |
| upnp:foreignMetadata::fmBody ::fmEmbeddedXML | upnp | <XML> | NO | Appendix B.23.1.5.3 |
| upnp:foreignMetadata::fmBody ::fmEmbeddedString | upnp | xsd:string | NO | Appendix B.23.1.5.4 |
| upnp:foreignMetadata::fmBody ::fmURI | upnp | xsd:anyURI | NO | Appendix B.23.1.5.5 |

### B.23.1 upnp:foreignMetadata

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: YES

**Description**: The upnp:foreignMetadata property is used to expose additional metadata for the object. The definition and format of the foreign metadata are defined by a third-party organization. In addition, the upnp:foreignMetadata property also contains a number of other properties which identify various information about the foreign metadata such as its format, the organization that defined that format, the object's type or class designation(s) as defined by the external organization, etc. The presence of the upnp:foreignMetadata property allows control points that are able to parse and interpret the foreign metadata to provide additional information about the object to the end-user. Control points that are not able to parse the foreign metadata may safely ignore it.

**Default Value**: None.

#### B.23.1.1 upnp:foreignMetadata@type

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The required *upnp:foreignMetadata@type* property defines the type (and, hence, the format) of the foreign metadata that is contained within the *upnp:foreignMetadata::fmBody* property. The value stored in the *upnp:foreignMetadata@type* property is defined by the organization that owns the metadata format. However, the value MUST conform to the following layout:

**<ICANN registered domain> "_" <typeID>**

where

**<ICANN registered domain>** is the registered name of the organization that owns the metadata format

and

**<typeID>** is a unique ID defined by the organization and uniquely identifies the specific metadata format.

**Example**: "ce.org_MetadataLayout1".

**Default Value**: N/A – This property is REQUIRED when the *upnp:foreignMetadata* property is present.

### B.23.1.2 *upnp:foreignMetadata::fmId*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *YES*

**Description**: The required *upnp:foreignMetadata::fmId* property is used to identify the object using the identification scheme that has been defined by the organization that owns the foreign metadata definition. This property allows the object to be identified using an alternate identification scheme that is different from the native ContentDirectory service identification scheme (i.e. the *upnp:@id* property). The format and allowed values of the *upnp:foreignMetadata::fmId* property are defined by the organization that owns the foreign metadata definition. If the organization has not defined an object identification scheme, then the value MUST be the empty string.

**Example**: "Event-0192837"

**Default Value**: N/A – The property is REQUIRED within every instance of the *upnp:foreignMetadata* property.

### B.23.1.3 *upnp:foreignMetadata::fmClass*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *YES*

**Description**: The required *upnp:foreignMetadata::fmClass* property is used to identify the class or type of this object using the class or type naming scheme that is defined by the organization that owns the foreign metadata definition. The property allows the object's type to be identified using an alternate class naming scheme that is different from the native ContentDirectory service class naming scheme (i.e. the *upnp:class* property). The *upnp:foreignMetadata::fmClass* property is distinct from the *upnp:foreignMetadata@type* property in that the *upnp:foreignMetadata@type* property identifies the format of the foreign metadata within this object whereas the *upnp:foreignMetadata::fmClass* property identifies the object's type. The allowed values of the *upnp:foreignMetadata::fmClass* property are defined by the organization that owns the foreign metadata definition. If the organization has not defined a class scheme, then the value MUST be the empty string.

**Example**: "Broadcast Network"

**Default Value**: N/A – The property is REQUIRED and MUST appear at least once within every instance of the *upnp:foreignMetadata* property.

### B.23.1.4 *upnp:foreignMetadata::fmProvider*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The required *upnp:foreignMetadata::fmProvider* property is used to identify the organization that provided the metadata values contained within the *upnp:foreignMetadata::fmBody* property. The foreign metadata provider should not be confused with the organization that owns the definition of the foreign metadata type. The owner of the foreign metadata type defines the format of the foreign metadata (for instance, its XML schema) whereas the foreign metadata provider assigns the actual values that exist within a given instance of the foreign metadata. The value of the *upnp:foreignMetadata:fmProvider* property MUST be the ICANN registered domain name of the provider. If the provider is not known, then the value MUST be set to the empty string.

**Example**: “tribune.com”

**Default Value**: N/A – The property is REQUIRED within every instance of the *upnp:foreignMetadata* property.

### B.23.1.5 *upnp:foreignMetadata::fmBody*

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: *NO*

**Description**: The required *upnp:foreignMetadata:fmBody* property provides access to the foreign metadata for this object. Access to the foreign metadata is achieved either directly or indirectly. Direct access means that the foreign metadata is embedded directly in a sub-element of the *upnp:foreignMetadata::fmBody* property. Indirect access means that a sub-element of the *upnp:foreignMetadata::fmBody* property identifies how to retrieve the foreign metadata. For example, the sub-element contains a URI. The foreign metadata referenced by the *upnp:foreignMetadata::fmBody* property MUST conform to the foreign metadata type identified by the *upnp:foreignMetadata@type* property and MUST be consistent with the value(s) of the *upnp:foreignMetadata::fmClass* property(ies).

**Default Value**: N/A – The property is REQUIRED within every instance of the *upnp:foreignMetadata* property.

#### B.23.1.5.1 *upnp:foreignMetadata::fmBody@xmlFlag*

**Namespace**: upnp **Property Data Type**: xsd:boolean **Multi-Valued**: *NO*

**Description**: The required *upnp:foreignMetadata::fmBody@xmlFlag* property indicates whether or not the contents of the *upnp:foreignMetadata::fmBody* property are well-formed XML. A value of “1” indicates that the contents are well-formed XML and a value of “0” indicates that the contents are not well-formed XML such as plain text.

Note: The *upnp:foreignMetadata::fmBody@xmlFlag* property takes precedence over the *upnp:foreignMetadata::fmBody@mimeType* property. Consequently, if there is ever a conflict between those two properties, the *upnp:foreignMetadata::fmBody@xmlFlag* property MUST be used as the definitive indicator regarding the presence of well-formed XML within the *upnp:foreignMetadata:fmBody* property.

**Example1:**

```
<upnp:foreignMetadata type="acme.org_MD1">
   <upnp:fmId></upnp:fmId>
   <upnp:fmClass></upnp:fmClass>
   <upnp:fmProvider>acme.org</upnp:fmProvider>
   <upnp:fmBody xmlFlag="0" mimeType="text/plain">
      <upnp:fmEmbeddedString>
         cost=$2.99,purchaseURI=http://www.acme.org/buynow/default.asp
      </upnp:fmEmbeddedString>
   </upnp:fmBody>
</upnp:foreignMetadata>
```

**Example2:**

```
<upnp:foreignMetadata type="acme.org_MD2">
   <upnp:fmId></upnp:fmId>
   <upnp:fmClass></upnp:fmClass>
   <upnp:fmProvider>acme.org</upnp:fmProvider>
   <upnp:fmBody xmlFlag="1" mimeType="text/xml">
      <upnp:fmEmbeddedXML>
         <objectData xmlns="urn:AcmeEpgData"
          xsi:schemaLocation="urn:someepg:\someEPG_schema.xsd">
            <cost>$2.99</cost>
            <purchaseURI>
               http://www.acme.org/buynow/default.asp
            </purchaseURI>
         </objectData>
      </upnp:fmEmbeddedXML>
   </upnp:fmBody>
</upnp:foreignMetadata>
```

**Default Value**:N/A – The property is REQUIRED.

### B.23.1.5.2 upnp:foreignMetadata::fmBody@mimeType

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: NO

**Description**: The upnp:foreignMetadata::fmBody@mimeType property identifies the MIME Type of the contents stored within the upnp:foreignMetadata::fmBody property. The value MUST comply with the MIME specification [RFC 2045]. Although this property is optional, if the MIME Type of the embedded foreign metadata is known, the presence of this property is strongly recommended.

Note: The upnp:foreignMetadata::fmBody@xmlFlag property takes precedence over the upnp:foreignMetadata::fmBody@mimeType property. Consequently, if there is ever a conflict between those two properties, the upnp:foreignMetadata::fmBody@xmlFlag property should be used as the definitive indicator regarding the presence of well-formed XML within the upnp:foreignMetadata::fmBody property.

**Example1:**

```
<upnp:foreignMetadata type="acme.org_MD1">
   <upnp:fmId></upnp:fmId>
   <upnp:fmClass></upnp:fmClass>
   <upnp:fmProvider>acme.org</upnp:fmProvider>
   <upnp:fmBody xmlFlag="0" mimeType="text/plain">
      <upnp:fmEmbeddedString>
         cost=$2.99,purchaseURI=http://www.acme.org/buynow/default.asp
      </upnp:fmEmbeddedString>
   </upnp:fmBody>
</upnp:foreignMetadata>
```

**Example2:**

```
<upnp:foreignMetadata type="acme.org_MD2">
   <upnp:fmId></upnp:fmId>
   <upnp:fmClass></upnp:fmClass>
   <upnp:fmProvider>acme.org</upnp:fmProvider>
   <upnp:fmBody xmlFlag="1" mimeType="text/xml">
      <upnp:fmEmbeddedXML>
         <objectData xmlns="urn:AcmeEpgData"
          xsi:schemaLocation="urn:someepg:\someEPG_schema.xsd">
            <cost>$2.99</cost>
            <purchaseURI>
               http://www.acme.org/buynow/default.asp
            </purchaseURI>
         </objectData>
      </upnp:fmEmbeddedXML>
   </upnp:fmBody>
</upnp:foreignMetadata>
```

**Default Value**: None.

### B.23.1.5.3 *upnp:foreignMetadata::fmBody::fmEmbeddedString*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: The *upnp:foreignMetadata::fmBody::fmEmbeddedString* property contains the actual foreign metadata values associated with this object. The contents of this property MUST be a properly escaped string. The contents of this property MUST conform to the foreign metadata type identified by the *upnp:foreignMetadata@type* property and MUST be consistent with the value(s) of the *upnp:foreignMetadata::fmClass* property(ies), if present. The presence of the *upnp:foreignMetadata::fmBody::fmEmbeddedString* property is mutually exclusive with the *upnp:foreignMetadata::fmBody::fmURI* and *upnp:foreignMetadata::fmBody::fmEmbeddedXML* properties.

**Example:**

In this example, the foreign metadata is embedded as a string whose format is defined by the "acme" company to consist of three comma-separated fields: cost, key, and purchaseURI.

Note: The contents of the "key" field represents the value "abc<def>ghi" but is shown properly escaped according to XML escaping rules.

```
<upnp:foreignMetadata type="acme.org_MD1">
   <upnp:fmId></upnp:fmId>
   <upnp:fmClass></upnp:fmClass>
   <upnp:fmProvider>acme.org</upnp:fmProvider>
   <upnp:fmBody xmlFlag="0" mimeType="text/plain">
      <upnp:fmEmbeddedString>
         cost=$2.99,key=abc&gt;def&lt;ghi,
         purchaseURI=http://www.buynow.com/default.asp
      </upnp:fmEmbeddedString>
   </upnp:fmBody>
</upnp:foreignMetadata>
```

**Default Value**: None.

### B.23.1.5.4 *upnp:foreignMetadata::fmBody::fmEmbeddedXML*

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: *NO*

**Description**: The *upnp:foreignMetadata::fmBody::fmEmbeddedXML* property contains the actual foreign metadata values associated with this object. The contents of this property MUST be a valid, but "headerless" XML document that constitutes the foreign metadata. The term "headerless" means that the embedded XML MUST NOT contain any XML headers and/or directives. Consequently, the embedded foreign metadata MUST be the same version of XML and have the same encoding as the outermost DIDL-Lite XML document. Additionally, the contents MUST conform to the foreign metadata type identified by the *upnp:foreignMetadata@type* property and MUST be consistent with the value(s) of the *upnp:foreignMetadata::fmClass* property(ies), if present. If both of these requirements can not be met (for example, an XML document that requires an XML header <?xml…>), then one of the other *upnp:foreignMetadata::fmBody* sub-elements MUST be used. The presence of the *upnp:foreignMetadata::fmBody::fmEmbeddedXML* property is mutually exclusive with the *upnp:foreignMetadata::fmBody::fmURI* and *upnp:foreignMetadata::fmBody::fmEmbeddedString* properties.

**Example:**

```
<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>2006-01-06T23:59:59-08:00</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>
```

**Default Value**: None.

### B.23.1.5.5 upnp:foreignMetadata::fmBody::fmURI

**Namespace**: upnp **Property Data Type**: xsd:anyURI **Multi-Valued**: YES

**Description**: The upnp:foreignMetadata::fmBody::fmURI property contains a URI reference to the foreign metadata associated with this object. The foreign metadata is retrieved by dereferencing the URI contained in this property. The foreign metadata referenced by this URI MUST conform to the foreign metadata type identified by the upnp:foreignMetadata@type property and MUST be consistent with the value(s) of the upnp:foreignMetadata::fmClass property(ies), if present. Multiple instances of the upnp:foreignMetadata::fmBody::fmURI property indicate that the actual foreign metadata is accessible from multiple URI locations (e.g. multiple servers). When multiple instances of the upnp:foreignMetadata::fmBody::fmURI property exist, all of the specified URIs MUST produce the same (identical) data. The presence of the upnp:foreignMetadata::fmBody:fmURI property is mutually exclusive with the upnp:foreignMetadata::fmBody::fmEmbeddedXML and upnp:foreignMetadata::fmBody::fmEmbeddedString properties.

**Example**:

```
<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:fmURI>http://192.168.1.100/obj123/metadata.xml</upnp:fmURI>
   </upnp:fmBody>
</upnp:foreignMetadata>
```

Where the URI “http://192.168.1.100/obj123/metadata.xml” contains:

```
<?xml version="1.0" encoding="UTF-8"?>
<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>2006-01-06T23:59:59-08:00</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 - January 6th 2006
      </ShortDescription>
   </Content>
</OpenEpg>
```

**Default Value**: None.

## B.24 Synchronized Playback-related Properties

### Table B-38: Synchronized Playback Properties Overview

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:resExt::clockSync* | upnp | <XML> | *NO* | Appendix B.24.1 |
| *upnp:resExt::clockSync @deviceClockInfoID* | upnp | xsd:string | *NO* | Appendix B.24.1.1 |
| *upnp:resExt::clockSync @supportedTimestampsID* | upnp | xsd:string | *NO* | Appendix B.24.1.2 |

### B.24.1 *upnp:resExt::clockSync*

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: *NO*

**Description**: This property contains the clock synchronization information associated with the content-binary identified in the corresponding res property. This data identifies how the device will timestamp the associated content-binary when it is send to the network using the transfer protocol and media format indicated by the *upnp:resExt::clockSync@supportedTimestampsID* property.

**Default Value**: None.

#### B.24.1.1 *upnp:resExt::clockSync@deviceClockInfoID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This REQUIRED property identifies the timestamp mechanism that will be used when the associated content-binary is streamed to the network by the device. Its value MUST equal the value of the `@id` attribute from one of the `<deviceClockInfo>` elements listed in the *Features* data structure contained in the FeatureList state variable of ConnectionManager service [CM].

**Default Value**: N/A. This property is REQUIRED if the *upnp:resExt::clockSync* property is present.

#### B.24.1.2 *upnp:resExt::clockSync@supportedTimestampsID*

**Namespace**: upnp **Property Data Type**: xsd:string **Multi-Valued**: *NO*

**Description**: This REQUIRED property (in conjunction with the *upnp:resExt::clockSync @deviceClockInfoID* property) identifies the timestamp mechanism that will be used when the associated content-binary is streamed to the network by the device. Its value MUST equal the value of the `@id` attribute from one of the `<supportedTimestamps>` elements listed in the *Features* data structure contained in the *FeatureList* state variable of ConnectionManager service.

**Default Value**: N/A. This property is REQUIRED if the *upnp:resExt::clockSync* property is present.

## B.25 DRMInfo-related Overview Properties

### Table B-39: DRMInfo-Related Properties Overview

| Property Name | NS | Data Type | M-Val | Reference |
|---|---|---|---|---|
| *upnp:resExt::DRMInfo* | upnp | <XML> | *YES* | Appendix B.25.1 |
| *upnp:resExt::DRMInfo::foreignMetadata* | upnp | <XML> | *NO* | Appendix B.23.1 |
| *upnp:resExt::DRMInfo::foreignMetadata@type* | upnp | xsd:string | *NO* | Appendix B.23.1.1 |
| *upnp:resExt::DRMInfo::foreignMetadata::fmId* | upnp | xsd:string | *YES* | Appendix B.23.1.2 |
| *upnp:resExt::DRMInfo::foreignMetadata::fmClass* | upnp | xsd:string | *YES* | Appendix B.23.1.3 |
| *upnp:resExt::DRMInfo::foreignMetadata::fmProvider* | upnp | xsd:string | *NO* | Appendix B.23.1.4 |
| *upnp:resExt::DRMInfo::foreignMetadata::fmBody* | upnp | <XML> | *NO* | Appendix B.23.1.5 |
| *upnp:resExt::DRMInfo::foreignMetadata::fmBody@xmlFlag* | upnp | xsd:boolean | *NO* | Appendix B.23.1.5.1 |
| *upnp:resExt::DRMInfo::foreignMetadata::fmBody@mimeType* | upnp | xsd:string | *NO* | Appendix B.23.1.5.2 |
| *upnp:resExt::DRMInfo::foreignMetadata::fmBody::fmEmbeddedXML* | upnp | <XML> | *NO* | Appendix B.23.1.5.4 |
| *upnp:resExt::DRMInfo::foreignMetadata::fmBody::fmEmbeddedString* | upnp | xsd:string | *NO* | Appendix B.23.1.5.3 |
| *upnp:resExt::DRMInfo::foreignMetadata::fmBody::fmURI* | upnp | xsd:anyURI | *NO* | Appendix B.23.1.5.5 |

### B.25.1 *upnp:resExt::DRMInfo*

**Namespace**: upnp **Property Data Type**: <XML> **Multi-Valued**: *YES*

**Description**: This property MUST contain the *upnp:foreignMetadata* information used to specify DRM related information associated with the content-binary identified in the corresponding *res* property.

See Section B.23, “Foreign Metadata-related Properties” for child properties of the *upnp:resExt::DRMInfo* property.

**Default Value**: None.

# Appendix C. AV Working Committee Class Definitions

## C.1 Class Hierarchy

The ContentDirectory service exposes a class hierarchy which is used to type all objects that can be retrieved from it. Each class is named using a string of the form described in Appendix C.1.1, “Class name syntax” below.

For each class, some properties are REQUIRED, others are OPTIONAL and some are PROHIBITED.

A class that is derived from another class MUST include all of the member properties of the parent class. The definition of a derived class MAY make some optional properties of the base class REQUIRED.

Each class definition includes a list of properties. Each property is expressed in XML as either an XML Element or an XML Attribute. Some independent properties are multi-valued for a class, meaning that, in an XML instance of the class, the property may occur more than once.

