
**Information technology — UPnP
Device Architecture —**

**Part 20-12:
Audio video device control protocol —
Level 4 — Content directory service**

Technologies de l'information — Architecture de dispositif UPnP —

*Partie 20-12: Protocole de contrôle de dispositif audio-vidéo —
Niveau 4 — Service d'Annuaire de contenu*



Reference number
ISO/IEC 29341-20-12:2017(E)

Documents
Document Sharing Hub
© ISO/IEC 2017



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2017, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

CONTENTS

1	Scope	1
2	Normative references	1
3	Terms, definitions, symbols and abbreviations.....	5
3.1	Provisioning terms	5
3.2	Symbols	5
4	Notations and Conventions	5
4.1	Notation	5
4.1.1	Data Types	6
4.1.2	Strings Embedded in Other Strings.....	6
4.1.3	Extended Backus-Naur Form.....	6
4.2	Derived Data Types	7
4.2.1	Summary	7
4.2.2	CSV Lists.....	7
4.3	Management of XML Namespaces in Standardized DCPs.....	8
4.3.1	Namespace Prefix Requirements.....	12
4.3.2	Namespace Names, Namespace Versioning and Schema Versioning	13
4.3.3	Namespace Usage Examples	15
4.4	Vendor-defined Extensions	15
4.4.1	Vendor-defined Action Names	15
4.4.2	Vendor-defined State Variable Names	15
4.4.3	Vendor-defined XML Elements and attributes	16
4.4.4	Vendor-defined Property Names.....	16
5	Service Modeling Definitions.....	16
5.1	Service Type	16
5.2	Key Concepts	16
5.2.1	<i>On-line</i> and <i>Off-line</i> Network States.....	16
5.2.2	object	16
5.2.3	Object Identity	17
5.2.4	Object Lifetime.....	18
5.2.5	Object Modification	18
5.2.6	class.....	19
5.2.7	<u><i>item</i></u>	19
5.2.8	<u><i>container</i></u>	19
5.2.9	Container Modification	19
5.2.10	ContentDirectory Tracking Changes Option	20
5.2.11	<u><i>ContainerUpdateIDValue</i></u> Indicator.....	20
5.2.12	ContentDirectory Service Object Organization	21
5.2.13	Hierarchical location	21
5.2.14	Subtree.....	22
5.2.15	Subtree Updates	22
5.2.16	XML Document	23
5.2.17	XML Fragment	23
5.2.18	DIDL-Lite XML Document.....	24
5.2.19	<i>CDS View</i>	25
5.2.20	<i>CDS Properties</i>	26

5.2.21	<i>reference, reference item, referenced item</i>	27
5.2.22	<i>CDS feature</i>	28
5.2.23	<i>Metadata vs. Foreign Metadata</i>	28
5.2.24	<i>Embedded XML Documents</i>	28
5.2.25	<i>Device Protection Option</i>	28
5.2.26	<i>Device Mode Option</i>	30
5.2.27	<i>Shortcut</i>	30
5.3	<i>State Variables</i>	30
5.3.1	<i>State Variable Overview</i>	31
5.3.2	<u><i>SearchCapabilities</i></u>	32
5.3.3	<u><i>SortCapabilities</i></u>	32
5.3.4	<u><i>SortExtensionCapabilities</i></u>	33
5.3.5	<u><i>SystemUpdateID</i></u>	33
5.3.6	<u><i>ContainerUpdateIDs</i></u>	35
5.3.7	<u><i>ServiceResetToken</i></u>	36
5.3.8	<u><i>LastChange</i></u>	37
5.3.9	<u><i>TransferIDs</i></u>	41
5.3.10	<u><i>FeatureList</i></u>	41
5.3.11	<u><i>DeviceMode</i></u>	41
5.3.12	<u><i>DeviceModeStatus</i></u>	42
5.3.13	<u><i>PermissionsInfo</i></u>	43
5.3.14	<u><i>A_ARG_TYPE_ObjectID</i></u>	45
5.3.15	<u><i>A_ARG_TYPE_Result</i></u>	45
5.3.16	<u><i>A_ARG_TYPE_SearchCriteria</i></u>	45
5.3.17	<u><i>A_ARG_TYPE_BrowseFlag</i></u>	47
5.3.18	<u><i>A_ARG_TYPE_Filter</i></u>	47
5.3.19	<u><i>A_ARG_TYPE_SortCriteria</i></u>	49
5.3.20	<u><i>A_ARG_TYPE_Index</i></u>	50
5.3.21	<u><i>A_ARG_TYPE_Count</i></u>	50
5.3.22	<u><i>A_ARG_TYPE_UpdateID</i></u>	50
5.3.23	<u><i>A_ARG_TYPE_TransferID</i></u>	50
5.3.24	<u><i>A_ARG_TYPE_TransferStatus</i></u>	51
5.3.25	<u><i>A_ARG_TYPE_TransferLength</i></u>	51
5.3.26	<u><i>A_ARG_TYPE_TransferTotal</i></u>	51
5.3.27	<u><i>A_ARG_TYPE_TagValueList</i></u>	51
5.3.28	<u><i>A_ARG_TYPE_URI</i></u>	51
5.3.29	<u><i>A_ARG_TYPE_CDSView</i></u>	51
5.3.30	<u><i>A_ARG_TYPE_QueryRequest</i></u>	51
5.3.31	<u><i>A_ARG_TYPE_QueryResult</i></u>	52
5.3.32	<u><i>A_ARG_TYPE_FFQCapabilities</i></u>	53
5.3.33	<u><i>A_ARG_TYPE_CPID</i></u>	54
5.3.34	<u><i>A_ARG_TYPE_DeviceModelID</i></u>	55
5.3.35	<u><i>A_ARG_TYPE_DeviceModeRequest</i></u>	55
5.4	<i>Eventing and Moderation</i>	58
5.5	<i>Actions</i>	59
5.5.1	<i>Action Overview</i>	59
5.5.2	<u><i>GetSearchCapabilities()</i></u>	61
5.5.3	<u><i>GetSortCapabilities()</i></u>	61

5.5.4	<u>GetSortExtensionCapabilities()</u>	61
5.5.5	<u>GetFeatureList()</u>	62
5.5.6	<u>GetSystemUpdateID()</u>	62
5.5.7	<u>GetServiceResetToken()</u>	63
5.5.8	<u>Browse()</u>	63
5.5.9	<u>Search()</u>	65
5.5.10	<u>CreateObject()</u>	66
5.5.11	<u>DestroyObject()</u>	72
5.5.12	<u>UpdateObject()</u>	73
5.5.13	<u>MoveObject()</u>	79
5.5.14	<u>ImportResource()</u>	80
5.5.15	<u>ExportResource()</u>	81
5.5.16	<u>DeleteResource()</u>	82
5.5.17	<u>StopTransferResource()</u>	83
5.5.18	<u>GetTransferProgress()</u>	84
5.5.19	<u>CreateReference()</u>	84
5.5.20	<u>FreeFormQuery()</u>	85
5.5.21	<u>GetFreeFormQueryCapabilities()</u>	87
5.5.22	<u>RequestDeviceMode()</u>	88
5.5.23	<u>ExtendDeviceMode()</u>	89
5.5.24	<u>CancelDeviceMode()</u>	90
5.5.25	<u>GetDeviceMode()</u>	91
5.5.26	<u>GetDeviceModeStatus()</u>	91
5.5.27	<u>GetPermissionsInfo()</u>	92
5.5.28	Non-Standard Actions Implemented by a UPnP Vendor	92
5.5.29	Common Error Codes	92
6	XML Service Description	94
7	Test	106
Annex A	(normative) Schemas	107
A.1	DIDL-Lite	107
A.2	UPnP Elements	107
A.3	Dublin Core Subset Elements	107
A.4	Event Schema	107
A.5	<u>FeatureList</u> State Variable Schema	107
Annex B	(normative) AV Working Committee Properties	108
B.1	Base Properties	117
B.1.1	<u>@id</u>	117
B.1.2	<u>@parentID</u>	118
B.1.3	<u>@refID</u>	118
B.1.4	<u>@restricted</u>	118
B.1.5	<u>@searchable</u>	118
B.1.6	<u>@childCount</u>	118
B.1.7	<u>@childContainerCount</u>	118
B.1.8	<u>dc:title</u>	119
B.1.9	<u>dc:creator</u>	119
B.1.10	<u>res</u>	119
B.1.11	<u>res@id</u>	119
B.1.12	<u>upnp:class</u>	119

B.1.13	<u>upnp:searchClass</u>	121
B.1.14	<u>upnp:createClass</u>	121
B.1.15	<u>upnp:writeStatus</u>	122
B.2	Resource Encoding Characteristics Properties	123
B.2.1	<u>res</u>	123
B.3	Resource Encoding Extension Properties	128
B.3.1	<u>upnp:resExt</u>	128
B.3.2	<u>upnp:resExt::uniqueContentIdentification</u>	129
B.4	Contributor-related Properties	130
B.4.1	<u>upnp:artist</u>	130
B.4.2	<u>upnp:actor</u>	130
B.4.3	<u>upnp:author</u>	130
B.4.4	<u>upnp:producer</u>	131
B.4.5	<u>upnp:director</u>	131
B.4.6	<u>dc:publisher</u>	131
B.4.7	<u>dc:contributor</u>	131
B.5	Affiliation-related Properties	131
B.5.1	<u>upnp:genre</u>	131
B.5.2	<u>upnp:album</u>	132
B.5.3	<u>upnp:playlist</u>	132
B.6	Associated Resources Properties	132
B.6.1	<u>upnp:albumArtURI</u>	132
B.6.2	<u>upnp:artistDiscographyURI</u>	133
B.6.3	<u>upnp:lyricsURI</u>	133
B.6.4	<u>dc:relation</u>	133
B.7	Storage-Related Properties	133
B.7.1	<u>upnp:storageTotal</u>	133
B.7.2	<u>upnp:storageUsed</u>	133
B.7.3	<u>upnp:storageFree</u>	133
B.7.4	<u>upnp:storageMaxPartition</u>	134
B.7.5	<u>upnp:storageMedium</u>	134
B.8	General Description (mainly for UI purposes) Properties	135
B.8.1	<u>dc:description</u>	135
B.8.2	<u>upnp:longDescription</u>	135
B.8.3	<u>upnp:icon</u>	135
B.8.4	<u>upnp:region</u>	136
B.8.5	<u>upnp:rights</u>	136
B.8.6	<u>dc:date</u>	136
B.8.7	<u>dc:language</u>	136
B.8.8	<u>upnp:playbackCount</u>	136
B.8.9	<u>upnp:lastPlaybackTime</u>	137
B.8.10	<u>upnp:lastPlaybackPosition</u>	137
B.8.11	<u>upnp:recordedStartDateTime</u>	137
B.8.12	<u>upnp:recordedEndDateTime</u>	138
B.8.13	<u>upnp:recordedDuration</u>	138
B.8.14	<u>upnp:recordedDayOfWeek</u>	138
B.8.15	<u>upnp:srsRecordScheduleID</u>	139
B.8.16	<u>upnp:srsRecordTaskID</u>	139