(Note that the set of properties that MUST be returned in the *Result* argument of the *Browse()* and *Search()* actions are only governed by the DIDL-Lite Schema [DIDL-LITE-XSD] requirements and not by any additional requirements, imposed by the class definitions.)

The support level for a dependent property varies based on the support level of its independent property. If the independent property does not exist, the dependent property is PROHIBITED. If the independent property is REQUIRED or OPTIONAL, its associated dependent properties can be either REQUIRED or OPTIONAL. REQUIRED means that the dependent property MUST exist if and only if the independent property exists. OPTIONAL means that the dependent property MAY exist but only if the independent property exists.

This Appendix defines three classes: the base class *object* and its two derived classes *object.item* and *object.container*, which make up the basic hierarchy from which all other classes (UPnP- or vendor-defined) are derived.

In addition to these classes, the AV Working Committee has defined a number of class descriptions that are derived from either the *item* or *container* classes. Figure 4and Figure 5 below show the hierarchy of these AV Working Committee-defined class definitions.

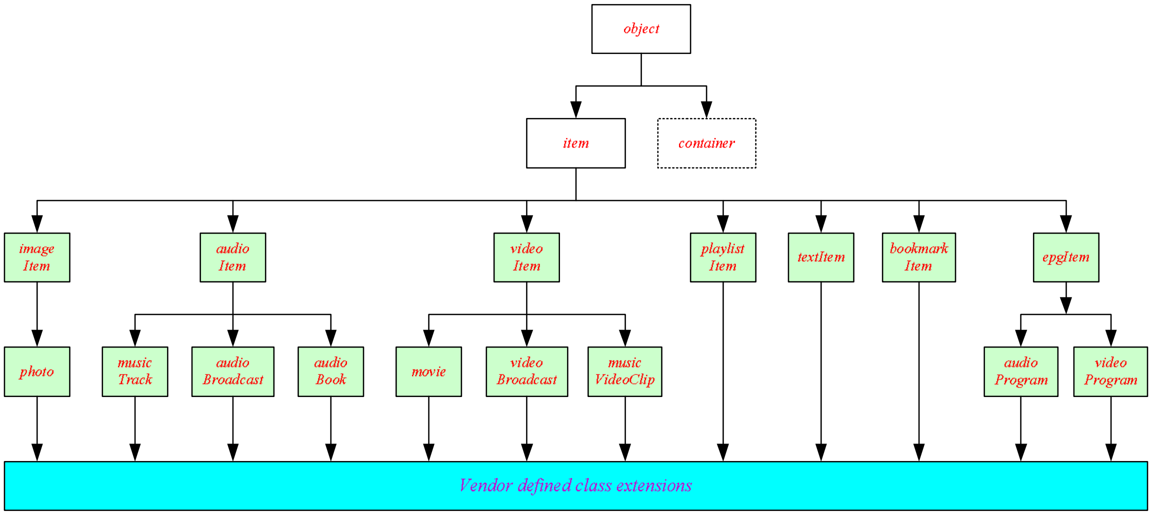


**Figure 4: Class hierarchy for the item base class.**

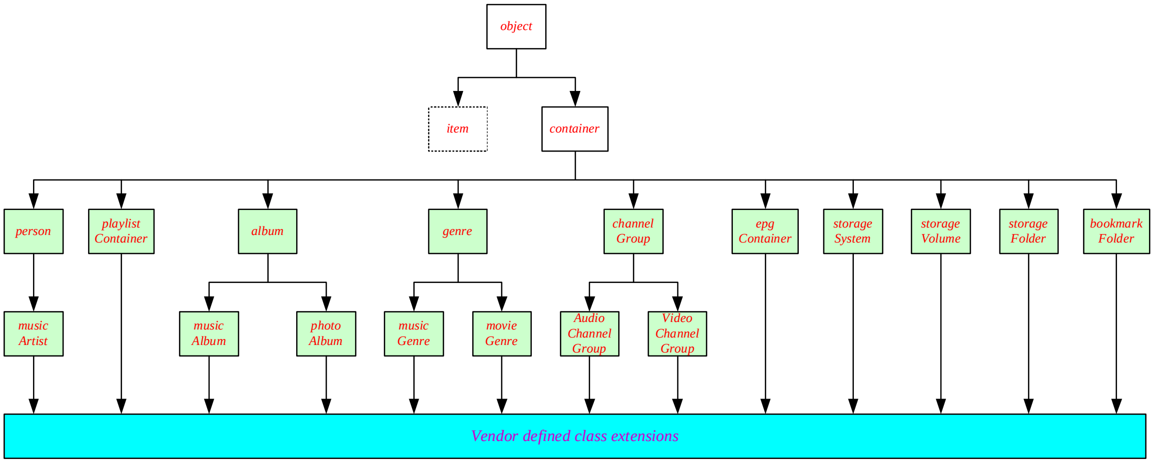


**Figure 5: Class hierarchy for the container base class.**

For each class in these figures, the REQUIRED and OPTIONAL properties that apply to instances of the class are listed. Any device that adds a property whose description matches one of the AV Working Committee-defined property descriptions MUST use the AV Working Committee-defined property name. In addition, any device that uses a property name from the ContentDirectory service specification MUST use it with the same semantics as the AV Working Committee-defined description of that property. ContentDirectory service providers are free to add other properties than those defined in Appendix B, “AV Working Committee Properties” to instances of one of the classes below, from any kind of XML namespace.

## C.1.1 Class name syntax

Class name syntax is formally described using EBNF as described in Section 1.2.3, “Extended Backus-Naur Form”.

```
className   ::= baseName|derivedName

baseName    ::= 'object'

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

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

## C.1.2 Class Properties Overview

The following table presents a complete overview of all the defined properties with an indication in which classes these properties are actually used, either OPTIONAL or REQUIRED. Dependent properties that are marked as REQUIRED are REQUIRED as long as their associated independent property exists. Red cells indicate that the property is PROHIBITED in that class. Blue cells indicate that the property is UNDEFINED for that class. The √ mark indicates that the property's support level is inherited from the parent class. The coloring still indicates the support level.

### Table C-1: Class Properties Overview

| | |
|---|---|
| R | REQUIRED |
| O | OPTIONAL |
| P | PROHIBITED |
| U | UNDEFINED |
| √ | INHERITED |

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkItem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| @id | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| @parentID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| @refID | U | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| @restricted | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| @searchable | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| @childCount | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| dc:title | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| dc:creator | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:class | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:class@name | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:searchClass | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:searchClass@name | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:searchClass@includeDerived | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:createClass | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

R REQUIRED
O OPTIONAL
P PROHIBITED
U UNDEFINED
√ INHERITED

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkItem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:createClass@name | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:createClass@includeDerived | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:writeStatus | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@id | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@protocolInfo | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@importUri | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@size | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@duration | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@protection | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@bitrate | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@bitsPerSample | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@sampleFrequency | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@nrAudioChannels | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@resolution | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@colorDepth | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@tspec | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@allowedUse | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@validityStart | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@validityEnd | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@remainingTime | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| R REQUIRED<br>O OPTIONAL<br>P PROHIBITED<br>U UNDEFINED<br>√ INHERITED<br>Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkitem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| res@usageInfo | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@rightsInfoURI | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@contentInfoURI | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@recordQuality | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@daylightSaving | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| resExt | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| resExt@id | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:artist | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:artist@role | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:actor | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:actor@role | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:author | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:author@role | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:producer | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:director | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| dc:publisher | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| dc:contributor | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:genre | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:genre@id | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:genre@extended | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:album | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| | |
|---|---|
| 🟩 R | REQUIRED |
| 🟨 O | OPTIONAL |
| 🟥 P | PROHIBITED |
| 🟦 U | UNDEFINED |
| ⬜ √ | INHERITED |

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkItem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:playlist | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:albumArtURI | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:artistDiscographyURI | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:lyricsURI | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| dc:relation | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:storageTotal | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | R | R | √ | √ |
| upnp:storageUsed | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | R | R | R | R |
| upnp:storageFree | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | R | R | √ | √ |
| upnp:storageMaxPartition | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | R | √ | √ | √ |
| upnp:storageMedium | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | R | R | √ | √ |
| dc:description | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:longDescription | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:icon | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:region | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| dc:rights | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| dc:date | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| dc:date@upnp:daylightSaving | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| dc:language | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:playbackCount | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:lastPlaybackTime | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:lastPlaybackTime@daylightSaving | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| | |
|---|---|
| R | REQUIRED |
| O | OPTIONAL |
| P | PROHIBITED |
| U | UNDEFINED |
| √ | INHERITED |

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkitem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:lastPlaybackPosition | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:recordedStartDateTime | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:recordedStartDateTime @daylightSaving | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:recordedDuration | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:recordedDayOfWeek | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:srsRecordScheduleID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:srsRecordTaskID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:recordable | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programTitle | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:seriesTitle | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programID@type | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:seriesID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:seriesID@type | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:channelID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:channelID@type | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:channelID@distriNetworkName | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:channelID@distriNetworkID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:episodeCount | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:episodeNumber | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programCode | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| | |
|---|---|
| R | REQUIRED |
| O | OPTIONAL |
| P | PROHIBITED |
| U | UNDEFINED |
| √ | INHERITED |

| Property Name | object | .item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkitem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:programCode@type | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:rating | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:rating@type | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:channelGroupName | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:channelGroupName@id | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:callSign | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:networkAffiliation | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:serviceProvider | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:price | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:price@currency | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:payPerView | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:epgProviderName | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:dateTimeRange | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:dateTimeRange @daylightSaving | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programPreserved | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programPreserved @startTime | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programPreserved @startTimeDaylightSavings | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programPreserved @endTime | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programPreserved @endTimeDaylightSavings | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

R REQUIRED
O OPTIONAL
P PROHIBITED
U UNDEFINED
√ INHERITED

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkItem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:preservedTimeRange | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:preservedTimeRange @startTime | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:preservedTimeRange @startTimeDaylightSavings | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:preservedTimeRange @endTime | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:preservedTimeRange @endTimeDaylightSavings | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programList | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programList::program | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:programList::program @preserved | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:radioCallSign | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:radioStationID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:radioBand | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:channelNr | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:channelName | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:scheduledStartTime | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:scheduledStartTime@usage | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:scheduledStartTime @daylightSaving | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:scheduledEndTime | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:scheduledEndTime @daylightSaving | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| | |
|---|---|
| 🟩 R | REQUIRED |
| 🟨 O | OPTIONAL |
| 🟥 P | PROHIBITED |
| 🟦 U | UNDEFINED |
| √ | INHERITED |

| Property Name | object | .item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkItem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:scheduledDuration | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:signalStrength | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:signalLocked | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:tuned | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:neverPlayable | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::isSyncAnchor | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup@groupID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup@required | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup::component | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup ::component@componentID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup ::component@supportive | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup ::component@supportID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup::component ::componentClass | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| | |
|---|---|
| R | REQUIRED |
| O | OPTIONAL |
| P | PROHIBITED |
| U | UNDEFINED |
| √ | INHERITED |

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkitem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:resExt::componentInfo ::componentGroup::component ::contentType | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup ::component::contentType @MIMEType | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup ::component::contentType @extendedType | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup ::component::language | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup ::component::compRes | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup::component ::compRes::res | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup::component ::compRes::res@protocolInfo | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup::component ::compRes::res@importUri | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup::component ::compRes::isSyncAnchor | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup ::component::compRes::refUDN | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| | |
|---|---|
| R | REQUIRED |
| O | OPTIONAL |
| P | PROHIBITED |
| U | UNDEFINED |
| √ | INHERITED |

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkItem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:resExt::componentInfo ::componentGroup::component ::compRes::refObjectID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::componentInfo ::componentGroup::component ::compRes::refResID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:segmentID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt:segmentInfo | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt:segmentInfo @baseObjectID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt:segmentInfo @baseResID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resEzt::segmentInfo ::timeRange | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt:segmentInfo ::timeRange@start | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt:segmentInfo ::timeRange@end | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resEzt::segmentInfo ::byteRange | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt:segmentInfo ::byteRange@start | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt:segmentInfo ::byteRange@end | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resEzt::segmentInfo ::frameRange | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt:segmentInfo ::frameRange@start | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

R REQUIRED
O OPTIONAL
P PROHIBITED
U UNDEFINED
√ INHERITED

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkitem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:resExt:segmentInfo ::frameRange@end | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:bookmarkID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:bookmarkedObjectID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:deviceUDN | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:deviceUDN@serviceType | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:deviceUDN@serviceID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:stateVariableCollection | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:stateVariableCollection @serviceName | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:stateVariableCollection @rcsInstanceType | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:stateVariableCollection:: stateVariable | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:stateVariableCollection:: stateVariable@variableName | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:DVDRegionCode | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:originalTrackNumber | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:toc | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:userAnnotation | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:containerUpdateID | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectUpdateID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:totalDeletedChildCount | U | P | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| res@updateCount | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| | |
|---|---|
| R | REQUIRED |
| O | OPTIONAL |
| P | PROHIBITED |
| U | UNDEFINED |
| √ | INHERITED |

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkItem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:inclusionControl | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:inclusionControl::role | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectOwner | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectOwner@lock | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectOwner::role | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink@groupID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink@headObjID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink@nextObjID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink@prevObjID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::title | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::startObject | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::mode | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::relatedInfo | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::relatedInfo@role | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::relatedInfo @roleText | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::startInfo | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::startInfo @targetObjID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::startInfo @targetGroupID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink@endAction | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

- 🟩 R REQUIRED
- 🟨 O OPTIONAL
- 🟥 P PROHIBITED
- 🟦 U UNDEFINED
- ⬜ √ INHERITED

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkItem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ...storageVolume | ...storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:objectLink@endAction @action | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::endAction @targetObjID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLink::endAction @targetGroupID | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLinkRef | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLinkRef@groupID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLinkRef@targetObjID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLinkRef@targetGroupID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLinkRef@return | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLinkRef::title | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLinkRef::startObject | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLinkRef::relatedInfo | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLinkRef::relatedInfo @role | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:objectLinkRef::relatedInfo @roleText | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:foreignMetadata | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:foreignMetadata@type | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:foreignMetadata::fmId | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:foreignMetadata::fmClass | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:foreignMetadata::fmProvider | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:foreignMetadata::fmBody | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| | |
|---|---|
| R | REQUIRED |
| O | OPTIONAL |
| P | PROHIBITED |
| U | UNDEFINED |
| √ | INHERITED |

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkItem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp:foreignMetadata::fmBody @xmlFlag | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:foreignMetadata::fmBody @mimeType | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:foreignMetadata::fmBody ::fmEmbeddedString | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:foreignMetadata::fmBody ::fmEmbeddedXML | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:foreignMetadata::fmBody ::fmURI | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::clockSync | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::clockSync @deviceClockInfoID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::clockSync @supportedTimestampsID | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::DRMInfo | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::DRMInfo ::foreignMetadata@type | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::DRMInfo ::foreignMetadata::fmId | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::DRMInfo ::foreignMetadata::fmClass | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt::DRMInfo ::foreignMetadata::fmProvider | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp:resExt:::DRMInfo ::foreignMetadata::fmBody | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp: resExt::DRMInfo ::foreignMetadata::fmBody@xmlFlag | R | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

🟩 R REQUIRED<br>
🟨 O OPTIONAL<br>
🟥 P PROHIBITED<br>
🟦 U UNDEFINED<br>
⬜ √ INHERITED

| Property Name | object | .Item | ..imageItem | ...photo | ..audioItem | ...musicTrack | ...audioBroadcast | ...audioBook | ..videoItem | ...movie | ...videoBradcast | ...musicVideoClip | ..playlistItem | ..textItem | ..bookmarkitem | ..epgItem | ...audioProgram | ...videoProgram | .container | ..person | ...musicArtist | ..playlistContainer | ..album | ...musicAlbum | ...photoAlbum | ..genre | ...musicGenre | ...movieGenre | ..channelGroup | ...audioChannelGroup | ...videoChannelGoup | ..epgContainer | ..storageSystem | ..storageVolume | ..storageFolder | ..bookmarkFolder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| upnp: resExt::DRMInfo ::foreignMetadata::fmBody @mimeType | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp: resExt::DRMInfo ::foreignMetadata::fmBody ::fmEmbeddedString | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp: resExt::DRMInfo ::foreignMetadata::fmBody ::fmEmbeddedXML | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| upnp: resExt::DRMInfo ::foreignMetadata::fmBody::fmURI | O | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

In the following sections, each of the class definitions includes its derivation (parent class) and a properties table. The properties table lists a set of properties for the class and how the properties are used by instances of the class. Each property has a REQUIRED or OPTIONAL entry in the table. An entry of REQUIRED indicates that every instance of that class MUST have a value for that property. An entry of OPTIONAL indicates that every instance of that class is RECOMMENDED to include a value for that property. Each derived class inherits the complete list of properties and their respective REQUIRED/OPTIONAL behaviors from its parent class. Each class may then add more properties to the inherited set by including its own properties list table. A derived class MAY change the behavior of an inherited property from OPTIONAL to REQUIRED, but a derived class MUST NOT change the behavior of a REQUIRED property to OPTIONAL. Unless expressly forbidden, any instance of any class MAY also include a value for any other OPTIONAL property defined in this specification.

## C.2 *object* (Base Class)

This is the root class of the entire ContentDirectory service class hierarchy. It can not be instantiated. No object can be created or otherwise exist in a ContentDirectory service whose *upnp:class* property has the value “*object*”. The *object* class defines properties that are common to both individual media items and logical collections of these items. The *object* class includes the following REQUIRED and OPTIONAL properties:

**Table C-2: *object* Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *@id* | DIDL-Lite | *R* | |
| *@parentID* | DIDL-Lite | *R* | |
| *@restricted* | DIDL-Lite | *R* | |
| *dc:title* | dc | *R* | |
| *upnp:class* | upnp | *R* | |
| *dc:creator* | dc | *O* | |
| *res* | DIDL-Lite | *O* | |
| *upnp:writeStatus* | upnp | *O* | |

### C.2.1 *item:object*

This is a derived class of *object* used to represent *individual* content objects, that is: objects that do not contain other objects; for example, a music track on an audio CD. The XML expression of any instance of a class that is derived from *item* is the `<item>` element. This class is derived from the *object* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-3: *item* Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:objectUpdateID* | upnp | *R/P* | Required if the Track Changes Option (TCO) is supported. Otherwise, prohibited. |
| *res@updateCount* | DIDL-Lite | *R/P* | Required if the Track Changes Option (TCO) is supported. Otherwise, prohibited. |
| *@refID* | DIDL-Lite | *R/P* | REQUIRED for *reference items*, otherwise prohibited. See Section 2.2, “Key Concepts” for details on *reference items.* |
| *upnp:bookmarkID* | upnp | *O* | |

#### C.2.1.1 *imageItem:item*

An *imageItem* instance represents a still image object. It typically has at least one *res* property. This class is derived from the *item* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

Table C-4: *imageItem:item* Properties

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:longDescription* | upnp | *O* | |
| *upnp:storageMedium* | upnp | *O* | |
| *upnp:rating* | upnp | *O* | |
| *dc:description* | dc | *O* | |
| *dc:publisher* | dc | *O* | |
| *dc:date* | dc | *O* | |
| *dc:rights* | dc | *O* | |

#### C.2.1.1.1 *photo:imageItem*

A *photo* instance represents a photo object (as opposed to, for example, an icon). It typically has at least one *res* property. This class is derived from the *imageItem* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

Table C-5: *photo:imageItem* Properties

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:album* | upnp | *O* | |

### C.2.1.2 *audioItem:item*

An *audioItem* instance represents content that is intended for listening. Movies, TV broadcasts, etc., that also contain an audio track are excluded from this definition; those objects are classified under *videoItem*. It typically has at least one *res* property. This class is derived from the *item* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

Table C-6: *audioItem:item* Properties

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:genre* | upnp | *O* | |