B.8.17	<u>upnp:recordable</u>	139
B.9	Recorded Object-related Properties	140
B.9.1	<u>upnp:programTitle</u>	140
B.9.2	<u>upnp:seriesTitle</u>	140
B.9.3	<u>upnp:programID</u>	141
B.9.4	<u>upnp:seriesID</u>	141
B.9.5	<u>upnp:channelID</u>	141
B.9.6	<u>upnp:episodeType</u>	143
B.9.7	<u>upnp:episodeCount</u>	143
B.9.8	<u>upnp:episodeNumber</u>	143
B.9.9	<u>upnp:episodeSeason</u>	143
B.9.10	<u>upnp:programCode</u>	143
B.9.11	<u>upnp:rating</u>	144
B.9.12	<u>upnp:recommendationID</u>	145
B.10	User Channel and EPG Related Properties	145
B.10.1	<u>upnp:channelGroupName</u>	145
B.10.2	<u>upnp:callSign</u>	146
B.10.3	<u>upnp:networkAffiliation</u>	146
B.10.4	<u>upnp:serviceProvider</u>	146
B.10.5	<u>upnp:price</u>	147
B.10.6	<u>upnp:payPerView</u>	147
B.10.7	<u>upnp:epgProviderName</u>	147
B.10.8	<u>upnp:dateTimeRange</u>	147
B.11	Preserved Program Properties	148
B.11.1	<u>upnp:programPreserved</u>	148
B.11.2	<u>upnp:preservedTimeRange</u>	149
B.11.3	<u>upnp:programList</u>	151
B.12	Radio Broadcast Properties	151
B.12.1	<u>upnp:radioCallSign</u>	151
B.12.2	<u>upnp:radioStationID</u>	151
B.12.3	<u>upnp:radioBand</u>	152
B.13	Video Broadcast Properties	152
B.13.1	<u>upnp:channelNr</u>	152
B.13.2	<u>upnp:channelName</u>	153
B.13.3	<u>upnp:scheduledStartTime</u>	153
B.13.4	<u>upnp:scheduledEndTime</u>	153
B.13.5	<u>upnp:scheduledDuration</u>	154
B.14	Physical Tuner Status-related Properties	154
B.14.1	<u>upnp:signalStrength</u>	154
B.14.2	<u>upnp:signalLocked</u>	154
B.14.3	<u>upnp:tuned</u>	155
B.15	MultiStream-related Properties	156
B.15.1	<u>upnp:resExt::isSyncAnchor</u>	157
B.15.2	<u>upnp:resExt::componentInfo</u>	157
B.16	Segmentation-related Properties	163
B.16.1	<u>upnp:segmentID</u>	164
B.16.2	<u>upnp:resExt::segmentInfo</u>	164
B.17	Bookmark-related Properties	166

B.17.1	<u>@neverPlayable</u>	166
B.17.2	<u>upnp:bookmarkID</u>	167
B.17.3	<u>upnp:bookmarkedObjectID</u>	167
B.17.4	<u>upnp:deviceUDN</u>	167
B.17.5	<u>upnp:stateVariableCollection</u>	167
B.18	Miscellaneous Properties.....	168
B.18.1	<u>upnp:DVDRegionCode</u>	169
B.18.2	<u>upnp:originalTrackNumber</u>	169
B.18.3	<u>upnp:toc</u>	169
B.18.4	<u>upnp:userAnnotation</u>	169
B.18.5	<u>desc</u>	169
B.19	Object Tracking Properties.....	170
B.19.1	<u>upnp:containerUpdateID</u>	170
B.19.2	<u>upnp:objectUpdateID</u>	170
B.19.3	<u>upnp:totalDeletedChildCount</u>	171
B.19.4	<u>res@updateCount</u>	171
B.20	Permission Properties.....	172
B.20.1	<u>upnp:inclusionControl</u>	172
B.21	Ownership Properties.....	172
B.21.1	<u>upnp:objectOwner</u>	173
B.22	Object Linking Properties.....	174
B.22.1	<u>upnp:objectLink</u>	174
B.22.2	<u>upnp:objectLinkRef</u>	178
B.23	Foreign Metadata-related Properties.....	180
B.23.1	<u>upnp:foreignMetadata</u>	180
B.24	Synchronized Playback-related Properties.....	187
B.24.1	<u>upnp:resExt::clockSync</u>	187
B.25	DRMInfo-related Overview Properties.....	188
B.25.1	<u>upnp:resExt::DRMInfo</u>	188
Annex C	(normative) AV Working Committee Class Definitions.....	189
C.1	Class Hierarchy.....	189
C.1.1	Class name syntax.....	190
C.1.2	Class Properties Overview.....	191
C.2	<u>object</u> (Base Class).....	203
C.2.1	<u>item:object</u>	203
C.2.2	<u>container:object</u>	211
Annex D	(Informative) Theory of Operation.....	219
D.1	Introduction.....	219
D.2	Generating Object ID Values.....	219
D.3	Content Setup for Browsing and Searching.....	220
D.4	Browsing.....	220
D.4.1	Retrieving Sort Capabilities.....	221
D.4.2	Browsing the Root Level Metadata.....	221
D.4.3	Browsing the Children of the Root Level.....	222
D.4.4	Browsing the Children of the My Music Folder.....	223
D.4.5	Browsing the Children of the Singles Soundtrack Music Album.....	223
D.4.6	Browsing the Children of the Album Art Folder.....	224
D.5	Searching.....	225

D.5.1	Retrieving Search Capabilities.....	225
D.5.2	Search for All Content Created by the performer Sting	225
D.5.3	Search for all Photos Taken During the Month of October	227
D.5.4	Search for All Objects in the My Photos Folder Containing the Word "Christmas"	227
D.5.5	Search for all <i>album</i> objects in the ContentDirectory service	228
D.6	Browsing, Searching, and References.....	229
D.6.1	Creating a reference to a photo in the Mexico Trip album inside the Christmas album	229
D.6.2	Search for All Photos Taken During the Month of October.....	229
D.6.3	Deletion of the Reference to the Photo in the Mexico Trip Album.....	230
D.7	Object Creation	230
D.7.1	Creating a New Object	230
D.7.2	Creating a New MusicTrack.....	230
D.8	Object Resource Binding (Importing a Resource)	231
D.8.1	Transfer Using the <i>ImportResource()</i> Action	231
D.8.2	Transfer Using Direct HTTP POST	232
D.9	Exporting ContentDirectory Resources	232
D.9.1	Transfer Using the <i>ExportResource()</i> Action	233
D.9.2	Transfer using HTTP GET	234
D.10	Playlist Manipulation.....	234
D.10.1	Playlist File Representation in the ContentDirectory Service	234
D.10.2	Playlist File Generation	234
D.11	Internet Content Representation	236
D.12	Multi-component media representation	236
D.12.1	Creating a multi-component video object	238
D.12.2	Adding a component to a multi-component video object	240
D.13	Segments Manipulation.....	243
D.13.1	Segment Item Example	243
D.13.2	Creating, Destroying and Updating Segments	244
D.13.3	Browse and Search Segment Items	246
D.14	Bookmark Manipulation.....	246
D.14.1	<i>bookmarkItem</i> Example	246
D.14.2	Creating and Destroying Bookmarks.....	248
D.14.3	Browsing Bookmarks.....	252
D.15	Processing FreeForm Queries.....	257
D.15.1	Retrieving the title of all music albums.....	258
D.15.2	Retrieving the audio items of Album 1.....	258
D.15.3	Retrieving a limited number of photo items	259
D.16	Foreign Metadata	260
D.16.1	Determining the Supported Foreign Metadata Types.....	260
D.16.2	Determining Whether an Object Contains Foreign Metadata.....	261
D.17	Monitoring Changes.....	262
D.17.1	Monitoring Changes while <i>on-line</i>	263
D.17.2	Monitoring Changes while <i>off-line</i>	268
D.18	Browsing preserved transitory content.....	270
D.18.1	Browsing broadcast items with preserved history	270
D.18.2	Browsing program items indicating preserved history (EPG data available but not exposed to control point)	271