| *dc:description* | dc | *O* | |
| *upnp:longDescription* | upnp | *O* | |
| *dc:publisher* | dc | *O* | |
| *dc:language* | dc | *O* | |
| *dc:relation* | dc | *O* | |
| *dc:rights* | dc | *O* | |

#### C.2.1.2.1 *musicTrack:audioItem*

A *musicTrack* instance represents music audio content (as opposed to, for example, a news broadcast or an audio book). It typically has at least one *res* property. This class is derived from the *audioItem* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

Table C-7: musicTrack:audioItem Properties

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:artist | upnp | O | |
| upnp:album | upnp | O | |
| upnp:originalTrackNumber | upnp | O | |
| upnp:playlist | upnp | O | |
| upnp:storageMedium | upnp | O | |
| dc:contributor | dc | O | |
| dc:date | dc | O | |

#### C.2.1.2.2 audioBroadcast:audioItem

An audioBroadcast instance represents a continuous stream from an audio broadcast (as opposed to, for example, a song or an audio book). It typically has at least one res property. This class is derived from the audioItem class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

Table C-8: audioBroadcast:audioItem Properties

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:region | upnp | O | |
| upnp:radioCallSign | upnp | O | |
| upnp:radioStationID | upnp | O | |
| upnp:radioBand | upnp | O | |
| upnp:channelNr | upnp | O | |
| upnp:signalStrength | upnp | O | |
| upnp:signalLocked | upnp | O | |
| upnp:tuned | upnp | O | |
| upnp:recordable | upnp | O | |

#### C.2.1.2.3 audioBook:audioItem

An audioBook instance represents audio content that is the narration of a book (as opposed to, for example, a news broadcast or a song). It typically has at least one res property. This class is derived from the audioItem class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

Table C-9: audioBook:audioItem Properties

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:storageMedium | upnp | O | |
| upnp:producer | upnp | O | |
| dc:contributor | dc | O | |
| dc:date | dc | O | |

### C.2.1.3 videoItem:item

A videoItem instance represents content intended for viewing (as a combination of video and audio). It typically has at least one res property. This class is derived from the item class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-10: videoItem:item Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:genre | upnp | O | |
| upnp:genre@id | upnp | O | |
| upnp:genre@type | upnp | O | |
| upnp:longDescription | upnp | O | |
| upnp:producer | upnp | O | |
| upnp:rating | upnp | O | |
| upnp:actor | upnp | O | |
| upnp:director | upnp | O | |
| dc:description | dc | O | |
| dc:publisher | dc | O | |
| dc:language | dc | O | |
| dc:relation | dc | O | |
| upnp:playbackCount | upnp | O | |
| upnp:lastPlaybackTime | upnp | O | |
| upnp:lastPlaybackPosition | upnp | O | |
| upnp:recordedDayOfWeek | upnp | O | |
| upnp:srsRecordScheduleID | upnp | O | |

#### C.2.1.3.1 movie:videoItem

A movie instance represents content that is a movie (as opposed to, for example, a continuous TV broadcast or a music video clip). It typically has at least one res property. This class is derived from the videoItem class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-11: movie:videoItem Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:storageMedium | upnp | O | |
| upnp:DVDRegionCode | upnp | O | |
| upnp:channelName | upnp | O | |
| upnp:scheduledStartTime | upnp | O | |
| upnp:scheduledEndTime | upnp | O | |

| | | | |
|---|---|---|---|
| upnp:scheduledDuration | upnp | O | |
| upnp:programTitle | upnp | O | |
| upnp:seriesTitle | upnp | O | |
| upnp:episodeCount | upnp | O | |
| upnp:episodeNr | upnp | O | |

#### C.2.1.3.2 videoBroadcast:videoItem

A videoBroadcast instance represents a continuous stream from a video broadcast (for example, a convential TV channel or a Webcast). It typically has at least one res property. This class is derived from the videoItem class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-12: videoBroadcast:videoItem Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:icon | upnp | O | |
| upnp:region | upnp | O | |
| upnp:channelNr | upnp | O | |
| upnp:signalStrength | upnp | O | |
| upnp:signalLocked | upnp | O | |
| upnp:tuned | upnp | O | |
| upnp:recordable | upnp | O | |
| upnp:callSign | upnp | O | |
| upnp:price | upnp | O | |
| upnp:payPerView | upnp | O | |

#### C.2.1.3.3 musicVideoClip:videoItem

A musicVideoClip instance represents video content that is a clip supporting a song (as opposed to, for example, a continuous TV broadcast or a movie). It typically has at least one res property. This class is derived from the videoItem class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-13: musicVideoClip:videoItem Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:artist | upnp | O | |
| upnp:storageMedium | upnp | O | |
| upnp:album | upnp | O | |
| upnp:scheduledStartTime | upnp | O | |
| upnp:scheduledStopTime | upnp | O | |
| upnp:director | upnp | O | |
| dc:contributor | dc | O | |

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| dc:date | dc | O | |

### C.2.1.4 playlistItem:item

A playlistItem instance represents a playable sequence of resources. It is different from musicAlbum in the sense that a playlistItem MAY contain a mix of audio, video and images and is typically created by a user, while an album is typically a fixed published sequence of songs (for example, an audio CD). A playlistItem is REQUIRED to have a res property for playback of the whole sequence. This res property is a reference to a playlist file authored outside of the ContentDirectory service (for example, an external M3U file). Rendering the playlistItem has the semantics defined by the playlist's resource (for example, ordering, transition effects, etc.). This class is derived from the item class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-14: playlistItem:item Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:artist | upnp | O | Applies to the resources inside the playlist. MAY be multi-valued to express multiple artists. |
| upnp:genre | upnp | O | Applies to the playlist as a whole, not any individual resources that it might reference. |
| upnp:longDescription | upnp | O | |
| upnp:storageMedium | upnp | O | Applies to the storageMedium of the playlist file itself, not the resources that the playlist file might reference. |
| dc:description | dc | O | |
| dc:date | dc | O | Applies to the creation date of the playlist file itself, not the resources that it might reference. |
| dc:language | dc | O | Applies to the resources inside the playlist. MAY be multi-valued to express multiple languages. |

### C.2.1.5 textItem:item

A textItem instance represents a content intended for reading. It typically has at least one res property. This class is derived from the item class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-15: textItem:item Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:author | upnp | O | |
| res@protection | upnp | O | |
| upnp:longDescription | upnp | O | |
| upnp:storageMedium | upnp | O | |

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:rating | upnp | O | |
| dc:description | dc | O | |
| dc:publisher | dc | O | |
| dc:contributor | dc | O | |
| dc:date | dc | O | |
| dc:relation | dc | O | |
| dc:language | dc | O | |
| dc:rights | dc | O | |

### C.2.1.6 bookmarkItem:item

A bookmarkItem instance represents a piece of data that can be used to recover previous state information of a AVTransport and a RenderingControl service instance. A bookmarkItem instance can be located in any container but all bookmark items in the ContentDirectory service MUST be accessible within one of the defined bookmark subtrees. This class is derived from the item class and inherits the properties defined by that class. Additionally, the following properties are either REQUIRED or RECOMMENDED for this class:

**Table C-16: bookmarkItem:item Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:bookmarkedObjectID | upnp | R | |
| upnp:neverPlayable | upnp | O | |
| upnp:deviceUDN | upnp | R | |
| upnp:serviceType | upnp | R | |
| upnp:serviceId | upnp | R | |
| dc:date | dc | O | |
| upnp:stateVariableCollection | upnp | R | |

### C.2.1.7 epgItem:item

An epgItem instance represents a program such as a single radio show, a single TV show or a series of programs. This class is derived from the item class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-17: epgItem:item Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:channelGroupName | upnp | O | |
| upnp:channelGroupName@id | upnp | O | |
| upnp:epgProviderName | upnp | O | |
| upnp:serviceProvider | upnp | O | |
| upnp:channelName | upnp | O | |

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:channelNr | upnp | O | |
| upnp:programTitle | upnp | O | |
| upnp:seriesTitle | upnp | O | |
| upnp:programID | upnp | O | |
| upnp:programID@type | upnp | O | |
| upnp:seriesID | upnp | O | |
| upnp:seriesID@type | upnp | O | |
| upnp:channelID | upnp | O | |
| upnp:channelID@type | upnp | O | |
| upnp:channelID @distriNetworkName | upnp | O | |
| upnp:channelID @distriNetworkID | upnp | O | |
| upnp:episodeCount | upnp | O | |
| upnp:episodeNumber | upnp | O | |
| upnp:programCode | upnp | O | |
| upnp:programCode@type | upnp | O | |
| upnp:rating | upnp | O | |
| upnp:rating@type | upnp | O | |
| upnp:episodeType | upnp | O | |
| upnp:genre | upnp | O | |
| upnp:genre@id | upnp | O | |
| upnp:genre@extended | upnp | O | |
| upnp:artist | upnp | O | |
| upnp:artist@role | upnp | O | |
| upnp:actor | upnp | O | |
| upnp:actor@role | upnp | O | |
| upnp:author | upnp | O | |
| upnp:author@role | upnp | O | |
| upnp:producer | upnp | O | |
| upnp:director | upnp | O | |
| dc:publisher | dc | O | |
| dc:contributor | dc | O | |
| upnp:callSign | upnp | O | |
| upnp:networkAffiliation | upnp | O | |

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:price | upnp | O | |
| upnp:price@currency | upnp | O | |
| upnp:payPerView | upnp | O | |
| upnp:epgProviderName | upnp | O | |
| dc:description | dc | O | |
| upnp:longDescription | upnp | O | |
| upnp:icon | upnp | O | |
| upnp:region | upnp | O | |
| upnp:rights | upnp | O | |
| dc:language | dc | O | |
| dc:relation | dc | O | |
| upnp:scheduledStartTime | upnp | O | |
| upnp:scheduledEndTime | upnp | O | |
| upnp:recordable | upnp | O | |
| upnp:foreignMetadata | upnp | O | |

#### C.2.1.7.1 audioProgram:epgItem

An audioProgram instance identifies a single instance of a broadcast audio program such as a radio show or a series of programs. This class is derived from the epgItem class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-18: audioProgram:epgItem Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:radioCallSign | upnp | O | |
| upnp:radioStationID | upnp | O | |
| upnp:radioBand | upnp | O | |

#### C.2.1.7.2 videoProgram:epgItem

A videoProgram instance is a video program such as a single TV show or a series of programs. This class is derived from the epgItem class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-19: videoProgram:epgItem Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:price | upnp | O | |
| upnp:price@currency | upnp | O | |
| upnp:payPerView | upnp | O | |

### C.2.2 container:object

This is a derived class of object used to represent a collection (container) of *individual* content objects and other collections of objects (nested containers). The XML expression of any instance of a class that is derived from container is the `<container>` element. This class is derived from the object class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-20: container Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:objectUpdateID | upnp | R/P | Required if the Track Changes Option (TCO) is supported. Otherwise, prohibited. |
| upnp:containerUpdateID | upnp | R/P | Required if the Track Changes Option (TCO) is supported. Otherwise, prohibited. |
| upnp:totalDeletedChildCount | upnp | R/P | Required if the Track Changes Option (TCO) is supported. Otherwise, prohibited. |
| @childCount | DIDL-Lite | R/O | Required if the Track Changes Option (TCO) is supported. Otherwise, optional. |
| upnp:createClass | upnp | O | |
| upnp:searchClass | upnp | O | |
| @searchable | DIDL-Lite | O | |
| @neverPlayable | DIDL-Lite | O | |

#### C.2.2.1 person:container

A person instance represents an unordered collection of objects associated with a person. It MAY have a res property for playback of all items belonging to the person container. A person container can contain objects of class album, item, or playlist. The classes of objects a person container MAY actually contain is device-dependent. This class is derived from the container class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-21: person:container Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| dc:language | dc | O | |

##### C.2.2.1.1 musicArtist:person

A musicArtist instance is a person instance, where the person associated with the container is a music artist. A musicArtist container can contain objects of class musicAlbum, musicTrack or musicVideoClip. The classes of objects a musicArtist container MAY actually contain is device-dependent. This class is derived from the person class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

Table C-22: *musicArtist:person* Properties

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:genre* | upnp | *O* | |
| *upnp:artistDiscographyURI* | upnp | *O* | |

### C.2.2.2 *playlistContainer:container*

A *playlistContainer* instance represents a collection of objects. It is different from a *musicAlbum* container in the sense that a *playlistContainer* instance MAY contain a mix of audio, video and images and is typically created by a user, while an *album* container typically holds a fixed published sequence of songs (for example, an audio CD). A *playlistContainer* instance MAY have a *res* property for playback of the whole playlist or not. This *res* property MAY be a dynamically created playlist resource, as described in Section 2.6.10.2, "Playlist File Generation", or a reference to a playlist file authored outside of the ContentDirectory service (for example, an external M3U file). This is device-dependent. In any case, rendering the playlist has the semantics defined by the playlist resource (for example, ordering, transition effects, etc.). If the *playlistContainer* instance has no *res* property, a control point needs to separately initiate rendering for each child object, typically in the order the children are received from a *Browse()* action. This class is derived from the *container* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

Table C-23: *playlistContainer:container* Properties

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:artist* | upnp | *O* | |
| *upnp:genre* | upnp | *O* | |
| *upnp:longDescription* | upnp | *O* | |
| *upnp:producer* | upnp | *O* | |
| *upnp:storageMedium* | upnp | *O* | |
| *dc:description* | dc | *O* | |
| *dc:contributor* | dc | *O* | |
| *dc:date* | dc | *O* | |
| *dc:language* | dc | *O* | |
| *dc:rights* | dc | *O* | |

### C.2.2.3 *album:container*

An *album* instance represents an ordered collection of objects. It MAY have a *res* property for playback of the whole *album* instance.When it does, rendering the *album* instance renders all of the objects sequentially. When it does not, a control point needs to separately initiate rendering for each child object. This class is derived from the *container* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

Table C-24: *album:container* Properties

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:storageMedium* | upnp | *O* | |
| *dc:longDescription* | dc | *O* | |

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *dc:description* | dc | *O* | |
| *dc:publisher* | dc | *O* | |
| *dc:contributor* | dc | *O* | |
| *dc:date* | dc | *O* | |
| *dc:relation* | dc | *O* | |
| *dc:rights* | dc | *O* | |

#### C.2.2.3.1 *musicAlbum:album*

A *musicAlbum* instance is an *album* container that contains items of class *musicTrack* (see Appendix C.2.1.2.1, "*musicTrack:audioItem*") or sub-*album* containers of class *musicAlbum*. It can be used to model, for example, an audio-CD. This class is derived from the *album* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-25: *musicAlbum:album* Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:artist* | upnp | *O* | |
| *upnp:genre* | upnp | *O* | |
| *upnp:producer* | upnp | *O* | |
| *upnp:albumArtURI* | upnp | *O* | |
| *upnp:toc* | upnp | *O* | |

#### C.2.2.3.2 *photoAlbum:album*

A *photoAlbum* instance is an *album* container that contains items of class *photo* (see Appendix C.2.1.1.1, "*photo:imageItem*") or sub-*album* containers of class *photoAlbum*. This class is derived from the *album* class and inherits the properties defined by that class. There are no additional RECOMMENDED properties.

**Table C-26: *photoAlbum:album* Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *Intentionally Left Blank* | | | |

### C.2.2.4 *genre:container*

A *genre* instance represents an unordered collection of objects that all belong to the same genre. It MAY have a *res* property for playback of all items of the *genre*, or not. In the first case, rendering the *genre* has the semantics of rendering each object in the collection, in some order. In the latter case, a control point needs to separately initiate rendering for each child object. A *genre* container can contain objects of class *person*, *album*, *audioItem*, *videoItem* or sub-*genre* containers of the same class (for example, Rock contains Alternative Rock). The classes of objects a *genre* container MAY actually contain is device-dependent. This class is derived from the *container* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-27: *genre:container* Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:genre* | upnp | *O* | |
| *upnp:longDescription* | upnp | *O* | |
| *dc:description* | dc | *O* | |

### C.2.2.4.1 *musicGenre:genre*

A *musicGenre* instance is a *genre* which is interpreted as a *style of music*. A *musicGenre* container can contain objects of class *musicArtist*, *musicAlbum*, *audioItem* or sub-*musicgenres* of the same class (for example, Rock contains Alternative Rock). The classes of objects a *musicGenre* container MAY actually contain is device-dependent. This class is derived from the *genre* class and inherits the properties defined by that class.

### C.2.2.4.2 *movieGenre:genre*

A *movieGenre* instance is a *genre* container where the genre indicates a *movie style*. A *movieGenre* container can contain objects of class *people*, *videoItem* or sub-*moviegenres* of the same class (for example, Western contains Spaghetti Western). The classes of objects a *movieGenre* container MAY actually contain is device-dependent. This class is derived from the *genre* class and inherits the properties defined by that class.

## C.2.2.5 *channelGroup:container*

A *channelGroup* container groups together a set of items that correspond to individual but related broadcast channels. For example, all preset channels for a particular tuner may be grouped together in a *channelGroup* container. A device that has multiple tuners may provide multiple *channelGroup* containers, one for each tuner. Alternatively, the device may choose to expose all tuners using just a single *channelGroup* container. This is especially useful when the tuners have equivalent capabilities. Moreover, a device with a single tuner may provide multiple *channelGroup* containers, each exposing only a subset of the available channels (for example, a set-top-box that contains a single tuner but supports three different input connections: terrestrial, cable, and satellite). For UI purposes, control points have the freedom to expose *channelGroup* containers separately, or blend the contents of multiple *channelGroup* containers together in a single view. A *channelGroup* container can only contain objects of class "*object.item.videoItem.videoBroadcast*" or "*object.item.videoItem.audioBroadcast*".

This class is derived from the *container* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-28: *channelGroup:container* Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:channelGroupName* | upnp | *O* | |
| *upnp:channelGroupName@id* | upnp | *O* | |
| *upnp:epgProviderName* | upnp | *O* | |
| *upnp:serviceProvider* | upnp | *O* | |
| *upnp:icon* | upnp | *O* | |
| *upnp:region* | upnp | *O* | |

### C.2.2.5.1 audioChannelGroup:channelGroup

An audioChannelGroup container groups together a set of items that correspond to individual but related audio broadcast channels. An audioChannelGroup container MUST only contain objects of class "object.item.audioItem.audioBroadcast". This class is derived from the channelGroup class and inherits the properties defined by that class.

### C.2.2.5.2 videoChannelGroup:channelGroup

A videoChannelGroup container groups together a set of items that correspond to individual but related video broadcast channels. A videoChannelGroup container MUST only contain objects of class "object.item.videoItem.videoBroadcast". This class is derived from the channelGroup class and inherits the properties defined by that class.

### C.2.2.6 epgContainer:container

An epgContainer instance (EPG container) is a program guide container which MAY contain any kind of objects for EPG information such as audio and video program items or other EPG containers to organize these program items. This class is derived from the container class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-29: epgContainer:container Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:channelGroupName | upnp | O | |
| upnp:channelGroupName@id | upnp | O | |
| upnp:epgProviderName | upnp | O | |
| upnp:serviceProvider | upnp | O | |
| upnp:channelName | upnp | O | |
| upnp:channelNr | upnp | O | |
| upnp:channelID | upnp | O | |
| upnp:channelID@type | upnp | O | |
| upnp:channelID @distriNetworkName | upnp | O | |
| upnp:channelID @distriNetworkID | upnp | O | |
| upnp:radioCallSign | upnp | O | |
| upnp:radioStationID | upnp | O | |
| upnp:radioBand | upnp | O | |
| upnp:callSign | upnp | O | |
| upnp:networkAffiliation | upnp | O | |
| upnp:serviceProvider | upnp | O | |
| upnp:price | upnp | O | |
| upnp:price@currency | upnp | O | |
| upnp:payPerView | upnp | O | |

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:epgProviderName | upnp | O | |
| upnp:icon | upnp | O | |
| upnp:region | upnp | O | |
| dc:language | dc | O | |
| dc:relation | dc | O | |
| upnp:dateTimeRange | upnp | O | |

### C.2.2.7 storageSystem:container

A storageSystem instance represents a potentially heterogeneous collection of storage media. A storageSystem MAY contain other objects, including storageSystem containers, storageVolume containers or storageFolder containers. A storageSystem MUST either be a child of the root container or a child of another storageSystem container. Examples of storageSystem instances are

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

This class is derived from the container class and inherits the properties defined by that class. Additionally, the following REQUIRED properties are defined for this class:

**Table C-30: storageSystem:container Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:storageTotal | upnp | R | |
| upnp:storageUsed | upnp | R | |
| upnp:storageFree | upnp | R | |
| upnp:storageMaxPartition | upnp | R | |
| upnp:storageMedium | upnp | R | |

Regarding the upnp:writeStatus property of a storageSystem container (see object class definition), if there are content items/containers in a storageSystem container that are not contained within any storageVolume container, then all of these *free* items are considered to be contained in a single virtual storageVolume container. For purposes of establishing the upnp:writeStatus property of a storageSystem container, this virtual volume is treated like all the other *real* storageVolumes containers in the storageSystem container.

If every storageVolume container in a storageSystem container has the same value for their upnp:writeStatus property, then the value of upnp:writeStatus property for the storageSystem container MUST also be set to that value.

If any two storageVolume containers in a storageSystem container have different values for their upnp:writeStatus property, then the value of upnp:writeStatus property for the storageSystem container MUST be set to “MIXED”.

### C.2.2.8 storageVolume:container

A storageVolume instance represents all, or a partition of, some physical storage unit of a single type (as indicated by the storageMedium property). The storageVolume container MAY be writable, indicating whether new items can be created as children of the storageVolume container. A storageVolume container

MAY contain other objects, except a *storageSystem* container or another *storageVolume* container. A *storageVolume* container MUST either be a child of the root container or a child of a *storageSystem* container. Examples of *storageVolume* instances are

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

This class is derived from the *container* class and inherits the properties defined by that class. Additionally, the following REQUIRED properties are defined for this class:

**Table C-31: *storageVolume:container* Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:storageTotal* | upnp | *R* | |
| *upnp:storageUsed* | upnp | *R* | |
| *upnp:storageFree* | upnp | *R* | |
| *upnp:storageMedium* | upnp | *R* | |

### C.2.2.9 *storageFolder:container*

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

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

This class is derived from the *container* class and inherits the properties defined by that class. Additionally, the following REQUIRED properties are defined for this class:

**Table C-32: *storageFolder:container* Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| *upnp:storageUsed* | upnp | *R* | |

### C.2.2.10 *bookmarkFolder:container*

A *bookmarkFolder* instance represents an unordered collection of objects that either belong to the "*object.item.bookmarkItem*" class and its derived classes or the "*object.container.bookmarkFolder*" class and its derived classes. A *bookmarkFolder* instance MAY appear anywhere in the ContentDirectory hierarchy.

If a bookmark container and its subtree contains bookmark items that will never have normal playable contents, then that bookmark container SHOULD specify the *@neverPlayable* property set to "*1*". See Appendix B.17.1, "*@neverPlayable*".

This class is derived from the *container* class and inherits the properties defined by that class. Additionally, the following OPTIONAL properties are RECOMMENDED for this class:

**Table C-33: genre:container Properties**

| Property Name | NS | R/O | Remarks |
|---|---|---|---|
| upnp:genre | upnp | O | |
| upnp:longDescription | upnp | O | |
| dc:description | dc | O | |

# Appendix D. EBNF Syntax Definitions

The following sections define the syntax used for some of the properties and classes described in the previous sections. The syntax is formally defined using EBNF as described in Section 1.2.3, “Extended Backus-Naur Form”.

## D.1 Date&time Syntax

```
sched-start     ::= date-time                  |
                    day-of-yr-time             |
                    named-day-time             |
                    T-labeled-time             |
                    'NOW'
start-range     ::= (date-time|'NOW') '/' (date-time|'INFINITY')
date-time-range ::= date-time '/' date-time

duration        ::= 'P' [n 'D'] time
duration-long   ::= duration|'INFINITY'
duration-any    ::= duration|'INFINITY'|'ANY'
duration-adj    ::= ('+'|'-') duration
duration-range  ::= duration '/' duration-long

date-time       ::= yyyy '-' mm '-' dd T-labeled-time
day-of-yr-time  ::=          mm '-' dd T-labeled-time
named-day-time  ::=          named-day T-labeled-time

T-labeled-time  ::= 'T' time [zone]
time            ::= HH ':' MM ':' SS
zone            ::= 'Z'|(('+'|'-') HH ':' MM)
                    (* if zone is omitted, local time is assumed *)

month-day       ::= mm '-' dd
named-day       ::= 'MON'|'TUE'|'WED'|'THU'|'FRI'|'SAT'|'SUN'|
                    'MON-FRI'|'MON-SAT'
n               ::= 1*DIGIT (* non-negative integer *)
yyyy            ::= 4DIGIT  (* 0001-9999 *)
mm              ::= 2DIGIT  (* 01-12 *)
dd              ::= 2DIGIT  (* 01-28, 01-29, 01-30, 01-31
                               based on month/year *)
HH              ::= 2DIGIT  (* 00-23 *)
MM              ::= 2DIGIT  (* 00-59 *)
SS              ::= 2DIGIT  (* 00-59 *)
```

# Appendix E. *CDS features*

This appendix defines a set of extended functionalities for the ContentDirectory service, called *CDS features*. These features have additional requirements beyond the general ContentDirectory service mechanisms to ensure interoperability. The requirements are given in this appendix on a per *CDS feature* basis. When an implementation supports a specific *CDS feature*, it MUST support that feature according to the rules in this appendix.

Each *CDS feature* MUST have an integer version number. Later versions – indicated by a larger version number – MUST support the full functionalitiy of all earlier, lower-numbered versions in the same way as the earlier version (that is, MUST be backward compatible).

Each *CDS feature* MAY also REQUIRE that a list of object IDs be included in its support information. This list identifies specific objects in the ContentDirectory service that control points need to know about to make effective use of the feature.

The names, versions, and, if required, list of object IDs for each implementation-supported feature are returned by the *GetFeatureList()* action. The format of the returned information is defined by [AVS-XSD].

**Example**:

```
<?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>bm3,bm5,bm9</objectIDs>
   </Feature>
   <Feature name="EPG" version="1">
      <objectIDs>epg1,epg2</objectIDs>
   </Feature>
</Features>
```

The normative names for the *CDS features* are listed in Table E-1, "*CDS features*". All *CDS features* are OPTIONAL. A vendor MAY use vendor defined feature names. In this case, the vendor-defined *CDS feature* name MUST be prefixed with the vendor's ICANN domain name followed by the underscore "_".

**Example**: company.com_MyFeature.

Table E-1: *CDS features*

| Name | Description |
|---|---|
| *EPG* | Electronic program guide. See Appendix E.1 "Requirements for the *EPG feature*, Version 1". |
| *TUNER* | Tuner information. See Appendix E.2 "Requirements for the *TUNER feature*, Version 1". |
| *BOOKMARK* | Bookmark management. See Appendix E.3 "Requirements for the *BOOKMARK feature*, Version 1". |
| *FOREIGN_METADATA* | Foreign Metadata. See Appendix E.4, "Requirements for the *FOREIGN_METADATA feature*, Version 1". |
| *FFQ* | FreeFormQuery Support Level. See Appendix E.5, "Requirements for the *FFQ feature*, Version 1" |

| Name | Description |
|---|---|
| MULTI_STREAM | Multi-component and multi-stream items. See Appendix E.6, "Requirements for the *MULTI_STREAM* feature, Version 1" |
| SEGMENTATION | Content segmentation management. See Appendix E.7, "Requirements for the *SEGMENTATION feature*, Version 1" |
| DEVICE_MODE | Device Mode Support. See Appendix E.8, "Requirements for the *DEVICE_MODE feature*, Version 1" |
| CLOCKSYNC | Synchronized Playback Support, See Appendix E.9, "Requirements for the *CLOCKSYNC feature*, Version 1" |
| CONTENT_PROTECTION | Content Protection Support. See Appendix E.10, "Requirements for the *CONTENT_PROTECTION feature*, Version 1". |
| *Vendor-defined* | |

## E.1 Requirements for the *EPG feature*, Version 1

The ContentDirectory service that supports the *EPG feature* provides electronic program guide information. It MUST satisfy the following requirements.

An EPG item is an instance of the class *object.item.epgItem* or one of its derived classes. An EPG container is an instance of the *object.container.epgContainer* class or one of its derived classes. An EPG *root* container is an EPG container whose ancestors do not include another EPG container.

A ContentDirectory service that supports the *EPG feature* MUST have one or more EPG root containers. The *@id* property of every EPG root container in the ContentDirectory service MUST appear in the `objectIDs` list in the EPG entry returned by the *GetFeatureList()* action. Every EPG item in the ContentDirectory service MUST be accessible from the subtree of at least one EPG root container. Also, all EPG items that reference channels belonging to a single channel group MUST be accessible from a single common EPG root container. This MUST be true, whether or not the *TUNER feature* is also supported. An item is accessible from a container (sub)tree if either it or a *reference item* that references it is a direct child of any container in the (sub)tree.

An EPG container MUST only contain EPG items (i.e. items of class *object.item.epgItem* or one of its derived classes), references to EPG items, and EPG containers.

Other than the aforementioned accessibility requirement on EPG items, Version 1 of the *EPG feature* does not require any particular structure under the EPG root containers.

Support for the *EPG feature* MUST be indicated by including the following *Features XML fragment* in the *Features XML Document* value of the *FeatureList* state variable. The actual value of the `objectIDs` attribute is determined at run time according to the requirements in the preceding paragraphs of this section:

```
<?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="EPG" version="1">
      <objectIDs>epg1,epg2</objectIDs>
   </Feature>
</Features>
```

The *EPG feature* <`Feature`> element has the following REQUIRED characteristics:

**Table E-2: REQUIRED characteristics of the *EPG feature* element**

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| `version` | R | Attribute of `<Feature>` | xsd:unsigndInt | Indicates the *EPG feature* version. MUST be set to “1” for this version. |
| `objectIDs` | R | Child element of `<Feature>` | CSV (xsd:string) | Contains the object IDs of all the EPG root containers in the ContentDirectory service. |

## E.2 Requirements for the *TUNER feature*, Version 1

The ContentDirectory service that supports the *TUNER feature* provides electronic tuner information. It MUST satisfy the following requirements.

A broadcast item is an instance of one of the classes *object.item.videoItem.videoBroadcast*, *object.item.audioItem.audioBroadcast* or one of their derived classes. A channel group container is an instance of the *object.container.channelGroup* class or one of its derived classes.

A ContentDirectory service that supports the *TUNER feature* MUST have one or more channel group containers. The *@id* property of every channel group container in the ContentDirectory service MUST appear in the `objectIDs` list in the TUNER entry returned by the *GetFeatureList()* action. Every broadcast item in the ContentDirectory service MUST be accessible from the subtree of at least one channel group container.

A channel group container MUST only contain broadcast items (i.e. items of classes *object.item.videoItem.videoBroadcast*, *object.item.audioItem.audioBroadcast* or one of their derived classes), references to broadcast items, and channel group containers.

Support for the *TUNER feature* MUST be indicated by including the following *Features XML fragment* in the *Features XML Document* value of the *FeatureList* state variable. The actual value of the `objectIDs` attribute is determined at run time according to the requirements in the preceding paragraphs of this section.

**Example**:

```
<?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="TUNER" version="1">
      <objectIDs>T1,T2</objectIDs>
   </Feature>
</Features>
```

The *TUNER feature* `<Feature>` element has the following REQUIRED characteristics:

**Table E-3: REQUIRED characteristics of the *TUNER feature* element**

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| `version` | R | Attribute of `<Feature>` | xsd:unsignedInt | Indicates the *TUNER feature* version. MUST be set to “1” for this version. |
| `objectIDs` | R | Child element of `<Feature>` | CSV (xsd:string) | Contains the object IDs of all the channel group containers in the ContentDirectory service. |

## E.3 Requirements for the *BOOKMARK feature*, Version 1

The ContentDirectory service that exposes the *BOOKMARK feature* provides support for bookmark manipulation. If the *BOOKMARK feature* name is exposed, the following requirements MUST be satisfied.

The ContentDirectory service MUST have at least one bookmark container (instances of class "*object.container.bookmarkFolder*" or one of its derived classes). A bookmark root container is defined as a bookmark container whose ancestors do not contain any other bookmark containers. The object ID of every bookmark root container MUST appear in the `objectIDs` child element of the *BOOKMARK feature* element in the *FeatureList* state variable. The container subtree rooted at a bookmark root container is called a bookmark subtree. Bookmark containers can be located anywhere in the ContentDirectory service.

A bookmark container MUST only contain bookmark items (i.e. items of class "*object.item.bookmarkItem*" or one of its derived classes), references to bookmark items, and bookmark containers. All bookmark items in the ContentDirectory service MUST be accessible from a bookmark subtree, either directly as a bookmark item or indirectly as a reference item to a bookmark item.

The version 1 *BOOKMARK feature* does not require a specific subtree structure under the bookmark root containers.

The ContentDirectory service MUST support *CreateObject()* and *DestroyObject()* actions to configure bookmark entries.

**Example**:

```
<?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>bm1,bm2</objectIDs>
   </Feature>
</Features>
```

The *BOOKMARK feature* `<Feature>` element has the following REQUIRED characteristics:

**Table E-4: REQUIRED characteristics of the *BOOKMARK feature* element**

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| `version` | *R* | Attribute of `<Feature>` | xsd:unsigndInt | Indicates the *BOOKMARK feature* version. MUST be set to "*1*" for this version. |
| `objectIDs` | *R* | Child element of `<Feature>` | xsd:string (CSV of string) | Contains the object IDs of all the bookmark root containers in the ContentDirectory service. |

## E.4 Requirements for the *FOREIGN_METADATA feature*, Version 1

A ContentDirectory service that supports the *FOREIGN_METADATA feature* MUST be capable of embellishing some of its objects with additional metadata beyond the defined set of DIDL-Lite and upnp properties. The definition and format of this foreign metadata are defined by a third-party organization. In addition to the actual foreign metadata values, the ContentDirectory service provides a number of upnp properties that identify various information about the foreign metadata such as its format, the organization that defined that format, the object's type or class designation(s) as defined by the external organization, etc. The presence of foreign metadata within an object allows those control points that are able to parse and interpret the foreign metadata to provide additional information about the object to the end-user. Control points that are not able to parse the foreign metadata may safely ignore it.

A ContentDirectory service that supports the *FOREIGN_METADATA feature* MUST support the *upnp:foreignMetadata* property as detailed in Appendix B.23, "Foreign Metadata-related Properties". This requirement does not mean that the *upnp:foreignMetadata* property must appear within each object but it does mean that the implementation MUST be capable of including the *upnp:foreignMetadata* property whenever the ContentDirectory service has access to any foreign metadata for a given object.

**Example**:

```
<?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>
```

The *FOREIGN_METADATA feature* <`Feature`> element has the following REQUIRED characteristics:

**Table E-5: REQUIRED characteristics of the *FOREIGN_METADATA feature* element**

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| `version` | *R* | Attribute of <`Feature`> | xsd:unsignedInt | Identifies the version of the feature that is supported by this implementation. MUST be set to "*1*". |
| `type` | *R* | Child element of <`Feature`> | xsd:string | Contains information about one of the foreign metadata types that is supported by this implementation. |
| `type@id` | *R* | Attribute of <`type`> | xsd:string | Identifies the type of foreign metadata that is supported by this implementation. Refer to the *upnp:foreignMetadata@type* property for details. |
| `type @provider` | *O* | Attribute of <`type`> | xsd:string | Identifies the provider of the foreign metadata values. |

## E.5 Requirements for the *FFQ feature*, Version 1

A ContentDirectory service that supports the *FreeFormQuery()* action MUST support the *FFQ feature*. This feature indicates for which subtrees the ContentDirectory service supports the *FreeFormQuery()* action and also indicates the support level for XQuery requests that can be used to search that subtree. Each subtree for which the *FreeFormQuery()* action is supported MUST be listed in a separate <objectID> element. The <objectID> element MUST contain the value of the *@id* property of that subtree root container. Subtrees MUST NOT overlap. In other words, any specified subtree MUST NOT have an ancestor that is also specified.

The value specified in the level attribute indicates the support level for XQuery requests that can be used to search the indicated subtree. Currently, only support level 0 is defined. This means full compliance with the XQuery 1.0 specification. Future versions of this specification MAY limit or restrict XQuery language features and syntax to accommodate compatibility with Relational Database implementations, such as SQL.

Support for the *FFQ feature* MUST be indicated by including the following *Features XML fragment* in the *Features XML Document* value of the *FeatureList* state variable:

```
<?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="FFQ" version="1">
      <objectID level="0">12</objectID>
      <objectID level="0">15</objectID>
   </Feature>
</Features>
```

The *FFQ feature* <Feature> element has the following REQUIRED characteristics:

**Table E-6: REQUIRED characteristics of the *FFQ feature* element**

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| version | *R* | Attribute of <Feature> | xsd:unsigndInt | Indicates the *FFQ_SUPPORT_LEVEL feature* version. MUST be set to “1” for this version. |
| objectID | *R* | Child element of <Feature> | xsd:string | Contains the object ID value of the root container of the subtree. |
| level | *R* | Attribute of <objectID> | xsd:string | Contains the support level of the *FreeFormQuery()* action for the subtree of which the root container is indicated by the <objectID> element value. |

## E.6 Requirements for the *MULTI_STREAM feature*, Version 1

A ContentDirectory service implementation that supports the *MULTI_STREAM feature* MUST be capable of exposing metadata properties that describe objects supporting multiple media components. More specifically, it MUST support the following properties as detailed in Appendix B.15:

- *upnp:resExt::isSyncAnchor*
- *upnp:resExt::componentInfo* and its child properties.

In addition, if a ContentDirectory service implementation supports creation of objects containing multiple components, that is, objects containing the *upnp:resExt::componentInfo* property and its child properties, then this is indicated by the <componentCreate> element. If the implementation supports creation of such objects, then the CreateObject(), DestroyObject() and UpdateObject() actions MUST be supported.

Support for the *MULTI_STREAM feature* MUST be indicated by including the following *Features XML fragment* in the *Features XML Document* value of the FeatureList state variable:

```
<?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="MULTI_STREAM" version="1">
      <componentCreate compResCreate="1" />
   </Feature>
</Features>
```

The *MULTI_STREAM feature* <Feature> element has the following REQUIRED characteristics:

**Table E-7: REQUIRED characteristics of the *MULTI_STREAM feature* element**

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| version | R | Attribute of <Feature> | xsd:unsigndInt | Indicates the *MULTI_STREAM feature* version. MUST be set to "1" for this version. |
| componentCreate | O | Child element of <Feature> | xsd:string | Indicates that the ContentDirectory service implementation supports creation of multi-component objects. The value of this element MUST be the empty string. |
| compResCreate | R | Attribute of <componentCreate> | xsd:boolean | If set to "1", then this indicates that the ContentDirectory service implementation supports creation of multi-component objects with the upnp:resExt ::componentInfo::componentGroup ::component::compRes property and its child properties, using the CreateObject() and UpdateObject() actions.<br>If set to "0", then the upnp:resExt ::componentInfo::componentGroup ::component::compRes property and its child properties MUST NOT be used in the CreateObject() and UpdateObject() actions. |

## E.7 Requirements for the *SEGMENTATION feature*, Version 1

A ContentDirectory service that exposes the *SEGMENTATION feature* provides support for segment items (as identified by the upnp:resExt::segmentInfo property). If the ContentDirectory service implementation supports creation of segment items, then each <segmentCreate> element identifies a content-binary format supported by the CreateObject() action. If the ContentDirectory service implementation supports segment item creation then the CreateObject() and DestroyObject()actions MUST be supported.

**Example**:

```
<?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="SEGMENTATION" version="1">
      <segmentCreate protocolInfo="http-get:*:video/mpeg:*" />
      <segmentCreate protocolInfo="http-get:*:video/x-ms-wmv:*">
         <additionalInfoRequired>byte</additionalInfoRequired>
         <additionalInfoRequired>frame</additionalInfoRequired>
      </segmentCreate>
      <segmentCreate protocolInfo="http-get:*:audio/mpeg:*" />
   </Feature>
</Features>
```

The *SEGMENTATION feature* `<Feature>` element has the following REQUIRED characteristics:

**Table E-8: REQUIRED characteristics of the *SEGMENTATION feature* element**

<table>
<tr><th>Name</th><th>R/O</th><th>XML Form</th><th>Type</th><th>Description</th></tr>
<tr><td>version</td><td>R</td><td>Attribute of <Feature></td><td>xsd:unsigndInt</td><td>Indicates the SEGMENTATION feature version. MUST be set to “1” for this version.</td></tr>
<tr><td>segmentCreate</td><td>O</td><td>Child element of <Feature></td><td>xsd:string</td><td>Indicates that the ContentDirectory service implementation supports creation of segments from base item res properties with a matching res@protocolInfo property value. When performing this matching operation the 1st and 3rd @protocolInfo fields are to be compared. The handling of contents of 4th field values is implementation specific. However, the value of “*” indicates that the contents of the 4th field are not significant to the implementation.</td></tr>
<tr><td>protocolInfo</td><td>R</td><td>Attribute of <segmentCreate></td><td>xsd:string</td><td>res@protocolInfo attribute value. See UPnP A/V ConnectionManager service [CM], Section 2.5.2.</td></tr>
<tr><td>additionalInfoRequired</td><td>O</td><td>Child element of <segmentCreate></td><td>xsd:string</td><td>Indicates the ContentDirectory service implementation requires information in addition to the upnp:resExt::segmentInfo::timeRange property to create segments for this media type.<br>The value of the <additionalInfoRequired> element indicates which additional upnp:resExt::segmentInfo properties that must be provided:<table><tr><td>Value</td><td>Property</td></tr></table></td></tr>
</table>

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| | | | | frame<br>upnp:resExt::segmentInfo::frameRange<br>byte<br>upnp:resExt::segmentInfo::byteRange<br>If multiple <additionalInfoRequired> elements are present, then any one of the indicated properties may be provided. |

## E.8 Requirements for the *DEVICE_MODE feature*, Version 1

A ContentDirectory service that supports the *DEVICE_MODE feature* MUST be capable of supporting one or more special operating modes.

**Example**:

```
<?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="DEVICE_MODE" version="1">
      <dmType id="ActionBurst" CPRequested="1">
         <actionNameProcessing support="1" enforce="0"/>
      </dmType>
      <dmType id="ExclusiveOwnership" CPRequested="1">
         <resourceID type="Device"></resourceID>
      </dmType>
   </Feature>
</Features>
```

The *DEVICE_MODE feature* <Feature> element has the following REQUIRED characteristics:

**Table E-9: REQUIRED characteristics of the *DEVICE_MODE feature* element**

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| Version | R | Attribute of <Feature> | xsd:unsignedInt | Identifies the version of the feature that is supported by this implementation. MUST be set to "1". |
| dmType | R | Child element of <Feature> | xsd:string | Identifies the special device modes that are supported by this implementation. |

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| dmType@id | R | Attribute of <dmType> | xsd:string | Identifies one of the special device modes that are supported by this implementation. The list of valid values includes:<br>"ActionBurst"<br>"ExclusiveOwnership" |
| dmType@CPrequested | R | Attribute of <dmType> | xsd:boolean | Identifies if the requested Device mode can be set by the control point. The list of valid values:<br>"0\|1" |
| actionNameProcessing | CR | Child element of <dmType> | xsd:string | Identifies how the optional actionName description is supported when the *ActionBurst* mode is applicable. |
| actionNameProcessing@support | R | Attribute of <actionNameProcessing> | xsd:boolean | Identifies if the ContentDirectory service processes the actionName element input on the *RequestDeviceMode()* and *ExtendDeviceMode()* actions. A value of "*1*" indicates that the device uses the actionName element input to optimize the *ActionBurst* mode response. A value of "*0*" indicates that the device ignores this input. |
| actionNameProcessing@enforce | R | Attribute of <actionNameProcessing> | xsd:boolean | Identifies if the ContentDirectory service processes the <actionName> element input on the *RequestDeviceMode()* action *ExtendDeviceMode()* action and expects complimentary action from the control point. A value of "*1*" indicates that the device (ContentDirectory service) expects the control point to keep track of invoked actions against the actionName element input with the expectation that the device may truncate the *ActionBurst* if invoked actions deviate measurably from the anticipated *ActionBurst* envelope, A value of "*1*" MUST NOT be included if the value of the support attribute of the <actionNameProcessing> element is "*0*" since the ContentDirectory service does not process this element. A value of "*0*" indicates that the device ignores this input. |
| resourceID | CR | Child element of <dmType> | xsd:string | Identifies the resource for which the *ExclusiveOwnership* mode is applicable. Note: For a resource type of "*Device*" the <resourceID> element value is empty. |
| resourceID@type | R | Attribute of <resourceID> | xsd:string | Identifies the resource type for which the *ExclusiveOwnership* mode is applicable. The list of valid values includes:<br>"*Device*" |

As described above, the `id` attribute of the `<dmType>` element of the *DEVICE_MODE feature* identifies one of the special device modes that are supported by this implementation. The modes currently defined are *ActionBurst* and *ExclusiveOwnership*. See Section 2.2.26, “Device Mode Option”.

## E.9 Requirements for the *CLOCKSYNC feature*, Version 1

The ContentDirectory service that supports the *CLOCKSYNC feature* provides clock synchronization information and synchronized playback functionality. It MUST satisfy the following requirements.

Support for the *CLOCKSYNC feature* MUST be indicated by including the following *Features XML fragment* in the *Features XML Document* value of the FeatureList state variable:

```
<?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="CLOCKSYNC" version="1">
</Feature>
</Features>
```

The *CLOCKSYNC feature* <`Feature`> element has the following REQUIRED characteristics:

**Table E-10: REQUIRED characteristics of the CLOCKSYNC *Feature* element**

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| version | R | Attribute of `<Feature>` | xsd:unsignedInt | Indicates the *CLOCKSYNC feature* version. MUST be set to “1” for this version. |

## E.10 Requirements for the *CONTENT_PROTECTION feature*, Version 1

A ContentDirectory service implementation that supports the *CONTENT_PROTECTION feature* can provide customized access to ContentDirectory service object metadata according to the identity of a control point or user invoking an action. If the *CONTENT_PROTECTION feature* is exposed then the ContentDirectory service:

- MUST be implemented on a device that also implements the DeviceProtection service [DP]
- MUST be capable of supporting *Action level access* as described in Appendix F.1.1, MUST support the *GetPermissionsInfo()* action and *PermissionInfo* state variable, and MUST modify the behavior of the *CreateObject()*,*UpdateObject()*, *Browse()*, *Search()*, *FreeFormQuery()*, *DestroyObject()*, *MoveObject()*, and *DeleteResource()* actions as described in sections F.2, “Behavior of actions with *Content_Protection feature*”.
- MUST be capable of supporting *Object level access* as described in Appendix F.1.4 and exposing metadata properties that describe this access, more specifically, it MUST support the following properties as detailed in Appendix B.20.1 and B.21.1 respectively:
  - *upnp: inclusionControl* and its child properties.
  - *upnp:objectOwner* and its child properties.

Support for the *CONTENT_PROTECTION feature* MUST be indicated by including the following *Features XML fragment* in the *Features XML Document* value of the *FeatureList* state variable:

```
<?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="CONTENT_PROTECTION" version="1">
   </Feature>
</Features>
```

The *CONTENT_PROTECTION feature* `<Feature>` element has the following REQUIRED characteristics:

**Table E-11: REQUIRED characteristics of the *CONTENT_PROTECTION feature* element**

| Name | R/O | XML Form | Type | Description |
|---|---|---|---|---|
| `version` | *R* | Attribute of `<Feature>` | xsd:unsigndInt | Indicates the *CONTENT_PROTECTION feature* version. MUST be set to “1” for this version. |

# Appendix F. *Content_Protection feature* (Normative)

The *CONTENT_PROTECTION feature* is an extension of the DeviceProtection Service [DP] to the actions (*Action level access*) and object metadata (*Object level access*) of the ContentDirectory service. By defining a set of AV *Roles* and fixed *Action level access* and *Object level access* a consistent experience can be implemented. Additionally, an implementation MAY define other vendor *Roles* with other *Action level access* and *Object level access*.

## F.1 AV *Roles* for Content_Protection

The following table lists pre-defined AV *Roles* for the *CONTENT_PROTECTION feature*. These *Roles* MUST be supported when the *CONTENT_PROTECTION feature* is implemented. This list of pre-defined *Roles* MAY be extended by the implementer with additional vendor specific *Roles*.

**Table F-1: Pre-defined *AV Roles* and *Public***

| Role Name | R/O[1] | Data Type | IncludeAll[2] | OwnAll[3] |
|---|---|---|---|---|
| *Public* | *CR*[4] | **string** | NO | NO |
| *AV:PublicWriter* | *CR*[4] | **string** | NO | NO |
| *AV:Writer* | *CR*[4] | **string** | NO | NO |
| *AV:SuperWriter* | *CR*[4] | **string** | NO | YES |
| *AV:Reader* | *CR*[4] | **string** | NO | NO |
| *AV:SuperReader* | *CR*[4] | **string** | YES | NO |

The DeviceProtection *Public* and new *AV:PublicWriter Roles* are intended to identify objects that may be read (or written) by control point(s) which do not participate in the device protection scheme or whose identity is not recognized by the DeviceProtection service.

The *Public Role* is defined in the DeviceProtection service and is assigned read related action permissions (that is actions that reveal information about ContentDirectory service objects to control points, such as, *Browse()* and *Search()*); see Table F-4 for details. This is the default DeviceProtection service *Role* and therefore default AV *Role*. It MUST be assigned to all control points including any unrecognized control points.

The *AV:PublicWriter Role* is defined by the ContentDirectory service and enables write related action permissions (that is actions that create, modify, or remove information on ContentDirectory service objects

[1] For a device this column indicates whether the action MUST be implemented or not, where *R* = REQUIRED, *O* = OPTIONAL, *CR* = CONDITIONALLY REQUIRED, *CO* = CONDITIONALLY OPTIONAL, *X* = Non-standard, add *-D* when deprecated (e.g., *R-D*, *O-D*).

[2] NO indicates that this *Role* MUST NOT appear in an <includeAll> element of the *PermissionsInfo* state variable; YES indicates that this *Role* MUST appear in an <includeAll> element of the *PermissionsInfo* state variable.

[3] NO indicates that this *Role* MUST NOT appear in an <ownAll> element of the *PermissionsInfo* state variable; YES indicates that this *Role* MUST appear in an <ownAll> element of the *PermissionsInfo* state variable.

[4] CONDITIONALLY REQUIRED if the *CONTENT_PROTECTION feature* is implemented.

at the direction of a control point, such as *CreateObject()*, *UpdateObject()*, *DestroyObject()* and *DeleteResource()*; see Table F-4 for details. The *AV:PublicWriter Role* is considered a complementary *Role* to the *Public Role* as these *Roles* when assigned together to a single control point allow access to all legacy (earlier versions) of ContentDirectory service actions. The DeviceProtection service [DP] SHOULD assign the *AV:PublicWriter Role* to all control points including any unrecognized control points. By having the read related and write related action permissions segmented for unrecognized control points it is possible to allow read related access to all objects while limiting write related access.

The *AV:Reader Role* is defined by the ContentDirectory service and enables read related action permissions. It SHOULD NOT be assigned to unrecognized control points. Assignment of the *AV:Reader Role* to unrecognized control points is PROHIBITED.By having permissions equivalent to the *Public Role* in a separate *Role*, additional partitioning of Content Directory service read related access can be provided between recognized and unrecognized control points.

The *AV:Writer Role* is defined by the ContentDirectory service and enables write related action permissions. The *AV:Writer Role* is similar to the *AV:PublicWriter Role*, however the *AV:Writer Role* MUST only be applied to recognized control points. Assignment of the *AV:Writer Role* to unrecognized control points is PROHIBITED. Similar to *AV:PublicWriter*, the *AV:Writer Role* enables additional partitioning of ContentDirectory service write related actions.

The *AV:SuperReader Role* is defined by the ContentDirectory service and enables read related action permissions on all objects at all times in the ContentDirectory service. This *Role* is exempt from the *upnp:inclusionControl* property restrictions described later in this specification. Assignment of the *AV:SuperReader Role* to unrecognized control points is PROHIBITED.

The *AV:SuperWriter Role* is defined by the ContentDirectory service and enables write related action permissions on all objects at all times in the ContentDirectory service. This *Role* is exempt from the *upnp:objectOwner* property restrictions described later in this specification. Assignment of the *AV:SuperWriter Role* to unrecognized control points is PROHIBITED.

The *Admin Role* is defined by the DeviceProtection service [DP]. The *Admin Role* has no effect with regards to AV actions. However, a control point with the *Admin Role* can add the *AV:SuperReader* and *AV:SuperWriter Roles* to any *Control Point* or *User Identity* enabling this *Identity* to have read/write permissions for all ContentDirectory service objects and actions. This is the RECOMMENDED method for creating a "master" control point for the ContentDirectory service.

It is RECOMMENDED that the *AV:PublicWriter*, *AV:Reader*, *AV:Writer*, *AV:SuperReader*, and *AV:SuperWriter Roles* not have additional permissions to manage the *Roles* and *Identities* of the device as described in the DeviceProtection service [DP].

### F.1.1 Access at action level

If a control point has at least one *Role* that is not restricted from invoking a specific action, then it is said to have *Action level access*, otherwise, the ContentDirectory service implementation MUST issue the error code 606 (see UPnP Device Architecture [DEVICE]) in response to the action invocation.

**Table F-2: Error Codes for *Action Level Access***

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 606 | Action not authorized | Action not authorized: The control point does not have privileges to invoke this action |

### F.1.2 Restrictable and Non-Restrictable Actions

AV actions are defined as *Restrictable* or *Non-Restrictable* (see section 2.2.25.1) when the *CONTENT_PROTECTION feature* is implemented. Table F-3 (below) lists pre-defined setting for *Restrictable* and *Non-Restrictable* ContentDirectory service actions. ContentDirectory service actions defined as *Non-Restrictable* MUST appear in one and only one *PermissionsInfo* state variable `<nonRestrictable>` element. In addition all *Non-Restrictable* actions (supported by an implementation) MUST be returned in the *DeviceProtection::GetRolesforAction()* response for all *Roles*. Actions defined as *Restrictable* MUST NOT appear in any *PermissionsInfo* `<nonRestrictable>` element. As indicated previously, unimplemented AV actions MUST NOT be reported by the DeviceProtection service [DP] *DeviceProtection::GetRolesforAction()* action or in the *PermissionsInfo* state variable.

**Table F-3: Pre-defined settings for *Restrictable and Non-Restrictable* AV Actions**

| AV Action Name | Category |
|---|---|
| *GetSearchCapabilities()* | *Non-Restrictable* |
| *GetSortCapabilities()* | *Non-Restrictable* |
| *GetSortExtensionCapabilities()* | *Non-Restrictable* |
| *GetFeatureList()* | *Non-Restrictable* |
| *GetSystemUpdateID()* | *Non-Restrictable* |
| *GetServiceResetToken()* | *Non-Restrictable* |
| *GetPermissionsInfo()* | *Non-Restrictable* |
| *Browse()* | *Restrictable* |
| *Search()* | *Restrictable* |
| *CreateObject()* | *Restrictable* |
| *DestroyObject()* | *Restrictable* |
| *UpdateObject()* | *Restrictable* |
| *MoveObject()* | *Restrictable* |
| *ImportResource()* | *Restrictable* |
| *ExportResource()* | *Restrictable* |
| *DeleteResource()* | *Restrictable* |
| *StopTransferResource()* | *Restrictable* |
| *GetTransferProgress()* | *Restrictable* |
| *CreateReference()* | *Restrictable* |
| *FreeFormQuery()* | *Restrictable* |
| *GetFreeFormQueryCapabilities()* | *Non-Restrictable* |
| *RequestDeviceMode* | *Restrictable* |
| *ExtendDeviceMode* | *Restrictable* |
| *CancelDeviceMode* | *Restrictable* |
| *GetDeviceModeStatus* | *Restrictable* |

| AV Action Name | Category |
|---|---|
| *GetDeviceMode* | *Restrictable* |

### F.1.3 *Action Level Access* using pre-defined AV *Roles*

The following table (Table F-4) shows AV actions accessible to a *User* or *Control Point Identity* assigned each of the pre-defined AV *Roles*. A *User* or *Control Point Identity* possessing more than one of these roles would be allowed access to any action permitted by any of the assigned *Roles*. A YES value indicates that *Action Level access* MUST be granted by the corresponding *Role*, while a NO value indicates that *Action Level access* MUST NOT granted by this *Role*. Note that a NO value does not explicitly prohibit *Action Level Access*, that is, another *Role* that a *User* or *Control Point Identity* possesses MAY permit *Action Level Access*.

**Table F-4: Pre-defined AV Action to AV *Role* permissions mapping**

| AV Action Name | *Public* | *AV:PublicWriter* | *AV:Reader* | *AV:Writer* | *AV:SuperReader* | *AV:SuperWriter* |
|---|---|---|---|---|---|---|
| *GetSearchCapabilities()* | YES | YES | YES | YES | YES | YES |
| *GetSortCapabilities()* | YES | YES | YES | YES | YES | YES |
| *GetSortExtensionCapabilities()* | YES | YES | YES | YES | YES | YES |
| *GetFeatureList()* | YES | YES | YES | YES | YES | YES |
| *GetSystemUpdateID()* | YES | YES | YES | YES | YES | YES |
| *GetServiceResetToken()* | YES | YES | YES | YES | YES | YES |
| *GetPermissionsInfo()* | YES | YES | YES | YES | YES | YES |
| *Browse()* | YES | NO | YES | NO | YES | NO |
| *Search()* | YES | NO | YES | NO | YES | NO |
| *CreateObject()* | NO | YES | NO | YES | NO | YES |
| *DestroyObject()* | NO | YES | NO | YES | NO | YES |
| *UpdateObject()* | NO | YES | NO | YES | NO | YES |
| *MoveObject()* | YES | YES | YES | YES | YES | YES |
| *ImportResource()* | NO | YES | NO | YES | NO | YES |
| *ExportResource()* | YES | NO | YES | NO | YES | NO |
| *DeleteResource()* | NO | YES | NO | YES | NO | YES |
| *StopTransferResource()* | NO | YES | NO | YES | NO | YES |
| *GetTransferProgress()* | NO | YES | NO | YES | NO | YES |
| *CreateReference()* | NO | YES | NO | YES | NO | YES |
| *FreeFormQuery()* | YES | NO | YES | NO | YES | NO |

| AV Action Name | *Public* | *AV:PublicWriter* | *AV:Reader* | *AV:Writer* | *AV:SuperReader* | *AV:SuperWriter* |
|---|---|---|---|---|---|---|
| *GetFreeFormQueryCapabilities()* | YES | YES | YES | YES | YES | YES |
| *RequestDeviceMode* | NO | NO | YES | YES | YES | YES |
| *ExtendDeviceMode* | NO | NO | YES | YES | YES | YES |
| *CancelDeviceMode* | NO | NO | YES | YES | YES | YES |
| *GetDeviceModeStatus* | NO | NO | YES | YES | YES | YES |
| *GetDeviceMode* | NO | NO | YES | YES | YES | YES |

### F.1.4 Access at object level

Assuming *Action level access* is available to a control point, the control point will also need to have at least one *Role* with *Object level access* to the target objects(s) to get a full response, such as for *Browse()* or *Search()*, or a successful response, such as for *UpdateObject()* or *MoveObject()*, to the action invocation. *Object level access* is determined by the *upnp:inclusionControl* property of an object. If the object is an item and has no *upnp:inclusionControl* property, then its parent container's *upnp:inclusionControl* property is used to determine *Roles* for *Object Level access* to the item (see Appendix B.20.1 for more details). If a *Control Point* or *User Identity* includes a *Role* which matches an applicable item or container *upnp:inclusionControl* property *Role* then the control point is said to have *Object Level access* to the item or container. If the invoking control point has *Action level access* but not *Object Level access* to the target object(s) of an action, then the ContentDirectory service implementation MUST issue the error code 740 in response to the action invocation.

- When an object (item or container) has a *upnp:inclusionControl* property present that *upnp:inclusionControl* property MUST determine *Object level access* for that object.
- When the object is an item and neither it or its parent container have a *upnp:inclusionControl* property present, then all control points with *Action level access* to that item MUST also have *Object level access* for that item.
- When the object is an item and does not have a *upnp:inclusionControl* property present but its parent container does, then the parent container's *upnp:inclusionControl* property MUST determine *Object level access* for that item.
- When the object is a container and does not have a *upnp:inclusionControl* property present but its parent container does, the absence of a *upnp:inclusionControl* property on the child container determines *Object level access,* that is all *Roles* have access to the container. Note, it is therefore possible to have more liberal access to items in a child container than in the parent, that is a child container does not inherit its parent's *Object level access*.

### F.1.5 *Role* assignments for unrecognized control points

According to the DeviceProtection service [DP], unrecognized control points MUST have the *Public Role*. According to the ContentDirectory service, unrecognized control points MAY also have the *AV:PublicWriter Role*. The *AV:Reader*, *AV:Writer*, *AV:SuperReader*, and *AV:SuperWriter Roles* MUST only be assigned to control points with a *Control Point* or *User Identity*. See Appendix B.20.1 for additional requirements related to the *upnp:inclusionControl* property.

### F.1.6 Implicit role assignments

*Roles* listed in a *PermissionsInfo* `<includeAll>` element are implicitly added to the *upnp:inclusionControl* property of all ContentDirectory service objects. The ContentDirectory service REQUIRES that the *AV:SuperReader* *Role* be included in the *PermissionsInfo* `<includeAll>` element. This results in the *AV:SuperReader* Role being a valid *Role* for all ContentDirectory service objects. It is HIGHLY RECOMMENDED that all *Roles* listed in the *PermissionsInfo* `<includeAll>` element be explicitly listed in all *upnp:inclusionControl* properties returned by the ContentDirectory service. The procedure for adding or removing *roles* from the *PermissionsInfo* state variable elements is implementation specific.Modifications to *upnp:inclusionControl* property

*Object Level access* to an object's *upnp:inclusionControl* property is controlled by the *upnp:objectOwner* property of the same object. That is, when the *CONTENT_PROTECTION feature* is supported, the *upnp:objectOwner* property provides a mechanism for controlling *Object level access* to itself and the same object's *upnp:inclusionControl* property. A *Role* indicated by the *upnp:objectOwner::role* property is defined to be an *owner* of that object or said to *own* that object. Conversely, *Roles* that are not included in a *upnp:objectOwner::role* property are by definition *non-owner Roles* of that object.

For example, if an object has the following *upnp:inclusionControl* and *upnp:objectOwner* properties:

```
<upnp:inclusionControl>
    <upnp:role>AV:Reader</upnp:role>
    <upnp:role>AV:Writer</upnp:role>
    <upnp:role>example.com:Child</upnp:role>
    <upnp:role>example.com:ParentReader</upnp:role>
    <upnp:role>example.com:ParentWriter</upnp:role>
</upnp:inclusionControl>

<upnp:objectOwner lock="1">
    <upnp:role>AV:Writer</upnp:role>
    <upnp:role>example.com:ParentWriter</upnp:role>
</upnp:objectOwner>
```

then only a *Control Point Identity* or *User Identity* with at least one of the *Roles* *AV:Reader*, *AV:Writer*, *AV:SuperReader*, *example.com:Child* or *example.com:ParentReader* would be able to browse the object. Only a *Control Point Identity* or *User Identity* with at least one of the *Roles* *AV:Writer*, *AV:SuperWriter*,or *example.com:ParentWriter* would be able to modify or delete the object. And finally, only an *owner* (a *Control Point Identity* or *User Identity* with with at least one of the *Roles* *AV:Writer*, *AV:SuperWriter*, or *example.com:ParentWriter*) would be able to modify the *upnp:objectOwner@lock* property and therefore the *upnp:objectOwner* and *upnp:inclusionControl* properties. See Appendix B.21.1 for additional details.

All *Roles* that have a *PermissionsInfo* `<ownAll>` element MUST be treated as having an entry in the *upnp:objectOwner* property whether explicitly included or not. Therefore the *AV:SuperWriter* *Role* MUST be considered to be a member of the *upnp:objectOwner* property when that property is present. Note, it is highly RECOMMENDED that for each *Role* with an `<ownAll>` element in the *PermissionsInfo* state variable, that the ContentDirectory service add a corresponding *upnp:objectOwner::role* to the object.

The *upnp:objectOwner@lock* property allows a current *owner* of an object to lock the *upnp:inclusionControl* and *upnp:objectOwner* properties to a read-only state. When this lock state is set all control points MUST be PROHIBITED from:

- creating, modifying or deleting the object's *upnp:inclusionControl* property,
- modifying or deleting the object's *upnp:objectOwner* property with the exception that an *owner* control point MAY change the value of the *upnp:objectOwner@lock* property from a value of "*1*"(true) to a value of "*0*"(false).

See *CreateObject()* (section F.2.1) and *UpdateObject()* (section F.2.2) in this appendix for additional details.

**Table F-5: Error Codes for *Object Level Access***

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 740 | Object not authorized | The control point does not have *Role* permissions to invoke this action on at least one of the target objects. |

When a control point attempts to invoke an action on an object that does not have either *Action level access* or *Object level access*, then the ContentDirectory service implementation MUST return the action level error code 606 in Table F-2.

## F.2 Behavior of actions with *Content_Protection feature*

When the *CONTENT_PROTECTION feature* is implemented then the behavior of the *Restrictable* actions (see Section F.1.2) is modified as follows:

### F.2.1 *CreateObject()* action with CONTENT_PROTECTION feature

When the *CONTENT_PROTECTION feature* is implemented, a successful invocation of the *CreateObject()* action (assuming the control point has *Action level access*) MUST be on a container that the control point has *Object level access* to (See Appendix F.1.4 above). In that case, a control point is allowed to create an object with fully populated *upnp:objectOwner* and *upnp:inclusionControl* properties, however the behavior of the ContentDirectory service implementation MUST be the same as if the combination of properties were added sequentially. For example a control point cannot lock, that is set the *upnp:inclusion* and *upnp:objectOwner* properties to the read-only state, by making the *upnp:objectOwner@lock* value equal to "*1*", of an object it does not own.

When the *CONTENT_PROTECTION feature* is implemented a successful invocation of the *CreateObject()* action does not require the inclusion of a *upnp:objectOwner* (See Appendix B.21.1) or *upnp:inclusionControl* property (See Appendix B.20.1) however, if one or both are supplied the following MUST apply:

- If the *upnp:inclusionControl* property is provided by the creating control point, then the provided child properties MUST be valid *Role(s)* (See Appendix F.1) otherwise the ContentDirectory service implementation MUST return the action level error code 734 in Table F-6.
- If the *upnp:objectOwner* property is provided by the creating control point, then the provided child properties MUST be valid *Role(s)* (See Appendix F.1) otherwise the ContentDirectory service MUST return the action level error code 734 in Table F-6.
- If the *upnp:objectOwner* property is provided by the creating control point with the *upnp:objectOwner@lock* property having a value of "*1*"(true) then the creating control point MUST be included as an *owner* of that object and a write related *Role* for the object, otherwise, the ContentDirectory service implementation MUST return the action level error code 735 in Table F-6.

Note, it is RECOMMENDED that when a new container is created, that any *upnp:inclusionControl* property pre-existing for the parent container be propagated to the new child container.

### F.2.2 *UpdateObject()* action with CONTENT_PROTECTION feature

When the *CONTENT_PROTECTION feature* is implemented, a successful invocation of the *UpdateObject()* action (assuming the control point has *Action level access*) MUST be on a object that the control point has *Object level access* to (See Appendix F.1.4). In that case, a control point is allowed to modify an objects *upnp:objectOwner* and *upnp:inclusionControl* properties, however the behavior of the

ContentDirectory service implementation MUST be the same as if the combination of properties were added sequentially. For example, a control point cannot delete a *upnp:inclusionControl* property of an object if its *upnp:objectOwner@lock* property has value "*1*"(true).

In the case, when an *UpdateObject()* action is invoked that includes modification of the *upnp:objectOwner* property the following apply:

- If the *upnp:objectOwner* property is currently present and the *upnp:objectOwner@lock* property has a value of "*1*"(true), then the invoking control point MUST be a current *owner* of the object and include a modification in the *NewTagValue* input parameter that first changes the *upnp:objectOwner@lock* property to a value of "*0*"(false) before it modifies any other value of the *upnp:objectOwner* property, otherwise, the ContentDirectory service implementation MUST return the action level error code 735 in Table F-6.

- If the *upnp:objectOwner* property is currently present and the *upnp:objectOwner@lock* property has a value of "*0*"(false), then the invoking control point MUST be a current *owner* of the object before it can change the *upnp:objectOwner@lock* property to a value of "*1*" (true), otherwise the ContentDirectory service implementation MUST return the action level error code 735 in Table F-6.

- If the invoking control point attempts to modify the *upnp:objectOwner* property with an invalid *Role* then the ContentDirectory service implementation MUST return the action level error code 734 in Table F-6.

In the case, when an *UpdateObject()* action is invoked that includes modification or deletion of the *upnp:inclusionControl* property the following apply:

- If the *upnp:objectOwner* property is currently present and the *upnp:objectOwner@lock* property has a value of "*1*"(true) then the invoking control point MUST be a current *owner* of the object and include a modification in the *NewTagValue* input parameter that first changes the *upnp:objectOwner@lock* property to a value of "*0*"(false) before it modifies any other value of the *upnp:inclusionControl* property, otherwise, the ContentDirectory service modification MUST return the action level error code 736 in Table F-6.

- If the invoking control point attempts to modify the *upnp:inclusionControl* property when an *upnp:objectOwner* property is present and the invoking control point is not a current *owner* of the object, then the ContentDirectory service implementation MUST return the action level error code 735 in Table F-6

- If the invoking control point attempts to modify the *upnp:inclusionControl* property with an invalid *Role* then the ContentDirectory service implementation MUST return the action level error code 734 in Table F-6.

**Table F-6: Error Codes for *CreateObject()* and *UpdateObject()* action with *upnp:objectOwner* and *upnp:objectOwner* property**

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 734 | Invalid *Role* for *upnp:inclusionControl* or *upnp:objectOwner* property | The *upnp:inclusionControl* or *upnp:objectOwner* property contains at least one invalid control point *Role*. |
| 735 | Invalid *Owner* | The *upnp:objectOwner* or *upnp:inclusionControl* property does not include a *Role* allowed to modify the property.. |

| | | |
|---|---|---|
| 736 | Object locked | The upnp:objectOwner property or upnp:inclusionControl property cannot be modified since they are currently locked. |

### F.2.3 Browse() action with CONTENT_PROTECTION feature

When the *CONTENT_PROTECTION feature* is implemented, then the Content Directory service implementation MUST modify the *Browse()* response as follows:

- When the *Browse()* action is invoked on a container object and the invoking control point has *Action level access* to that container, then the *Browse()* action response MUST be modified as follows:
    - For child objects that the control point has *Object level access* to, the full set of metadata MUST be returned with the following exception.
        - If the object contains an *@refID* which references an item that the invoking control does not have access to, then the returned response MUST be as below (see next bullet).
    - For child objects that the control point does not have *Object level access* to, a reduced set of metadata MUST be returned restricted to the following properties:
        - *item* or *container* property,
        - *@id* property,
        - *@parentID* property,
        - *@restricted* property,
        - *upnp:class* property,
        - *dc:title* property with value "[Access Not Allowed]",
        - *@refID* property (when present),
        - *upnp:objectUpdateID* property (when present),
        - *upnp:containerUpdateID* property (when present),
        - *@childCount* property (when present),
        - *upnp:totalDeletedChildCount* property (when present).

### F.2.4 Search() action with CONTENT_PROTECTION feature

When the *CONTENT_PROTECTION feature* is implemented, then the Content Directory service implementation MUST construct the *Search()* response as if the available metadata is from a *Restricted DIDL-Lite view* (See Section 2.2.25.1). Note, even though the *AV:SuperReader* and *AV:SuperWriter Roles* are implicitly members of all *upnp:objectOwner* and *upnp::inclusionControl* properties, searches for their values in these properties is discouraged since the results may be implementation dependent, that is, the ContentDirectory service implementation is not required to explicitly include them in the actual object metadata.

### F.2.5 FreeFormQuery() action when CONTENT_PROTECTION feature

When the *CONTENT_PROTECTION feature* is implemented, then the Content Directory service implementation MUST construct the *FreeFormQuery()* response as if the available metadata is from a *Restricted DIDL-Lite view*. Note, even though the *AV:SuperReader* and *AV:SuperWriter Roles* are implicitly members of all *upnp:objectOwner* and *upnp:inclusionControl* properties, queries for their values in these properties is discouraged since the results may be implementation dependent, that is, the ContentDirectory service implementation is not required to explicitly include them in the actual object metadata

### F.2.6 DestroyObject() action with CONTENT_PROTECTION feature

When the *CONTENT_PROTECTION feature* is implemented, then the Content Directory service implementation MUST only delete objects that the invoking control point has *Action level access* and *Object level access*. If the target of the *DestroyObject()* action is a container then the action MUST destroy (delete) all descendant items and containers that the control point has both action and *Object level access*. If there are descendant objects that cannot be destroyed (deleted) then an ancestor container path to the root container MUST be preserved.

### F.2.7 MoveObject() action with CONTENT_PROTECTION feature

When the *CONTENT_PROTECTION feature* is implemented, then the control point invoking the *MoveObject()* action MUST have read related access to the all objects it will be moving and write related access to the destination container otherwise the ContentDirectory service MUST return an *Object level access* error code as follows:

- If the control point does not have *Object level access* to an object to be moved, the action MUST fail with error code 737 in Table F-7.
- If the control point does not have *Object level access* to the destination container, the action MUST fail with error code 738 in Table F-7.
- When both the error conditions described above occur in the same *MoveObject()* action, error code 738 MUST fail with error code 738 in Table F-7.

**Table F-7: Error Codes for *MoveObject()* action with *CONTENT_PROTECTION feature***

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UpnP Device Architecture section on Control. |
| 737 | Input object not authorized | *MoveObject()* failed because the control point does not have *Object level access* to at least one of the objects it is trying to move. |
| 738 | Output object not authorized | *MoveObject()* failed because the control point does not have *Object level access* to the target container. |

### F.2.8 DeleteResource() action with CONTENT_PROTECTION feature

When the *CONTENT_PROTECTION feature* is implemented, then the control point invoking the *DeleteResource()* action MUST have access to the all objects containing target resources otherwise the ContentDirectory service implementation MUST return the action level error code 739 in Table F-8.

**Table F-8: Error Codes for *DeleteResource()* action with *CONTENT_PROTECTION feature***

| errorCode | errorDescription | Description |
|---|---|---|
| 400-499 | TBD | See UPnP Device Architecture section on Control. |
| 500-599 | TBD | See UPnP Device Architecture section on Control. |
| 739 | Source resource access denied | *DeleteResource()* failed because the control point does not have *Role* permissions to invoke this action on at least one of the objects referencing the resource specified by the *ResourceURI* argument. |

# Appendix G. Content Authoring using Object Linking (Informative)

## G.1 Introduction

Object Linking provides content authors a way to organize and define relationships between separate ContentDirectory service items.

In prior ContentDirectory service versions, content items are largely self-describing. However, UPnP provides control points little guidance in terms of how to organize individual content items into groups. Each item is considered separate entity and a control point can arrange these entities as it sees fit.

In terms of presenting content to an end-user, many control points rely on the container organization provided by the ContentDirectory service to convey how content is organized. UPnP leaves container organization to the discretion of the ContentDirectory service implementation.

This provides content authors little support in defining presentations that span more than a single ContentDirectory service item. Unfortunately, the rendering of single item content leaves no possibility of end-user interactivity since UPnP treats rendered items as monolithic objects. A much richer user experience is possible if content items include information to assist control points in directing the flow between content items.

To some extent Object Linking relieves the control point of some design decisions relating to content organization by describing some basic constructs to organize content. This provides control points information from content authors as to how to present groups of related items.

## G.2 Object Linking Metadata Properties

The Object Linking properties:

- *upnp:objectLink*
- *upnp:objectLinkRef*

Provide a way to create ordered lists of ContentDirectory service items and allows an item within a list to reference an item within the same list or within a different list.

Each *upnp:objectLink* property within an item indicates the next and previous *items* that are participating in a list. In addition, each *upnp:objectLink* property indicates the first (or head) item of a list.

The *upnp:objectLink* property for the head of each list contains child properties that:

- Provides a displayable title for the list.
- Provides the intended handling for members in the list.
- Provides a pointer to one or more start items which can directly or indirectly reference this list
- Provides how items on the list are related to each other.
- Provides what to do when the end of the list is reached.

As discussed earlier, an *item* within a list can “reference” an item within the same list or within a different list. This reference is defined by the *upnp:objectLinkRef* property. The *upnp:objectLinkRef @targetGroupID* and *upnp:objectLinkRef@targetObjID* properties identify the target list and item being referenced. Occurrences of the *upnp:objectLinkRef* property do not alter the selection of the next item to be played unless the end-user indicates to the control point that the reference is to be played. When this selection is made, the type of list which the selected item is participating in provides information to the control point as to how to proceed.

A *upnp:objectLinkRef* property with its *upnp:objectLinkRef@return* property set to ”*1*” indicates that this is a “useful” return point. A control point upon encountering this property stores return information to the current list and current item and current playback state. A control point will only retain return information if the end-user actually selects the corresponding *upnp:objectLinkRef* property causing a new item and list to be processed. The control point’s implementation of the return function can vary from a simple return stack to a more complex history buffer allowing an end-user to directly select a previously saved return point. When the end-user selects a saved return point, the control point restores the playback state that existed at the time the return point was saved.

Since a given ContentDirectory service item can participate in multiple Object Linked lists, there could be multiple *upnp:objectLink* and *upnp:objectLinkRef* properties present in an item. However, each list is required to be assigned a unique “Group ID” value. All *upnp:objectLink* and *upnp:objectLinkRef* properties containing the same *upnp:objectLink@groupID* or *upnp:objectLinkRef@groupID* properties are related to the same uniquely defined list.

There are two major categories of Object Link lists:

- Lists intended for playback

  Members of Playback lists are intended to be played back with seamless transitions between the end of playback of a list item and the start of playback of the next item on the list.

- Lists intended for indexing

  Members of Index lists are intended to organize content for end-users and to allow end-users to quickly select portions of a potentially large collection of items for playback. The display of items participating in an Index list is dependent on the capabilities of the control point. A control point processing of an Index list can:

  - Display an Index item’s title metadata locally on the control point’s user interface.
  - Display an Index item’s title metadata and additionally support a “preview” window to display an Index content-binary item locally on the control point’s user interface.
  - Display an Index item’s title metadata locally and use an associated renderer to display the Index item’s content-binary.

Each item in an Index list is expected to contain a *upnp:objectLinkRef* property to reference an item to be accessed if the end-user selects the Index item for playback.

Note that Object Linking is relatively free from linkage constraints:

- Items of various of media classes can be linked together in a Playback list using the *upnp:objectLink* property.
- An Index list can contain *upnp:objectLinkRef* property which point to other Index lists as well as to Playback lists.
- A Playback list can contain *upnp:objectLinkRef* properties that refer to Index lists as well as to Playback lists.

## G.3 Table of Contents (Index) Lists

Suppose a content author wants to “outline” a large set of content. Traditional UPnP control point handling would list each content item regardless of whether the items represented were meaningful entry points into the larger content set or not. The “Table of Contents” or “Index” list is introduced in order to organize large sets of content providing users quick access to parts of the content set.

The Index list uses the *upnp:objectLinkRef* property as described in the previous section. However, the handling of members of the Index list is different than other lists.

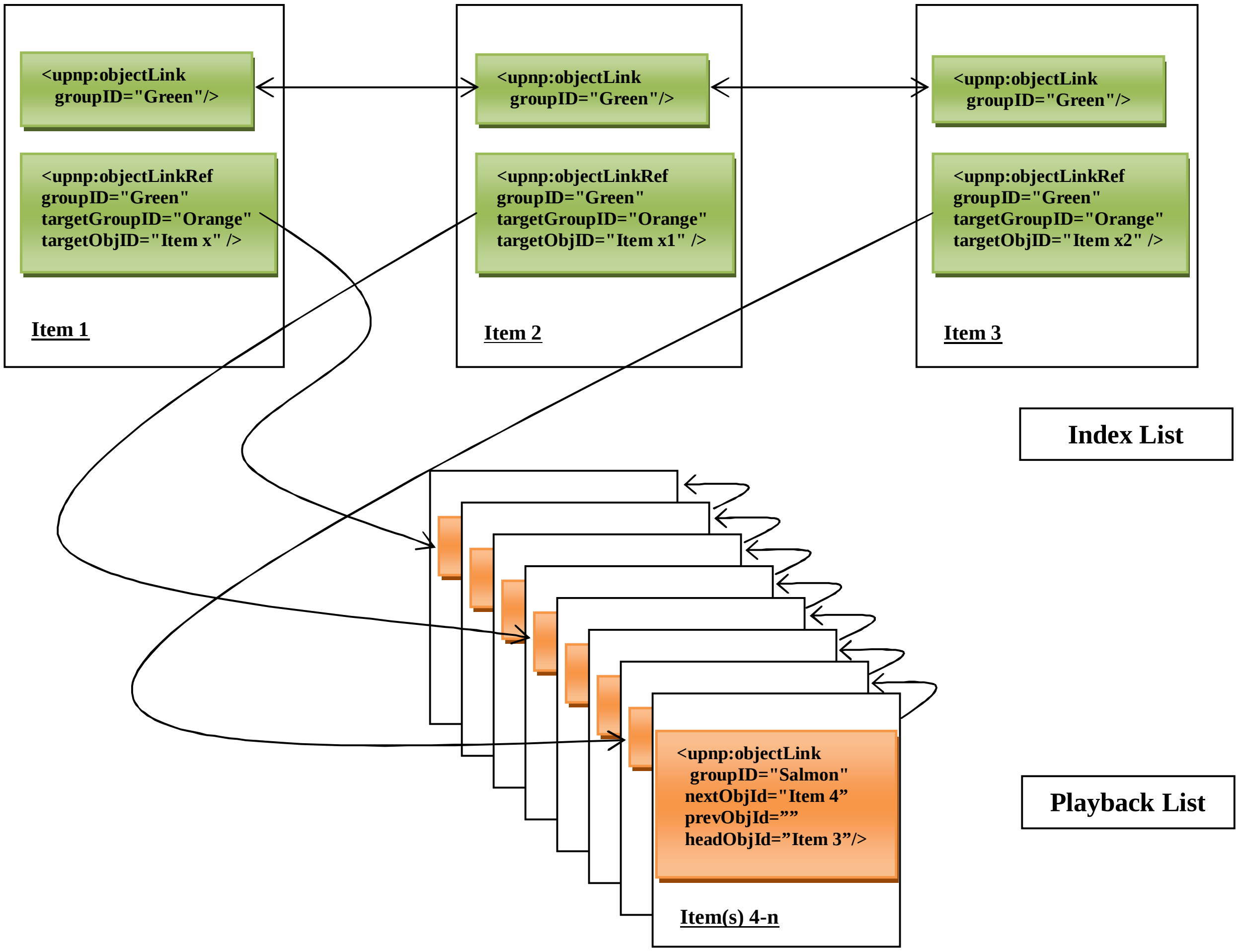

<upnp:objectLink groupID="Green"/>
<upnp:objectLinkRef groupID="Green" targetGroupID="Orange" targetObjID="Item x" />
Item 1
<upnp:objectLink groupID="Green"/>
<upnp:objectLinkRef groupID="Green" targetGroupID="Orange" targetObjID="Item x1" />
Item 2
<upnp:objectLink groupID="Green"/>
<upnp:objectLinkRef groupID="Green" targetGroupID="Orange" targetObjID="Item x2" />
Item 3
Index List
<upnp:objectLink groupID="Salmon" nextObjId="Item 4" prevObjId="" headObjId="Item 3"/>
Item(s) 4-n
Playback List

**Figure 6: Example of Object Link "Index" list**

Members of an Index list are intended to be listed at a control point. The items in an Index list are related by *upnp:objectLink* properties just as any other list. However, the members of an Index list are processed as a group to form a listing similar to what a control point would list when displaying contents of a UPnP container. Each member of the Index list normally contains a *upnp:objectLinkRef* property which describes an entry point into a list of content. In the current figure, content is identified as being contained in the "Orange" list but an Index list could refer to elements in multiple lists. When an end-user selects a member of an Index list for playback, the *upnp:objectLinkRef* property corresponding to the Index list is processed. This processing is the same as if a *upnp:objectLinkRef* property was selected during normal playback. As discussed in the previous section, processing of a *upnp:objectLinkRef* property causes a transition to the indicated list and to the indicated member within that list.

## G.4 Playback and Step Lists

Content authors can use Playback and Step lists to relate individual items intended for playback. These items could be segments of a larger content item (See AppendixB.16, "Segmentation-related Properties").

Playback and Step lists are described as follows:

- Playback list

The control point is expected to play media items described in the current list (as indicated by upnp:objectLink@groupID) on the rendering device sequentially without pauses between items. This does not preclude normal AVTransport controls such as Play(), Next(), Previous(), Pause() and Stop(). However, when processing Object Linking metadata, the next and previous items (upnp:objectLink@nextObjID, upnp:objectLink@prevObjID) are identified by the current list being processed. In addition, the control point is expected to indicate the presence of upnp:objectLinkRef properties. The approach a control point uses to display this information to the end-user is left to the control point implementer. In the case of DVD media, this is commonly done by displaying a (camera angle) icon during playback.

- Step list

  The control point is expected to play the media items described by the list sequentially, but is expected to pause after playing each item. The AVTransport Next() function (or equivalent local remote control key) indicates the user wishes to continue with the next item.

## G.5 References between lists of items

Suppose a content author wants to make additional content available, but does not want the content to be automatically played. The following diagram illustrates metadata to provide references between list items:

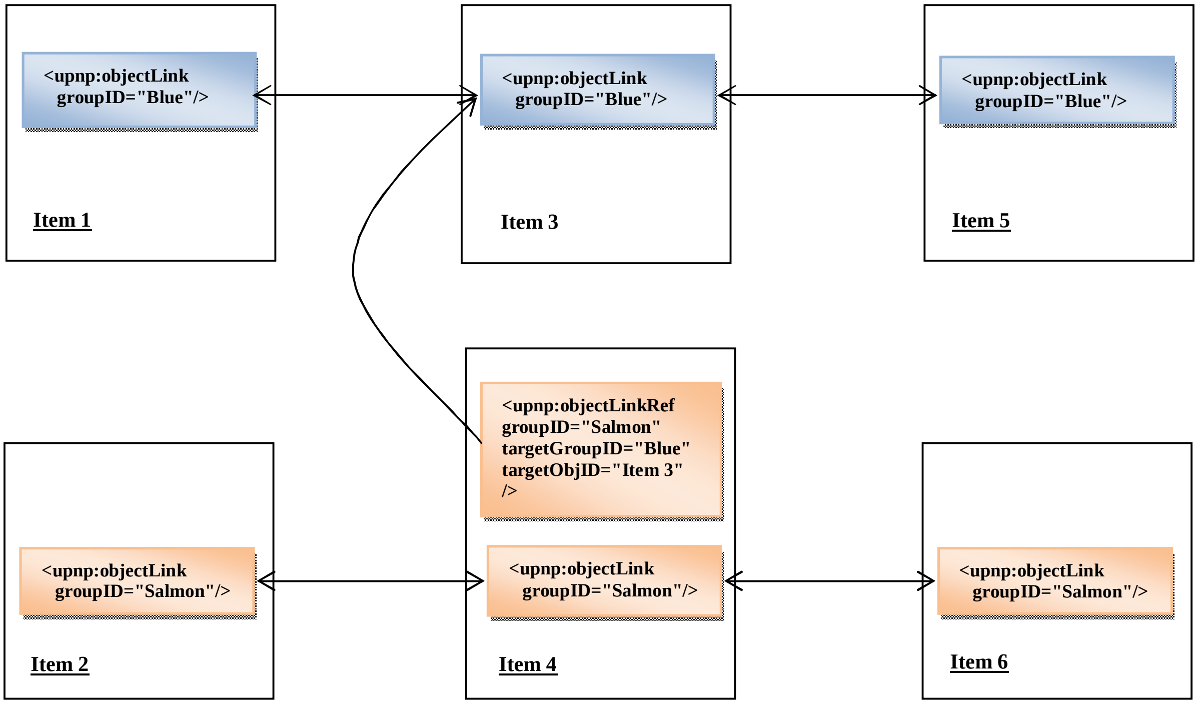

<upnp:objectLink groupID="Blue"/>
Item 1
<upnp:objectLink groupID="Blue"/>
Item 3
<upnp:objectLink groupID="Blue"/>
Item 5
<upnp:objectLinkRef groupID="Salmon" targetGroupID="Blue" targetObjID="Item 3" />
<upnp:objectLink groupID="Salmon"/>
Item 2
<upnp:objectLink groupID="Salmon"/>
Item 4
<upnp:objectLink groupID="Salmon"/>
Item 6

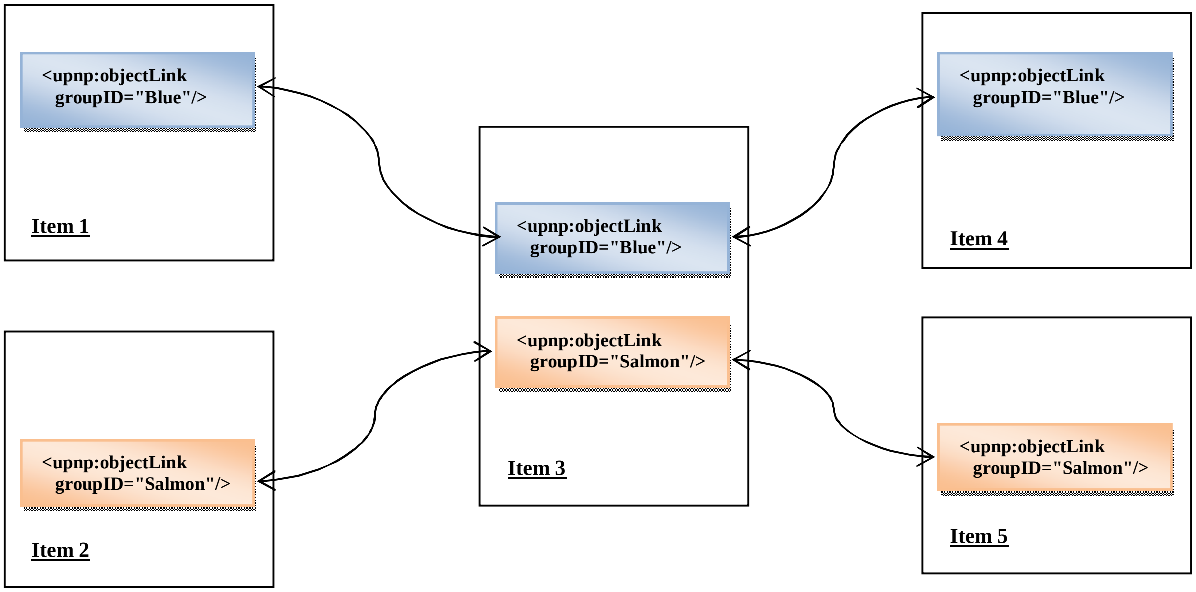
**Figure 7: Example of Object Link list reference**

The upnp:objectLinkRef property shown here is used to describe the availability of optional content. This property describes an optional branch point to an item either in the same list or a different list. In this example the upnp:objectLinkRef property in Item 4 is associated with the “Salmon” list. This property indicates the optional content to be made available if the “Salmon” list is being played. If the end-user elects to play this optional content, the current list would transition from the “Salmon” list to the “Blue” list as indicated by the upnp:objectLink@targetGroupID property and Item 3 in that the “Blue” list would be accessed as indicated by the upnp:objectLink@targetObjID property.

## G.6 Sharing items in multiple lists

This diagram shows five items (Item 1-5) participating in two lists (Blue and Salmon). Item 3 participates in both lists. Although the same metadata property upnp:objectLink is used to describe the linkages between the items, the upnp:objectLink@groupID dependent property indicates which list the upnp:objectLink property is describing.

By utilizing the flexibility of UPnP metadata to provide multi-valued properties we can include a given item in multiple distinct lists:



<upnp:objectLink groupID="Blue"/>
Item 1
<upnp:objectLink groupID="Salmon"/>
Item 2
<upnp:objectLink groupID="Blue"/>
<upnp:objectLink groupID="Salmon"/>
Item 3
<upnp:objectLink groupID="Blue"/>
Item 4
<upnp:objectLink groupID="Salmon"/>
Item 5

**Figure 8: Example of Object Link lists sharing an item**

## G.7 Return Model

An Object Linking "return" is functionally similar to an "automatic bookmark" function. The occurrence of a upnp:objectLinkRef property with a upnp:objectLinkRef@return property set to "1" indicates that the control point retain return information to the current Object Link list if the end-user selects the upnp:objectLinkRef property providing new group ID and item @id.

See the upnp:objectLinkRef@return property for further details on designating return points.

A return point for a upnp:objectLinkRef property appearing in a "Playback/Step" list is recorded at the point at which the upnp:objectLinkRef property processed, that is the point at which the end-user indicated they wanted to view the alternate list of items indicated by the upnp:objectLinkRef property.

A return point for a upnp:objectLinkRef property appearing in an Index list indicates that the Index list is to be redisplayed with the control point's "cursor" (if any) on the Index list item the end-user had previously selected for playback.

When a control point saves a return point, it would typically use a LIFO return stack. Each end-user request for a control point to do a return will remove and restore the latest saved information from the return stack. Control points will typically clear the return stack when the end-user selects a new starting object.

A control point can additionally provide a user initiated bookmark function, that is, a user-selected return point. However, implementation of this additional function is vendor specific.

## G.8 Control Point processing of Object Linked items

The *upnp:objectLink* property connects items together to form lists. These lists are then connected to each other by *upnp:objectLinkRef* properties. The lists will typically form a hierarchy with one or more item at the top of the hierarchy.

A control point can determine the initial starting object ID and group ID by accessing any object which contains *upnp:objectLink* properties. The *upnp:objectLink@groupID* and *upnp:objectLink@headObjID* properties of each item identify a *upnp:objectLink* property at the head of the Object Linked list the object participates in. The *upnp:objectLink::startInfo* property for the selected *upnp:objectLink* property at the head of the list identifies a starting object ID and a starting group ID values.

An additional method to determine if a *upnp:objectLink* property represents a displayable title is to search for a *upnp:objectLink::startObject* property with the value "*1*".

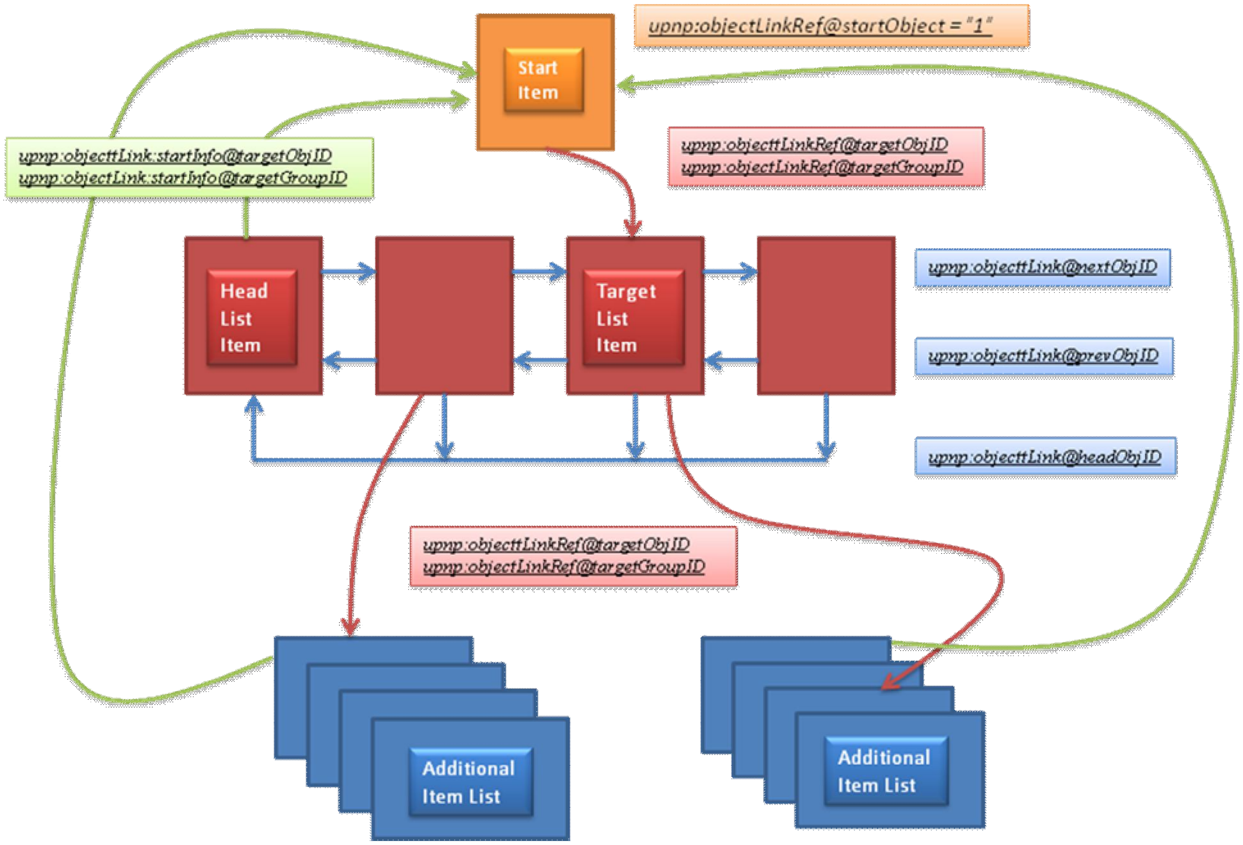

upnp:objectLinkRef@startObject = "1"
Start Item
upnp:objecttLink:startInfo@targetObjID
upnp:objectLink:startInfo@targetGroupID
upnp:objecttLinkRef@targetObjID
upnp:objectLinkRef@targetGroupID
Head List Item
Target List Item
upnp:objecttLink@nextObjID
upnp:objecttLink@prevObjID
upnp:objecttLink@headObjID
upnp:objecttLinkRef@targetObjID
upnp:objectLinkRef@targetGroupID
Additional Item List
Additional Item List

**Figure 9: Example of Starting Item and Object Link list hierarchy**

# Appendix H. Example ContentDirectory Hierarchy

The following example ContentDirectory hierarchy is used in the FreeFormQuery() search examples. The notation used is not *DIDL-Lite View.* An XML-like notation is used to express the nesting of containers and items.