D.18.3 Browsing program items for recording (EPG exposed to control point).....	272
D.19 Object Linking	274
D.19.1 Displaying Object Link titles	274
D.19.2 Locating the head Object Link property of an Object Linked list.....	275
D.19.3 Starting an Object Linked presentation	275
D.19.4 Object Linking Example.....	275
D.20 <i>DEVICE_MODE</i> feature	282
D.20.1 Initiating and Managing <i>ActionBurst</i> mode	283
D.20.2 Initiating and Managing <i>ExclusiveOwnership</i> mode	287
D.21 Synchronized Playback.....	290
D.21.1 Precision Time Synchronized Playback for RTSP-RTP Transport	290
D.21.2 Precision Time Synchronized Playback for HTTP Transport	293
D.22 Usage of the <i>CONTAINER_SHORTCUTS</i> feature	294
Annex E (normative) EBNF Syntax Definitions	296
E.1 Summary.....	296
E.2 Date&time Syntax.....	296
Annex F (normative) <i>CDS</i> features	297
F.1 Requirements for the <i>EPG</i> feature, Version 1	298
F.2 Requirements for the <i>TUNER</i> feature, Version 1.....	299
F.3 Requirements for the <i>BOOKMARK</i> feature, Version 1	300
F.4 Requirements for the <i>FOREIGN_METADATA</i> feature, Version 1.....	300
F.5 Requirements for the <i>FFQ</i> feature, Version 1	301
F.6 Requirements for the <i>MULTI_STREAM</i> feature, Version 1	302
F.7 Requirements for the <i>SEGMENTATION</i> feature, Version 1	303
F.8 Requirements for the <i>DEVICE_MODE</i> feature, Version 1.....	304
F.9 Requirements for the <i>CLOCKSYNC</i> feature, Version 1	306
F.10 Requirements for the <i>CONTENT_PROTECTION</i> feature, Version 1	306
F.11 Requirements for the <i>CONTAINER_SHORTCUTS</i> feature, Version 1.....	307
Annex G (normative) <i>CONTENT_PROTECTION</i> feature	312
G.1 AV Roles for <i>CONTENT_PROTECTION</i>	312
G.1.1 Access at action level	313
G.1.2 Restrictable and Non-Restrictable Actions	313
G.1.3 <i>Action Level</i> Access using pre-defined AV Roles	314
G.1.4 Access at object level	315
G.1.5 <i>Role</i> assignments for unrecognized control points.....	316
G.1.6 Implicit role assignments.....	316
G.2 Behavior of actions with <i>CONTENT_PROTECTION</i> feature	317
G.2.1 <u>CreateObject()</u> action with <i>CONTENT_PROTECTION</i> feature	317
G.2.2 <u>UpdateObject()</u> action with <i>CONTENT_PROTECTION</i> feature.....	318
G.2.3 <u>Browse()</u> action with <i>CONTENT_PROTECTION</i> feature	319
G.2.4 <u>Search()</u> action with <i>CONTENT_PROTECTION</i> feature.....	319
G.2.5 <u>FreeFormQuery()</u> action with <i>CONTENT_PROTECTION</i> feature	320
G.2.6 <u>DestroyObject()</u> action with <i>CONTENT_PROTECTION</i> feature.....	320
G.2.7 <u>MoveObject()</u> action with <i>CONTENT_PROTECTION</i> feature	320
G.2.8 <u>DeleteResource()</u> action with <i>CONTENT_PROTECTION</i> feature	320
Annex H (informative) Content Authoring using Object Linking	321
H.1 Introduction	321
H.2 Object Linking Metadata Properties.....	321

H.3	Table of Contents (Index) Lists	322
H.4	Playback and Step Lists.....	323
H.5	References between lists of items.....	324
H.6	Sharing items in multiple lists.....	324
H.7	Return Model.....	325
H.8	Control Point processing of Object Linked items.....	325
Annex I (informative)	Example ContentDirectory Hierarchy	327
Annex J (informative)	Bibliography	334

List of Tables

Table 1 — EBNF Operators.....	7
Table 2 — CSV Examples.....	8
Table 3 — Namespace Definitions.....	9
Table 4 — Schema-related Information	11
Table 5 — Default Namespaces for the AV Specifications.....	13
Table 6 — Properties in XML	26
Table 7 — State variables.....	31
Table 8 — <u>SearchCapabilities</u> requirements for supporting <i>Tracking Changes Option</i>	32
Table 9 — Sort Modifiers	33
Table 10 — <u>ContainerUpdateIDs</u> Example.....	35
Table 11 — <u>ContainerUpdateIDs</u> Example.....	36
Table 12 — Event moderation	58
Table 13 — Actions.....	60
Table 14 — Arguments for <u>GetSearchCapabilities()</u>	61
Table 15 — Error Codes for <u>GetSearchCapabilities()</u>	61
Table 16 — Arguments for <u>GetSortCapabilities()</u>	61
Table 17 — Error Codes for <u>GetSortCapabilities()</u>	61
Table 18 — Arguments for <u>GetSortExtensionCapabilities()</u>	62
Table 19 — Error Codes for <u>GetSortExtensionCapabilities()</u>	62
Table 20 — Arguments for <u>GetFeatureList()</u>	62
Table 21 — Error Codes for <u>GetFeatureList()</u>	62
Table 22 — Arguments for <u>GetSystemUpdateID()</u>	62
Table 23 — Error Codes for <u>GetSystemUpdateID()</u>	63
Table 24 — Arguments for <u>GetServiceResetToken()</u>	63
Table 25 — Error Codes for <u>GetServiceResetToken()</u>	63
Table 26 — Arguments for <u>Browse()</u>	64
Table 27 — Error Codes for <u>Browse()</u>	65
Table 28 — Arguments for <u>Search()</u>	66
Table 29 — Error Codes for <u>Search()</u>	66
Table 30 — Arguments for <u>CreateObject()</u>	71
Table 31 — Error codes for <u>CreateObject()</u>	71
Table 32 — Arguments for <u>DestroyObject()</u>	73
Table 33 — Error Codes for <u>DestroyObject()</u>	73
Table 34 — Update examples	76
Table 35 — Arguments for <u>UpdateObject()</u>	78
Table 36 — Error Codes for <u>UpdateObject()</u>	78
Table 37 — Arguments for <u>MoveObject()</u>	80
Table 38 — Error Codes for <u>MoveObject()</u>	80
Table 39 — Arguments for <u>ImportResource()</u>	81
Table 40 — Error Codes for <u>ImportResource()</u>	81
Table 41 — Arguments for <u>ExportResource()</u>	82

Table 42 — Error Codes for <u>ExportResource()</u>	82
Table 43 — Arguments for <u>DeleteResource()</u>	83
Table 44 — Error Codes for <u>DeleteResource()</u>	83
Table 45 — Arguments for <u>StopTransferResource()</u>	83
Table 46 — Error Codes for <u>StopTransferResource()</u>	84
Table 47 — Arguments for <u>GetTransferProgress()</u>	84
Table 48 — Error Codes for <u>GetTransferProgress()</u>	84
Table 49 — Arguments for <u>CreateReference()</u>	85
Table 50 — Error Codes for <u>CreateReference()</u>	85
Table 51 — Arguments for <u>FreeFormQuery()</u>	86
Table 52 — Error Codes for <u>FreeFormQuery()</u>	87
Table 53 — Arguments for <u>GetFreeFormQueryCapabilities()</u>	87
Table 54 — Error Codes for <u>GetFreeFormQueryCapabilities()</u>	88
Table 55 — Arguments for <u>RequestDeviceMode()</u>	88
Table 56 — Error Codes for <u>RequestDeviceMode()</u>	89
Table 57 — Arguments for <u>ExtendDeviceMode()</u>	89
Table 58 — Error Codes for <u>ExtendDeviceMode()</u>	90
Table 59 — Arguments for <u>CancelDeviceMode()</u>	90
Table 60 — Error Codes for <u>CancelDeviceMode()</u>	91
Table 61 — Arguments for <u>GetDeviceMode()</u>	91
Table 62 — Error Codes for <u>GetDeviceMode()</u>	91
Table 63 — Arguments for <u>GetDeviceModeStatus()</u>	91
Table 64 — Error Codes for <u>GetDeviceModeStatus()</u>	92
Table 65 — Arguments for <u>GetPermissionsInfo()</u>	92
Table 66 — Error Codes for <u>GetPermissionsInfo()</u>	92
Table 67 — Common Error Codes	93
Table B.1 — ContentDirectory Service Properties Overview.....	108
Table B.2 — Base Properties Overview	117
Table B.3 — Allowed values for <u>upnp:class</u>	120
Table B.4 — Allowed values for <u>upnp:writeStatus</u>	122
Table B.5 — Resource Encoding Characteristics Properties Overview	123
Table B.6 — Allowed values for <u>res@daylightSaving</u>	128
Table B.7 — Resource Encoding Extension Properties Overview	128
Table B.9 — Contributor-related Properties Overview	130
Table B.10 — Affiliation-related Properties Overview	131
Table B.11 — Associated Resources Properties Overview	132
Table B.12 — Storage-Related Properties Overview	133
Table B.13 — General Description (mainly for UI purposes) Properties Overview	135
Table B.14 — Allowed values for <u>upnp:recordedDayOfWeek</u>	139
Table B.15 — Recorded Object-related Properties Overview.....	140
Table B.16 — User Channel and EPG Related Properties Overview.....	145
Table B.17 — Preserved Program Properties Overview	148
Table B.18 — Allowed values for <u>upnp:programPreserved</u>	148