```
<container id="0" parentID="-1" restricted="1">
    <dc:title>Example Server</dc:title>
    <upnp:class>object.container</upnp:class>

    <container id="1" parentID="0" restricted="1">
        <dc:title>Music</dc:title>
        <upnp:class>object.container</upnp:class>

        <container id="1-1" parentID="1" restricted="1">
            <dc:title>Music by Albums</dc:title>
            <upnp:class>object.container</upnp:class>

            <container id="1-1-1" parentID="1-1" restricted="1" >
                <dc:title>Album 1</dc:title>
                <upnp:class>object.container.album.musicAlbum</upnp:class>
                <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>
                <item id="1-1-1-4" parentID="1-1-1" restricted="1">
                    <dc:title>Album 1 Song 4</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-MP3-14.mp3
                    </res>
                </item>
                <item id="1-1-1-5" parentID="1-1-1" restricted="1">
                    <dc:title>Album 1 Song 5</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-MP3-15.mp3
                    </res>
                </item>
                <item id="1-1-1-6" parentID="1-1-1" restricted="1">
                    <dc:title>Album 1 Song 6</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-MP3-16.mp3
                    </res>
                </item>
                <item id="1-1-1-7" parentID="1-1-1" restricted="1">
                    <dc:title>Album 1 Song 7</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-MP3-17.mp3
                    </res>
                </item>
                <item id="1-1-1-8" parentID="1-1-1" restricted="1">
                    <dc:title>Album 1 Song 8</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-MP3-18.mp3
                    </res>
                </item>
                <item id="1-1-1-9" parentID="1-1-1" restricted="1">
                    <dc:title>Album 1 Song 9</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-MP3-19.mp3
                    </res>
                </item>
                <item id="1-1-1-10" parentID="1-1-1" restricted="1">
                    <dc:title>Album 1 Song 10</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-MP3-110.mp3
                    </res>
                </item>
                <item id="1-1-1-11" parentID="1-1-1" restricted="1">
                    <dc:title>Album 1 Song 11</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-MP3-111.mp3
                    </res>
                </item>
                <item id="1-1-1-12" parentID="1-1-1" restricted="1">
                    <dc:title>Album 1 Song 12</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-MP3-112.mp3
                    </res>
                </item>
                <item id="1-1-1-13" parentID="1-1-1" restricted="1">
                    <dc:title>Album 1 Song 13</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-MP3-113.mp3
                    </res>
                </item>
            </container>

            <container id="1-1-2" parentID="1-1" restricted="1">
                <dc:title>Album 2</dc:title>
                <upnp:class>object.container.album.musicAlbum</upnp:class>
                <item id="1-1-2-1" parentID="1-1-2" restricted="1">
                    <dc:title>Album 2 Song 1</dc:title>
                    <upnp:class>object.item.audioItem</upnp:class>
                    <upnp:album>Album 2</upnp:album>
                    <upnp:genre>Unknown</upnp:genre>
                    <res protocolInfo="http-get:*:audio/mpeg:*">
                        http://10.0.0.1/audio/O-MP3-21.mp3
                    </res>
                </item>
                <item id="1-1-2-2" parentID="1-1-2" restricted="1">
                    <dc:title>Album 2 Song 2</dc:title>
                    <upnp:class>object.item.audioItem</upnp:class>
                    <upnp:album>Album 2</upnp:album>
                    <upnp:genre>Unknown</upnp:genre>
                    <res protocolInfo="http-get:*:audio/mpeg:*">
                        http://10.0.0.1/audio/O-MP3-22.mp3
                    </res>
                </item>
                <item id="1-1-2-3" parentID="1-1-2" restricted="1">
                    <dc:title>Album 2 Song 3</dc:title>
                    <upnp:class>object.item.audioItem</upnp:class>
                    <upnp:album>Album 2</upnp:album>
                    <upnp:genre>Unknown</upnp:genre>
                    <res protocolInfo="http-get:*:audio/mpeg:*">
                        http://10.0.0.1/audio/O-MP3-23.mp3
                    </res>
                </item>
                <item id="1-1-2-4" parentID="1-1-2" restricted="1">
                    <dc:title>Album 2 Song 4</dc:title>
                    <upnp:class>object.item.audioItem</upnp:class>
                    <upnp:album>Album 2</upnp:album>
                    <upnp:genre>Unknown</upnp:genre>
                    <res protocolInfo="http-get:*:audio/mpeg:*">
                        http://10.0.0.1/audio/O-MP3-24.mp3
                    </res>
                </item>
                <item id="1-1-2-5" parentID="1-1-2" restricted="1">
                    <dc:title>Album 2 Song 5</dc:title>
                    <upnp:class>object.item.audioItem</upnp:class>
                    <upnp:album>Album 2</upnp:album>
                    <upnp:genre>Unknown</upnp:genre>
                    <res protocolInfo="http-get:*:audio/mpeg:*">
                        http://10.0.0.1/audio/O-MP3-25.mp3
                    </res>
                </item>
                <item id="1-1-2-6" parentID="1-1-2" restricted="1">
                    <dc:title>Album 2 Song 6</dc:title>
                    <upnp:class>object.item.audioItem</upnp:class>
                    <upnp:album>Album 2</upnp:album>
                    <upnp:genre>Unknown</upnp:genre>
                    <res protocolInfo="http-get:*:audio/mpeg:*">
                        http://10.0.0.1/audio/O-MP3-26.mp3
                    </res>
                </item>
```

```
            <item id="1-1-2-7" parentID="1-1-2" restricted="1">
                <dc:title>Album 2 Song 7</dc:title>
                <upnp:class>object.item.audioItem</upnp:class>
                <upnp:album>Album 2</upnp:album>
                <upnp:genre>Unknown</upnp:genre>
                <res protocolInfo="http-get:*:audio/mpeg:*">
                    http://10.0.0.1/audio/O-MP3-27.mp3
                </res>
            </item>
            <item id="1-1-2-8" parentID="1-1-2" restricted="1">
                <dc:title>Album 2 Song 8</dc:title>
                <upnp:class>object.item.audioItem</upnp:class>
                <upnp:album>Album 2</upnp:album>
                <upnp:genre>Unknown</upnp:genre>
                <res protocolInfo="http-get:*:audio/mpeg:*">
                    http://10.0.0.1/audio/O-MP3-28.mp3
                </res>
            </item>
            <item id="1-1-2-9" parentID="1-1-2" restricted="1">
                <dc:title>Album 2 Song 9</dc:title>
                <upnp:class>object.item.audioItem</upnp:class>
                <upnp:album>Album 2</upnp:album>
                <upnp:genre>Unknown</upnp:genre>
                <res protocolInfo="http-get:*:audio/mpeg:*">
                    http://10.0.0.1/audio/O-MP3-29.mp3
                </res>
            </item>
            <item id="1-1-2-10" parentID="1-1-2" restricted="1">
                <dc:title>Album 2 Song 10</dc:title>
                <upnp:class>object.item.audioItem</upnp:class>
                <upnp:album>Album 2</upnp:album>
                <upnp:genre>Unknown</upnp:genre>
                <res protocolInfo="http-get:*:audio/mpeg:*">
                    http://10.0.0.1/audio/O-MP3-210.mp3
                </res>
            </item>
            <item id="1-1-2-11" parentID="1-1-2" restricted="1">
                <dc:title>Album 2 Song 11</dc:title>
                <upnp:class>object.item.audioItem</upnp:class>
                <upnp:album>Album 2</upnp:album>
                <upnp:genre>Unknown</upnp:genre>
                <res protocolInfo="http-get:*:audio/mpeg:*">
                    http://10.0.0.1/audio/O-MP3-211.mp3
                </res>
            </item>
            <item id="1-1-2-12" parentID="1-1-2" restricted="1">
                <dc:title>Album 2 Song 12</dc:title>
                <upnp:class>object.item.audioItem</upnp:class>
                <upnp:album>Album 2</upnp:album>
                <upnp:genre>Unknown</upnp:genre>
                <res protocolInfo="http-get:*:audio/mpeg:*">
                    http://10.0.0.1/audio/O-MP3-212.mp3
                </res>
            </item>
            <item id="1-1-2-13" parentID="1-1-2" restricted="1">
                <dc:title>Album 2 Song 13</dc:title>
                <upnp:class>object.item.audioItem</upnp:class>
                <upnp:album>Album 2</upnp:album>
                <upnp:genre>Unknown</upnp:genre>
                <res protocolInfo="http-get:*:audio/mpeg:*">
                    http://10.0.0.1/audio/O-MP3-213.mp3
                </res>
            </item>
        </container>

    </container>


</container>

<container id="2" parentID="0" restricted="1">
```

```
        <dc:title>Movies</dc:title>
        <upnp:class>object.container</upnp:class>

        <container id="2-1" parentID="2" restricted="1">
            <dc:title>Movies by Title</dc:title>
            <upnp:class>object.container</upnp:class>

            <item id="2-1-1" parentID="2-1" restricted="1">
                <dc:title>Movie 1</dc:title>
                <upnp:class>object.item.videoItem</upnp:class>
                <res protocolInfo="http-get:*:video/mpeg:*">
                    http://10.0.0.1/video/B-MP2PS_N-11.mpeg
                </res>
            </item>
            <item id="2-1-2" parentID="2-1" restricted="1">
                <dc:title>Movie 2</dc:title>
                <upnp:class>object.item.videoItem</upnp:class>
                <res protocolInfo="http-get:*:video/mpeg:*">
                    http://10.0.0.1/video/B-MP2PS_N-12.mpeg
                </res>
            </item>
            <item id="2-1-3" parentID="2-1" restricted="1">
                <dc:title>Movie 3</dc:title>
                <upnp:class>object.item.videoItem</upnp:class>
                <res protocolInfo="http-get:*:video/mpeg:*">
                    http://10.0.0.1/video/B-MP2PS_N-13.mpeg
                </res>
            </item>
            <item id="2-1-4" parentID="2-1" restricted="1">
                <dc:title>Movie 4</dc:title>
                <upnp:class>object.item.videoItem</upnp:class>
                <res protocolInfo="http-get:*:video/mpeg:*">
                    http://10.0.0.1/video/B-MP2PS_N-14.mpeg
                </res>
            </item>
            <item id="2-1-5" parentID="2-1" restricted="1">
                <dc:title>Movie 5</dc:title>
                <upnp:class>object.item.videoItem</upnp:class>
                <res protocolInfo="http-get:*:video/mpeg:*">
                    http://10.0.0.1/video/B-MP2PS_N-15.mpeg
                </res>
            </item>

        </container>
    </container>

    <container id="3" parentID="0" restricted="1">
        <dc:title>Photos</dc:title>
        <upnp:class>object.container</upnp:class>
            <container id="3-1" parentID="3" restricted="1" >
                <dc:title>Album 1</dc:title>
                <upnp:class>object.container.album.musicAlbum</upnp:class>
                <item id="3-1-1" parentID="3-1" restricted="1">
                    <dc:title>Album 1 Photo 1</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-11.jpg
                    </res>
                </item>
                <item id="3-1-2" parentID="3-1" restricted="1">
                    <dc:title>Album 1 Photo 2</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-12.jpg
                    </res>
                </item>
                <item id="3-1-3" parentID="3-1" restricted="1">
                    <dc:title>Album 1 Photo 3</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-13.jpg
                    </res>
                </item>
                <item id="3-1-4" parentID="3-1" restricted="1">
                    <dc:title>Album 1 Photo 4</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-14.jpg
                    </res>
                </item>
                <item id="3-1-5" parentID="3-1" restricted="1">
                    <dc:title>Album 1 Photo 5</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-15.jpg
                    </res>
                </item>
                <item id="3-1-6" parentID="3-1" restricted="1">
                    <dc:title>Album 1 Photo 6</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-16.jpg
                    </res>
                </item>
                <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>
                <item id="3-1-12" parentID="3-1" restricted="1">
                    <dc:title>Album 1 Photo 12</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-112.jpg
                    </res>
                </item>
                <item id="3-1-13" parentID="3-1" restricted="1">
                    <dc:title>Album 1 Photo 13</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-113.jpg
                    </res>
                </item>
            </container>
    </container>
</container>
```