Table B.19 — Radio Broadcast Properties Overview	151
Table B.20 — Allowed values for <u>upnp:radioBand</u>	152
Table B.21 — Video Broadcast Properties Overview	152
Table B.22 — Allowed values for <u>upnp:scheduledStartTime@usage</u>	153
Table B.23 — Physical Tuner Status-related Properties Overview	154
Table B.24 — MultiStream Properties Overview	156
Table B.25 — Allowed values for <u>upnp:resExt::componentInfo::componentGroup ::component::componentClass</u>	159
Table B.26 — Segmentation-related Properties Overview	163
Table B.27 — Bookmark-related Properties Overview	166
Table B.28 — Allowed values for <u>upnp:stateVariableCollection@rcsInstanceType</u>	168
Table B.29 — Miscellaneous Properties Overview	168
Table B.30 — Object Tracking Properties Overview	170
Table B.31 — Permission Properties Overview	172
Table B.32 — Ownership Properties Overview	172
Table B.33 — Object Linking Properties Overview	174
Table B.34 — Allowed values for <u>upnp:objectLink::mode</u>	176
Table B.35 — Allowed values for <u>upnp:objectLink::relatedInfo@role</u>	177
Table B.36 — Allowed values for <u>upnp:objectLink::endAction@action</u>	178
Table B.37 — Allowed values for <u>upnp:objectLinkRef::relatedInfo@role</u>	180
Table B.38 — Foreign Metadata-related Properties Overview	180
Table B.39 — Synchronized Playback Properties Overview	187
Table B.40 — DRMInfo-Related Properties Overview	188
Table C.1 — Class Properties Overview	191
Table C.2 — <u>object</u> Properties	203
Table C.3 — <u>item</u> Properties	203
Table C.4 — <u>imageItem:item</u> Properties	204
Table C.5 — <u>photo:imageItem</u> Properties	204
Table C.6 — <u>audioItem:item</u> Properties	204
Table C.7 — <u>musicTrack:audioItem</u> Properties	205
Table C.8 — <u>audioBroadcast:audioItem</u> Properties	205
Table C.9 — <u>audioBook:audioItem</u> Properties	205
Table C.10 — <u>videoItem:item</u> Properties	206
Table C.11 — <u>movie:videoItem</u> Properties	206
Table C.12 — <u>videoBroadcast:videoItem</u> Properties	207
Table C.13 — <u>musicVideoClip:videoItem</u> Properties	207
Table C.14 — <u>playlistItem:item</u> Properties	208
Table C.15 — <u>textItem:item</u> Properties	208
Table C.16 — <u>bookmarkItem:item</u> Properties	209
Table C.17 — <u>epgItem:item</u> Properties	210
Table C.18 — <u>audioProgram:epgItem</u> Properties	211
Table C.19 — <u>videoProgram:epgItem</u> Properties	211
Table C.20 — <u>container</u> Properties	212

Table C.21 — <u>person:container</u> Properties	212
Table C.22 — <u>musicArtist:person</u> Properties.....	212
Table C.23 — <u>playlistContainer:container</u> Properties	213
Table C.24 — <u>album:container</u> Properties	213
Table C.25 — <u>musicAlbum:album</u> Properties	214
Table C.26 — <u>photoAlbum:album</u> Properties	214
Table C.27 — <u>genre:container</u> Properties	214
Table C.28 — <u>channelGroup:container</u> Properties	215
Table C.29 — <u>epgContainer:container</u> Properties	216
Table C.30 — <u>storageSystem:container</u> Properties	217
Table C.31 — <u>storageVolume:container</u> Properties	217
Table C.32 — <u>storageFolder:container</u> Properties.....	218
Table C.33 — <u>genre:container</u> Properties	218
Table F.1 — <i>CDS features</i>	298
Table F.2 — Required characteristics of the <i>EPG feature</i> element	299
Table F.3 — Required characteristics of the <i>TUNER feature</i> element.....	299
Table F.4 — Required characteristics of the <i>BOOKMARK feature</i> element	300
Table F.5 — Required characteristics of the <i>FOREIGN_METADATA feature</i> element	301
Table F.6 — Required characteristics of the <i>FFQ feature</i> element.....	302
Table F.7 — Required characteristics of the <i>MULTI_STREAM feature</i> element.....	303
Table F.8 — Required characteristics of the <i>SEGMENTATION feature</i> element.....	304
Table F.9 — Required characteristics of the <i>DEVICE_MODE feature</i> element	305
Table F.10 — Required characteristics of the <i>CLOCKSYNC feature</i> element.....	306
Table F.11 — Required characteristics of the <i>CONTENT_PROTECTION feature</i> element	307
Table F.12 — Required characteristics of the <i>CONTAINER_SHORTCUTS feature</i> element	307
Table F.13 — Allowed values for the Shortcut Name element	308
Table G.1 — Pre-defined AV Roles and <i>Public</i>	312
Table G.2 — Error Codes for <i>Action Level Access</i>	313
Table G.3 — Pre-defined settings for <i>Restrictable and Non-Restrictable</i> AV Actions.....	314
Table G.4 — Pre-defined AV Action to AV Role permissions mapping	315
Table G.5 — Error Codes for <i>Object Level Access</i>	317
Table G.6 — Error Codes for <u>CreateObject()</u> and <u>UpdateObject()</u> action with <u>upnp:objectOwner</u> and <u>upnp:objectOwner</u> property	319
Table G.7 — Error Codes for <u>MoveObject()</u> action with <i>CONTENT_PROTECTION</i> <i>feature</i>	320
Table G.8 — Error Codes for <u>DeleteResource()</u> action with <i>CONTENT_PROTECTION</i> <i>feature</i>	320

List of Figures

Figure 1 — ContentDirectory Service Object Organization	21
Figure 2 — Flattened DIDL-Lite hierarchical structure	25
Figure C.1 — Class hierarchy for the item base class	189
Figure C.2 — Class hierarchy for the container base class	190
Figure D.1 — Example Handshaking for <i>DEVICE_MODE</i> feature	283
Figure H.1 — Example of Object Link “Index” list	323
Figure H.2 — Example of Object Link list reference	324

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see <http://www.iso.org/directives>).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of Standard, the meaning of the ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: [Foreword – Supplementary information](#)

ISO/IEC 29341-20-12 was prepared by UPnP Forum and adopted, under the PAS procedure, by joint technical committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

The list of all currently available parts of ISO/IEC 29341 series, under the general title *Information technology — UPnP Device Architecture*, can be found on the [ISO web site](#).

Introduction

ISO and IEC draw attention to the fact that it is claimed that compliance with this document may involve the use of patents as indicated below.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights. The holders of these patent rights have assured ISO and IEC that they are willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC.

Intel Corporation has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Intel Corporation
Standards Licensing Department
5200 NE Elam Young Parkway
MS: JFS-98
USA – Hillsboro, Oregon 97124

Microsoft Corporation has informed IEC and ISO that it has patent applications or granted patents as listed below:

6101499 / US; 6687755 / US; 6910068 / US; 7130895 / US; 6725281 / US; 7089307 / US;
7069312 / US; 10/783 524 /US

Information may be obtained from:

Microsoft Corporation
One Microsoft Way
USA – Redmond WA 98052

Philips International B.V. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Philips International B.V. – IP&S
High Tech campus, building 44 3A21
NL – 5656 Eindhoven

NXP B.V. (NL) has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

NXP B.V. (NL)
High Tech campus 60
NL – 5656 AG Eindhoven

Matsushita Electric Industrial Co. Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Matsushita Electric Industrial Co. Ltd.
1-3-7 Shiromi, Chuoh-ku
JP – Osaka 540-6139

Hewlett Packard Company has informed IEC and ISO that it has patent applications or granted patents as listed below:

5 956 487 / US; 6 170 007 / US; 6 139 177 / US; 6 529 936 / US; 6 470 339 / US; 6 571 388 / US; 6 205 466 / US

Information may be obtained from:

Hewlett Packard Company
1501 Page Mill Road
USA – Palo Alto, CA 94304

Samsung Electronics Co. Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Digital Media Business, Samsung Electronics Co. Ltd.
416 Maetan-3 Dong, Yeongtang-Gu,
KR – Suwon City 443-742

Huawei Technologies Co., Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Huawei Technologies Co., Ltd.
Administration Building, Bantian Longgang District
Shenzhen – China 518129

Qualcomm Incorporated has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Qualcomm Incorporated
5775 Morehouse Drive
San Diego, CA – USA 92121

Telecom Italia S.p.A. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Telecom Italia S.p.A.
Via Reiss Romoli, 274
Turin - Italy 10148

Cisco Systems informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA – USA 95134

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 29341-20-12:2017(E)

Original UPnP Document

Reference may be made in this document to original UPnP documents. These references are retained in order to maintain consistency between the specifications as published by ISO/IEC and by UPnP Implementers Corporation and later by UPnP Forum. The following table indicates the original UPnP document titles and the corresponding part of ISO/IEC 29341:

UPnP Document Title	ISO/IEC 29341 Part
UPnP Device Architecture 1.0	ISO/IEC 29341-1:2008
UPnP Device Architecture Version 1.0	ISO/IEC 29341-1:2011
UPnP Device Architecture 1.1	ISO/IEC 29341-1-1:2011
UPnP Device Architecture 2.0	ISO/IEC 29341-1-2
UPnP Basic:1 Device	ISO/IEC 29341-2
UPnP AV Architecture:1	ISO/IEC 29341-3-1:2008
UPnP AV Architecture:1	ISO/IEC 29341-3-1:2011
UPnP AVTransport:1 Service	ISO/IEC 29341-3-10
UPnP ConnectionManager:1 Service	ISO/IEC 29341-3-11
UPnP ContentDirectory:1 Service	ISO/IEC 29341-3-12
UPnP RenderingControl:1 Service	ISO/IEC 29341-3-13
UPnP MediaRenderer:1 Device	ISO/IEC 29341-3-2
UPnP MediaRenderer:2 Device	ISO/IEC 29341-3-2:2011
UPnP MediaServer:1 Device	ISO/IEC 29341-3-3
UPnP AVTransport:2 Service	ISO/IEC 29341-4-10:2008
UPnP AVTransport:2 Service	ISO/IEC 29341-4-10:2011
UPnP ConnectionManager:2 Service	ISO/IEC 29341-4-11:2008
UPnP ConnectionManager:2 Service	ISO/IEC 29341-4-11:2011
UPnP ContentDirectory:2 Service	ISO/IEC 29341-4-12
UPnP RenderingControl:2 Service	ISO/IEC 29341-4-13:2008
UPnP RenderingControl:2 Service	ISO/IEC 29341-4-13:2011
UPnP ScheduledRecording:1	ISO/IEC 29341-4-14
UPnP ScheduledRecording:2	ISO/IEC 29341-4-14:2011
UPnP MediaRenderer:2 Device	ISO/IEC 29341-4-2
UPnP MediaServer:2 Device	ISO/IEC 29341-4-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-4-4:2008
UPnP AV Datastructure Template:1	ISO/IEC 29341-4-4:2011
UPnP DigitalSecurityCamera:1 Device	ISO/IEC 29341-5-1
UPnP DigitalSecurityCameraMotionImage:1 Service	ISO/IEC 29341-5-10
UPnP DigitalSecurityCameraSettings:1 Service	ISO/IEC 29341-5-11
UPnP DigitalSecurityCameraStillImage:1 Service	ISO/IEC 29341-5-12
UPnP HVAC_System:1 Device	ISO/IEC 29341-6-1
UPnP ControlValve:1 Service	ISO/IEC 29341-6-10
UPnP HVAC_FanOperatingMode:1 Service	ISO/IEC 29341-6-11
UPnP FanSpeed:1 Service	ISO/IEC 29341-6-12
UPnP HouseStatus:1 Service	ISO/IEC 29341-6-13
UPnP HVAC_SetpointSchedule:1 Service	ISO/IEC 29341-6-14
UPnP TemperatureSensor:1 Service	ISO/IEC 29341-6-15
UPnP TemperatureSetpoint:1 Service	ISO/IEC 29341-6-16
UPnP HVAC_UserOperatingMode:1 Service	ISO/IEC 29341-6-17
UPnP HVAC_ZoneThermostat:1 Device	ISO/IEC 29341-6-2

UPnP BinaryLight:1 Device	ISO/IEC 29341-7-1
UPnP Dimming:1 Service	ISO/IEC 29341-7-10
UPnP SwitchPower:1 Service	ISO/IEC 29341-7-11
UPnP DimmableLight:1 Device	ISO/IEC 29341-7-2
UPnP InternetGatewayDevice:1 Device	ISO/IEC 29341-8-1
UPnP LANHostConfigManagement:1 Service	ISO/IEC 29341-8-10
UPnP Layer3Forwarding:1 Service	ISO/IEC 29341-8-11
UPnP LinkAuthentication:1 Service	ISO/IEC 29341-8-12
UPnP RadiusClient:1 Service	ISO/IEC 29341-8-13
UPnP WANCableLinkConfig:1 Service	ISO/IEC 29341-8-14
UPnP WANCommonInterfaceConfig:1 Service	ISO/IEC 29341-8-15
UPnP WANDSLLinkConfig:1 Service	ISO/IEC 29341-8-16
UPnP WANEthernetLinkConfig:1 Service	ISO/IEC 29341-8-17
UPnP WANIPConnection:1 Service	ISO/IEC 29341-8-18
UPnP WANPOTSLinkConfig:1 Service	ISO/IEC 29341-8-19
UPnP LANDevice:1 Device	ISO/IEC 29341-8-2
UPnP WANPPPConnection:1 Service	ISO/IEC 29341-8-20
UPnP WLANConfiguration:1 Service	ISO/IEC 29341-8-21
UPnP WANDevice:1 Device	ISO/IEC 29341-8-3
UPnP WANConnectionDevice:1 Device	ISO/IEC 29341-8-4
UPnP WLANAccessPointDevice:1 Device	ISO/IEC 29341-8-5
UPnP Printer:1 Device	ISO/IEC 29341-9-1
UPnP ExternalActivity:1 Service	ISO/IEC 29341-9-10
UPnP Feeder:1.0 Service	ISO/IEC 29341-9-11
UPnP PrintBasic:1 Service	ISO/IEC 29341-9-12
UPnP Scan:1 Service	ISO/IEC 29341-9-13
UPnP Scanner:1.0 Device	ISO/IEC 29341-9-2
UPnP QoS Architecture:1.0	ISO/IEC 29341-10-1
UPnP QosDevice:1 Service	ISO/IEC 29341-10-10
UPnP QosManager:1 Service	ISO/IEC 29341-10-11
UPnP QosPolicyHolder:1 Service	ISO/IEC 29341-10-12
UPnP QoS Architecture:2	ISO/IEC 29341-11-1
UPnP QosDevice:2 Service	ISO/IEC 29341-11-10
UPnP QosManager:2 Service	ISO/IEC 29341-11-11
UPnP QosPolicyHolder:2 Service	ISO/IEC 29341-11-12
UPnP QOS v2 Schema Files	ISO/IEC 29341-11-2
UPnP RemoteUIClientDevice:1 Device	ISO/IEC 29341-12-1
UPnP RemoteUIClient:1 Service	ISO/IEC 29341-12-10
UPnP RemoteUIServer:1 Service	ISO/IEC 29341-12-11
UPnP RemoteUIServerDevice:1 Device	ISO/IEC 29341-12-2
UPnP DeviceSecurity:1 Service	ISO/IEC 29341-13-10
UPnP SecurityConsole:1 Service	ISO/IEC 29341-13-11
UPnP ContentDirectory:3 Service	ISO/IEC 29341-14-12:2011
UPnP MediaServer:3 Device	ISO/IEC 29341-14-3:2011
UPnP ContentSync:1	ISO/IEC 29341-15-10:2011
UPnP Low Power Architecture:1	ISO/IEC 29341-16-1:2011
UPnP LowPowerProxy:1 Service	ISO/IEC 29341-16-10:2011

ISO/IEC 29341-20-12:2017(E)

UPnP LowPowerDevice:1 Service	ISO/IEC 29341-16-11:2011
UPnP QoS Architecture:3	ISO/IEC 29341-17-1:2011
UPnP QoSDevice:3 Service	ISO/IEC 29341-17-10:2011
UPnP QoSManager:3 Service	ISO/IEC 29341-17-11:2011
UPnP QoSPolicyHolder:3 Service	ISO/IEC 29341-17-12:2011
UPnP QoSDevice:3 Addendum	ISO/IEC 29341-17-13:2011
UPnP RemoteAccessArchitecture:1	ISO/IEC 29341-18-1:2011
UPnP InboundConnectionConfig:1 Service	ISO/IEC 29341-18-10:2011
UPnP RADAConfig:1 Service	ISO/IEC 29341-18-11:2011
UPnP RADASync:1 Service	ISO/IEC 29341-18-12:2011
UPnP RATAConfig:1 Service	ISO/IEC 29341-18-13:2011
UPnP RAClient:1 Device	ISO/IEC 29341-18-2:2011
UPnP RAServer:1 Device	ISO/IEC 29341-18-3:2011
UPnP RADiscoveryAgent:1 Device	ISO/IEC 29341-18-4:2011
UPnP SolarProtectionBlind:1 Device	ISO/IEC 29341-19-1:2011
UPnP TwoWayMotionMotor:1 Service	ISO/IEC 29341-19-10:2011
UPnP AV Architecture:2	ISO/IEC 29341-20-1
UPnP AVTransport:3 Service	ISO/IEC 29341-20-10
UPnP ConnectionManager:3 Service	ISO/IEC 29341-20-11
UPnP ContentDirectory:4 Device	ISO/IEC 29341-20-12
UPnP RenderingControl:3 Service	ISO/IEC 29341-20-13
UPnP ScheduledRecording:2 Service	ISO/IEC 29341-20-14
UPnP MediaRenderer:3 Service	ISO/IEC 29341-20-2
UPnP MediaServer:4 Device	ISO/IEC 29341-20-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-20-4
UPnP InternetGatewayDevice:2 Device	ISO/IEC 29341-24-1
UPnP WANIPConnection:2 Service	ISO/IEC 29341-24-10
UPnP WANIPv6FirewallControl:1 Service	ISO/IEC 29341-24-11
UPnP WANConnectionDevice:2 Service	ISO/IEC 29341-24-2
UPnP WANDevice:2 Device	ISO/IEC 29341-24-3
UPnP Telephony Architecture:2	ISO/IEC 29341-26-1
UPnP CallManagement:2 Service	ISO/IEC 29341-26-10
UPnP MediaManagement:2 Service	ISO/IEC 29341-26-11
UPnP Messaging:2 Service	ISO/IEC 29341-26-12
UPnP PhoneManagement:2 Service	ISO/IEC 29341-26-13
UPnP AddressBook:1 Service	ISO/IEC 29341-26-14
UPnP Calendar:1 Service	ISO/IEC 29341-26-15
UPnP Presense:1 Service	ISO/IEC 29341-26-16
UPnP TelephonyClient:2 Device	ISO/IEC 29341-26-2
UPnP TelephonyServer:2 Device	ISO/IEC 29341-26-3
UPnP Friendly Info Update:1 Service	ISO/IEC 29341-27-1
UPnP MultiScreen MultiScreen Architecture:1	ISO/IEC 29341-28-1
UPnP MultiScreen Application Management:1 Service	ISO/IEC 29341-28-10
UPnP MultiScreen Screen:1 Device	ISO/IEC 29341-28-2
UPnP MultiScreen Application Management:2 Service	ISO/IEC 29341-29-10
UPnP MultiScreen Screen:2 Device	ISO/IEC 29341-29-2
UPnP IoT Management and Control Architecture Overview:1	ISO/IEC 29341-30-1

UPnP DataStore:1 Service	ISO/IEC 29341-30-10
UPnP IoT Management and Control Data Model:1 Service	ISO/IEC 29341-30-11
UPnP IoT Management and Control Transport Generic:1 Service	ISO/IEC 29341-30-12
UPnP IoT Management and Control:1 Device	ISO/IEC 29341-30-2
UPnP Energy Management:1 Service	ISO/IEC 29341-31-1

1 Scope

This document specifies the characteristics of the UPnP networked service named ContentDirectory, version 4. This service definition is compliant with UPnP Device Architecture 1.0 [14].

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 needs to 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 let 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 enables 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.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[1] – *XML Schema for RenderingControl AllowedTransformSettings*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/schemas/av/AllowedTransformSettings-v1-20130331.xsd>.

Latest version available at: <http://www.upnp.org/schemas/av/AllowedTransformSettings.xsd>.

[2] – *AV Datastructure Template:1*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/specs/av/UPnP-av-AVDataStructureTemplate-v1-20130331.pdf>.

Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-AVDataStructureTemplate-v1.pdf>.

[3] – *XML Schema for UPnP AV Common XML Data Types*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/schemas/av/av-v3-20130331.xsd>.

Latest version available at: <http://www.upnp.org/schemas/av/av.xsd>.

[4] – *XML Schema for UPnP AV Common XML Structures*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/schemas/av/avs-v3-20130331.xsd>.

Latest version available at: <http://www.upnp.org/schemas/av/avs.xsd>.

[5] – *AVTransport:3*, UPnP Forum, March 31, 2013.

Available at: <http://www.upnp.org/specs/av/UPnP-av-AVTransport-v3-Service-20130331.pdf>.

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

[6] – *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>.

[7] – *ContentDirectory:4*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v4-Service-20130331.pdf>.
Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-ContentDirectory-v4-Service.pdf>.

[8] – *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>.

[9] – *ConnectionManager:3*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v3-Service-20130331.pdf>.
Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-ConnectionManager-v3-Service.pdf>.

[10] – *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>.

[11] – *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>.

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

[13] – *DCMI term declarations represented in XML schema language*.
Available at: <http://www.dublincore.org/schemas/xmls>.

[14] – *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>.

[15] – *XML Schema for ContentDirectory Structure and Metadata (DIDL-Lite)*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/didl-lite-v3-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/didl-lite.xsd>.

[16] – *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>.

[17] – *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>.

[18] – *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>.

- [19] – ISO/IEC 14977, *Information technology - Syntactic metalanguage - Extended BNF*, December 1996.
- [20] – *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>.
- [21] – *RenderingControl:3*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v3-Service-20130331.pdf>.
Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-RenderingControl-v3-Service.pdf>.
- [22] – *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>.
- [23] – *XML Schema for ConnectionManager RendererInfo*, UPnP Forum, December 31, 2010.
Available at: <http://www.upnp.org/schemas/av/rri-v1-20101231.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/rri.xsd>.
- [24] – *XML Schema for AVTransport PlaylistInfo*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/rpl-v1-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/rpl.xsd>.
- [25] – *ScheduledRecording:2*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v2-Service-20130331.pdf>.
Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-ScheduledRecording-v2-Service.pdf>.
- [26] – *XML Schema for ScheduledRecording Metadata and Structure*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/srs-v2-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/srs.xsd>.
- [27] – *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>.
- [28] – *XML Schema for RenderingControl TransformSettings*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/TransformSettings-v1-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/TransformSettings.xsd>.
- [29] – *XML Schema for ContentDirectory Metadata*, UPnP Forum, March 31, 2013.
Available at: <http://www.upnp.org/schemas/av/upnp-v4-20130331.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/upnp.xsd>.
- [30] – *The “xml:” Namespace*, November 3, 2004.
Available at: <http://www.w3.org/XML/1998/namespace>.
- [31] – *XML Schema for the “xml:” Namespace*.
Available at: <http://www.w3.org/2001/xml.xsd>.
- [32] – *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>.
- [33] – *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>.

[34] – *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>.

[35] – *XML Schema for XML Schema*.
Available at: <http://www.w3.org/2001/XMLSchema.xsd>.

[36] – *DeviceProtection:1*, UPnP Forum, February 24, 2011.
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>.

[37] – *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>.

[38] – *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>.

[39] – *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>.

[40] – *IETF RFC 3986, Uniform Resource Identifiers (URI): Generic Syntax*, January 2005.
Available at: <http://www.ietf.org/rfc/rfc3986.txt>.

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

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

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

[44] – *IETF RFC 1321, The MD5 Message-Digest Algorithm*, R. Rivest, April 1992.
Available at: <http://tools.ietf.org/html/rfc1321>.

[45] – *IETF RFC 3174, US Secure Hash Algorithm 1 (SHA1)*, D. Eastlake et al, September 2001.
Available at: <http://tools.ietf.org/html/rfc3174>.

[46] – *IETF RFC 4078, The TV-Anytime Content Reference Identifier (CRID)*, N. Earnshaw et al, May 2005.
Available at: <http://www.ietf.org/rfc/rfc4078.txt>.

[47] – *Data elements and interchange formats – Information interchange -- Representation of dates and times*, International Standards Organization, December 21, 2000.
Available at: [ISO 8601:2000](http://www.iso.org/iso/8601).

[48] – *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>.

[49] – *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>.

3 Terms, definitions, symbols and abbreviations

For the purposes of this document, the terms and definitions given in [14] and the following subclauses 3.1 and 3.2 apply.

3.1 Provisioning terms

3.1.1

allowed

A

The definition or behavior is allowed.

3.1.2

conditionally allowed

CA

The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is allowed, otherwise it is not allowed.

3.1.3

conditionally required

CR

The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is required. Otherwise the definition or behavior is allowed as default unless specifically defined as not allowed.

3.1.4

required

R

The definition or behavior is required.

3.1.5

R/A

Used in a table column heading to indicate that each abbreviated entry in the column declares the provisioning status of the item named in the entry's row.

3.1.6

X

Vendor-defined, non-standard.

3.1.7

-D

Declares that the item referred to is deprecated, when it is appended to any of the other abbreviated provisioning terms.

3.1.8

CSV list (or CSV)

Comma separated value list. List—or one-dimensional array—of values contained in a string and separated by commas

3.2 Symbols

3.2.1

::

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.

4 Notations and Conventions

4.1 Notation

- UPnP interface names defined in the UPnP Device Architecture specification [14] are styled in **green bold underlined** text.

- UPnP interface names defined outside of the UPnP Device Architecture specification [14] are styled in *red italic underlined* text.
- Some additional non-interface names and terms are styled in *italic* text.
- Words that are emphasized are also styled in *italic* text. The difference between italic terms and italics for emphasis will be apparent by context.
- Strings that are to be taken literally are enclosed in “double quotes”.

4.1.1 Data Types

Data type definitions come from three sources:

- All state variable and action argument data types are defined in [14].
- Basic data types for properties are defined in [34].
- Additional data types for properties are defined in the XML schema(s) (see [3]) associated with this service.

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 shall 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 shall be accepted. Nevertheless, it is strongly recommended that all Boolean properties be represented as “0” and “1”.

4.1.2 Strings Embedded in Other Strings

Some string variables and arguments described in this document contain substrings that shall be independently identifiable and extractable for other processing. This requires the definition of appropriate substring delimiters and an escaping mechanism so that these delimiters can also appear as ordinary characters in the string and/or its independent substrings. This document uses embedded strings in two contexts – Comma Separated Value (CSV) lists (see subclause 4.2.2) 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

4.1.3 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 [19].

4.1.3.1 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 shall 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 in Table 1:

Table 1 — EBNF Operators

Operator	Semantics
<code>::=</code>	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.
<code> </code>	alternative separator – separates sequences on the right that are independently allowed definitions for the non-terminal on the left.
<code>*</code>	null repetition – means the expression to its left may occur zero or more times.
<code>+</code>	non-null repetition – means the expression to its left shall occur at least once and may occur more times.
<code>[]</code>	optional – the expression between the brackets is allowed.
<code>()</code>	grouping – groups the expressions between the parentheses.
<code>-</code>	character range – represents all characters between the left and right character operands inclusively.

4.2 Derived Data Types

4.2.1 Summary

Subclause 4.2 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 in subclause 4.2.2 applies to both string data types.

4.2.2 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 [14], 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 (all values can be of different types). 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 [14] (that is: allowed leading sign, allowed 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 [14]: [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 [34]: 0, false, 1, true.
- Escaping conventions for the comma and backslash characters are defined in 4.1.2.
- 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 2 — CSV Examples

Type refinement of string	Value	Comments
CSV (<u>string</u>) or CSV (xsd:string)	+artist,-date	List of 2 property sort criteria.
CSV (<u>int</u>) or CSV (xsd:integer)	1,-5,006,0,+7	List of 5 integers.
CSV (<u>boolean</u>) or CSV (xsd:Boolean)	0,1,1,0	List of 4 booleans
CSV (<u>string</u>) or CSV (xsd:string)	Smith\, Fred,Jones\, Davey	List of 2 names, “Smith, Fred” and “Jones, Davey”
CSV (<u>i4</u> , <u>string</u> , <u>ui2</u>) 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 (<u>i4</u>) or CSV (xsd:int)	3, 4	Illegal CSV. White space is not allowed as part of an integer value.
CSV (<u>string</u>) 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 <u>string</u> , a department <u>string</u> and years-of-service <u>ui2</u> or a name xsd:string, a department xsd:string and years-of-service xsd:unsignedShort.

4.3 Management of XML Namespaces in Standardized DCPs

UPnP specifications make extensive use of XML namespaces. This enables 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 shall 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 shall be locally unique to the document. Any valid XML no-colon-name may be used. And, in formal XML usage, different XML documents may use different namespace prefixes to refer to the same namespace. The creation and use of the namespace prefix was standardized by the W3C

XML Committee in [32] 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, shall use the predefined namespace prefixes when referring to CDS properties (“upnp:”, “dc:”, etc).

All of the namespaces used in this specification are listed in Table 3 and Table 4. For each such namespace, Table 3 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 ScheduledRecording 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 4 to cross-reference additional namespace information. Table 4 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 Clause 2 for its associated schema.

The normative definitions for these namespaces are the documents referenced in Table 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 3 — Namespace Definitions

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
<i>AV Working Committee defined namespaces</i>			
atrs	urn:schemas-upnp-org:av:AllowedTransformSettings	AllowedTransformSettings and AllowedDefaultTransformSettings state variables for RenderingControl	[21]
av	urn:schemas-upnp-org:av:av	Common data types for use in AV schemas	[3]
avdt	urn:schemas-upnp-org:av:avdt	Datastructure Template	[2]
avs	urn:schemas-upnp-org:av:avs	Common structures for use in AV schemas	[4]
avt-event	urn:schemas-upnp-org:metadata-1-0/AVT/	Evented LastChange state variable for AVTransport	[5]

Standard Name-space Prefix	Namespace Name	Namespace Description	Normative Definition Document Reference
cds-event	urn:schemas-upnp-org:av:cds-event	Evented LastChange state variable for ContentDirectory	[7]
cm-dciu	urn:schemas-upnp-org:av:cm-deviceClockInfoUpdates	Evented DeviceClockInfoUpdates state variable for ConnectionManager	[9]
cm-ftrlst	urn:schemas-upnp-org:av:cm-featureList	FeatureList state variable for ConnectionManager	[9]
didl-lite	urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/	Structure and metadata for ContentDirectory	[7]
dmo	urn:schemas-upnp.org:av:dmo	Evented DeviceMode state variable for ContentDirectory	[7]
dmor	urn:schemas-upnp.org:av:dmor	A_ARG_TYPE_DeviceModeRequest state variable for ContentDirectory	[7]
dmos	urn:schemas-upnp.org:av:dmos	DeviceModeStatus state variable for ContentDirectory	[7]
pi	urn:schemas-upnp.org:av:pi	PermissionsInfo state variable for ContentDirectory	[7]
rds-event	urn:schemas-upnp-org:metadata-1-0/RCS/	Evented LastChange state variable for RenderingControl	[21]
rii	urn:schemas-upnp-org:av:rii	A_ARG_TYPE_RenderingInfoList state variable for ConnectionManager	[9]
rpl	urn:schemas-upnp-org:av:rpl	A_ARG_TYPE_PlaylistInfo state variable for AVTransport	[5]
srs	urn:schemas-upnp-org:av:srs	Metadata and structure for ScheduledRecording	[25]
srs-event	urn:schemas-upnp-org:av:srs-event	Evented LastChange state variable for ScheduledRecording	[25]
trs	urn:schemas-upnp-org:av:TransformSettings	TransformSettings and DefaultTransformSettings state variables for RenderingControl	[21]
upnp	urn:schemas-upnp-org:metadata-1-0/upnp/	Metadata for ContentDirectory	[7]
<i>Externally defined namespaces</i>			
dc	http://purl.org/dc/elements/1.1/	Dublin Core	[13]
xsd	http://www.w3.org/2001/XMLSchema	XML Schema Language 1.0	[33], [34]
xsi	http://www.w3.org/2001/XMLSchema-instance	XML Schema Instance Document schema	[33] 2.6 & 3.2.7
xml	http://www.w3.org/XML/1998/namespace	The "xml:" Namespace	[30]

Table 4 — Schema-related Information

Standard Name-space Prefix	Relative URI and File Name ^a • Form 1, Form 2, Form3	Valid Root Element(s)	Schema Reference
<i>AV Working Committee Defined Namespaces</i>			
atrs	AllowedTransformSetting s-vn-yyyymmdd.xsd AllowedTransformSetting s-vn.xsd AllowedTransformSetting s.xsd	<TransformList>	[1]
av	av-vn-yyyymmdd.xsd av-vn.xsd av.xsd	n/a	[3]
avdt	avdt-vn-yyyymmdd.xsd avdt-vn.xsd avdt.xsd	<AVDT>	[2]
avs	avs-vn-yyyymmdd.xsd avs-vn.xsd avs.xsd	<Capabilities> <Features> <stateVariableValuePairs>	[4]
avt-event	avt-event-vn- yyyymmdd.xsd avt-event-vn.xsd avt-event.xsd	<Event>	[6]
cds-event	cds-event-vn- yyyymmdd.xsd cds-event-vn.xsd cds-event.xsd	<StateEvent>	[8]
cm-dciu	cm- deviceClockInfoUpdates- vn-yyyymmdd.xsd cm- deviceClockInfoUpdates -vn.xsd cm- deviceClockInfoUpdates. xsd	<DeviceClockInfoUpdates>	[10]
cm-ftrlst	cm-featureList-vn- yyyymmdd.xsd cm-featureList-vn.xsd cm-featureList.xsd	<Features>	[11]
didl-lite	didl-lite-vn- yyyymmdd.xsd didl-lite-vn.xsd didl-lite.xsd	<DIDL-Lite>	[15]
dmo	dmo-vn-yyyymmdd.xsd dmo-vn.xsd dmo.xsd	<DeviceMode>	[16]
dmor	dmor-vn-yyyymmdd.xsd dmor-vn.xsd dmor.xsd	<DeviceModeRequest>	[17]
dmos	dmos-vn-yyyymmdd.xsd dmos-vn.xsd dmos.xsd	<DeviceModeStatus>	[18]

Standard Name-space Prefix	Relative URI and File Name ^a • Form 1, Form 2, Form3	Valid Root Element(s)	Schema Reference
pi	pi-vn-yyyymmdd.xsd pi-vn.xsd pi.xsd	<PermissionsInfo>	[20]
rce-event	rce-event-vn-yyyymmdd.xsd rce-event-vn.xsd rce-event.xsd	<Event>	[22]
rii	rii-vn-yyyymmdd.xsd rii-vn.xsd rii.xsd	<rendererInfo>	[23]
rpl	rpl-vn-yyyymmdd.xsd rpl-vn.xsd rpl.xsd	<PlaylistInfo>	[24]
trs	TransformSettings-vn-yyyymmdd.xsd TransformSettings-vn.xsd TransformSettings.xsd	<TransformSettings>	[28]
srs	srs-vn-yyyymmdd.xsd srs-vn.xsd srs.xsd	<srs>	[26]
srs-event	srs-event-vn-yyyymmdd.xsd srs-event-vn.xsd srs-event.xsd	<StateEvent>	[27]
upnp	upnp-vn-yyyymmdd.xsd upnp-vn.xsd upnp.xsd	n/a	[29]
Externally Defined Namespaces			
dc	Absolute URL: http://dublincore.org/schemas/xmls/simpledc20021212.xsd		[12]
xsd	n/a	<schema>	[35]
xsi	n/a		n/a
xml	n/a		[31]
^a Absolute URIs are generated by prefixing the relative URIs with " http://www.upnp.org/schemas/av/ "			

4.3.1 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 shall use the standard namespace prefixes as declared in Table 3. In order to properly process the XML documents described herein, control points and devices shall use namespace-aware XML processors [32] for both reading and writing. As allowed by [32], 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 shall 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. However, 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 5.

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

Table 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	rct-event
ScheduledRecording	srs

4.3.2 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 complies with one or more specific XML schemas, which define XML namespaces. Each namespace is uniquely identified by an assigned namespace name. The namespace names that are defined by the AV Working Committee are URNs. See Table 3 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 qualify and disambiguate element and attribute names that are defined within different namespaces. The schemas that correspond to the listed namespaces are identified by URI values that are listed in the `schemaLocation` attribute also within the root element (see subclause 4.3.3).

In order to enable both forward and backward compatibility, namespace names are permanently assigned and shall not change even when a new version of a specification changes the definition of a namespace. However, all changes to a namespace definition shall be backward-compatible. In other words, the updated definition of a namespace shall not invalidate any XML documents that comply with an earlier definition of that same namespace. This means, for example, that a namespace shall not be changed so that a new element or attribute becomes required in a conforming instance document. Although namespace names shall 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 shall 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 4 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 namespace 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, shall use Form 3.
- All UPnP AV published schemas that reference other UPnP AV schemas shall 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 shall contain a version attribute in the <schema> root element. Its value shall 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".

4.3.3 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>
```

4.4 Vendor-defined Extensions

Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation shall follow the naming conventions and XML rules as specified below in subclauses 4.4.1 to 4.4.4.

4.4.1 Vendor-defined Action Names

Vendor-defined action names shall begin with “X”. Additionally, it should be followed by an ICANN assigned domain name owned by the vendor followed by the underscore character (“_”). It shall then be followed by the vendor-assigned action name. The vendor-assigned action name shall 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 shall 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 shall 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 shall not be “XML” in any combination of case.

4.4.2 Vendor-defined State Variable Names

Vendor-defined state variable names shall begin with “X”. Additionally, it should be followed by an ICANN assigned domain name owned by the vendor, followed by the underscore character (“_”). It shall then be followed by the vendor-assigned state variable name. The vendor-assigned state variable name shall not contain a hyphen character (“-”, 2D Hex in UTF-8). Vendor-assigned action names are case sensitive. The first character of the name shall 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 shall 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 shall not be "XML" in any combination of case.

4.4.3 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 shall be scoped by a vendor-owned XML namespace. Arbitrary XML shall be enclosed in an element that begins with "X," and this element shall 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".

4.4.4 Vendor-defined Property Names

UPnP vendors may add non-standard properties to the ContentDirectory service. Each property addition shall be scoped by a vendor-owned namespace. The vendor-assigned property name shall not contain a hyphen character ("-", 2D Hex in UTF-8). Vendor-assigned property names are case sensitive. The first character of the name shall 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 shall 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 shall not be "XML" in any combination of case.

5 Service Modeling Definitions

5.1 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.

5.2 Key Concepts

5.2.1 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 1.2.2 of the UPnP Device Architecture specification [14] 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 1.2.2 of the UPnP Device Architecture specification [14] 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 1.2.2 of the UPnP Device Architecture specification [14].

5.2.2 object

A ContentDirectory object is a structured set of metadata properties representing entertainment content that might 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 subclauses 5.5.8 and 5.5.9 for examples. See subclause 5.3.15 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 5.2.20 for details.

For identification purposes, the ContentDirectory service shall 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 5.2.3 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 5.2.6 for details.

5.2.3 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 can 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 5.2.1). 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 might have

changed). A ContentDirectory service implementation shall enforce the above rule for object [@id](#) reuse during periods when the [ServiceResetToken](#) state variable is constant. See 5.3.7 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 D.2 for examples of mechanisms to generate persistent and unique [@id](#) property values.

5.2.4 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 5.2.3, 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 D.2 for details.

5.2.5 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](#), [@childContainerCount](#) or [upnp:totalDeletedChildCount](#) properties). However, a change to any property belonging to any of the container's child objects shall not be treated as an *Object Modification* of the container. See 5.2.9 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 shall not be treated as an *Object Modification*. For example, the [SystemUpdateID](#) state variable and all [upnp:containerUpdateID](#) properties (defined below) shall not be incremented when either of these properties are modified.

5.2.6 class

A class is used to assign a type to an object. It also identifies the minimum required set of properties that shall be included in the object's metadata and the allowed properties that may be included. Classes are organized in a hierarchy with certain classes being derived from others as in a typical object-oriented system. At the root of the class hierarchy is the [object](#) base class. Examples are [object.item.audioItem.musicTrack](#) and [object.container.album.musicAlbum](#). See C.1.1 for a definition of the format of the class specification for an object.

5.2.7 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.

5.2.8 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 that a ContentDirectory service implementation is required to maintain a [ContainerUpdateIDValue](#) indicator for each of its containers. See 5.2.11 for details.

5.2.9 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 5.2.8) 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 5.2.5 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 5.2.5.

Note: The exceptions listed under the definition of object modification also apply to a container modification. Refer to the exceptions listed in 5.2.5.

5.2.10 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 shall:

- Support the LastChange state variable as defined in 5.3.8.
- 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 shall be implemented for all items for which the ContentDirectory service implementation is tracking changes:

- upnp:objectUpdateID
- res@updateCount

The following metadata properties shall 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 shall not support the LastChange state variable and the following metadata properties shall 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.

5.2.11 ContainerUpdateIDValue Indicator

The ContainerUpdateIDValue indicator is an internal, unsigned integer that shall 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 allowed 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) shall be the same as the property value defined in B.19.1.

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 5.2.9 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 shall invoke the *Service Reset Procedure* as defined in 5.3.7.1. 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.

5.2.12 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. Figure 1 illustrates the concepts. (The figure represents a hypothetical ContentDirectory service content structure.)

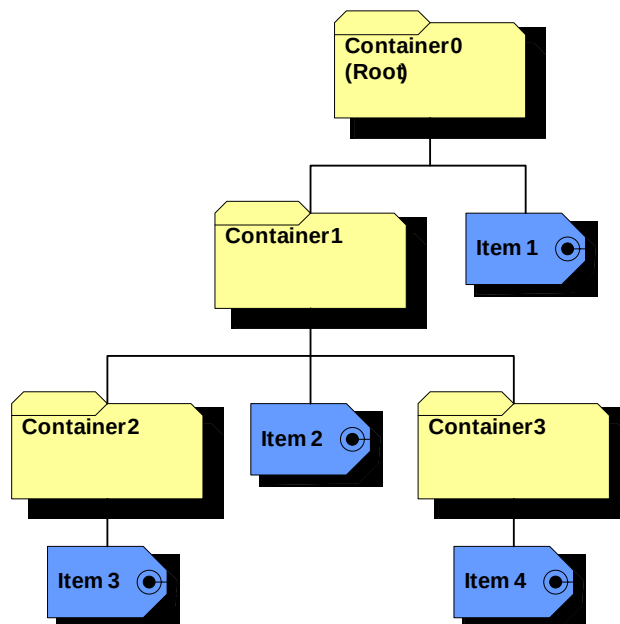


Figure 1 — ContentDirectory Service Object Organization

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

5.2.13 Hierarchical location

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

container is said to have a parent relationship with the object. The object is called a *child (object)* of the container (one level down) and the object is said to have a child relationship with that container.

A child object can be either an item object or a container object. In Figure 1, Item 1 is a child of Container 0 (the Root container). Likewise, Item 3 is a child of Container 2. Any object can only have one parent container, except for the root container, which has none (indicated by setting its [@parentID](#) property value to “-1”). A parent container can have multiple child objects. In Figure 1, Container 1 is the parent of Container 2, Item 2, and Container 3.

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

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

See B.1.1 and B.1.2 for additional information.

5.2.14 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.

5.2.15 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 can be created, modified, and/or deleted in a short period of time.

As described in 5.3.8, each object update triggers an event that might 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, can 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 enables 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 enables 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 could 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 shall be obeyed:

- Each subtree update shall 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 can be the ContentDirectory service root container (*@id*="0").
- All objects represented by an object modification event with the *stUpdate* attribute set to one (“1”) shall be a descendant of one and only one designated subtree root. In other words, the root of an active subtree update operation shall not be a descendant of the root of another active subtree update operation.

5.2.16 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 4.

For example, the phrase *DIDL-Lite XML Document* refers to a valid XML 1.0 document according to the DIDL-Lite schema [15]. Such a document comprises a single <DIDL-Lite ...> root element, and it is allowed to be 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>”

5.2.17 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 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>"

5.2.18 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 [15].

Such a document comprises a single <DIDL-Lite ...> root element, and it is allowed to be 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 intrinsically 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 shall not be embedded in <container> elements.

Figure 2 illustrates this flattened view. To the right, the corresponding (incomplete, simplified) *DIDL-Lite XML Document* is also included.

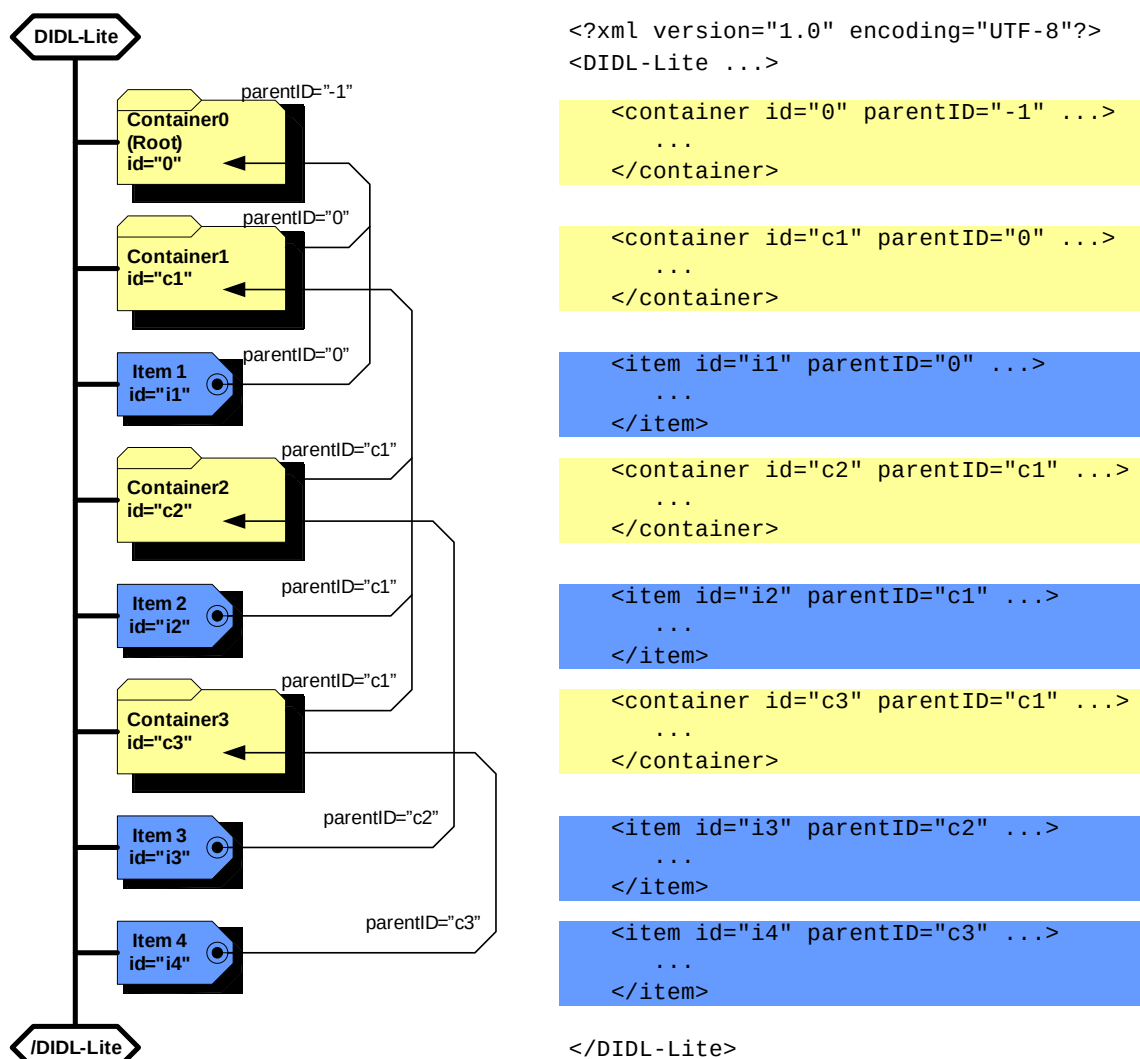


Figure 2 — Flattened DIDL-Lite hierarchical structure

Note: The information about an object in a *DIDL-Lite XML Document* is not necessarily sufficient to determine its hierarchical location in the ContentDirectory tree.

Indeed, to determine the correct hierarchical location, 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, it is 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.

5.2.19 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*).

5.2.19.1 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.

5.2.20 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 could 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 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, could have independent child properties of their own. To fully qualify a nested property, the fully qualified parent name is used, followed by “::”, followed by the child property name, recursively. For example, xxx::yyy::zzz where zzz is the child of yyy and yyy is the child of xxx. The data types and meanings for all properties are defined in Annex B.

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 6.

Table 6 — Properties in XML

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