
AVTransport:3 Service

For UPnP Version 1.0

Status: Standardized DCP (SDCP)

Date: March 31, 2013

Service Template Version 3.0

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

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

© 2014, UPnP Forum. All rights Reserved.

| Authors | Company |
|--|---------------------------------|
| Alan Presser | Allegrosoft |
| Wouter van der Beek | Cisco Systems |
| Gary Langille | Echostar |
| Gerrie Shults | HP |
| Raj Bopardikar | Intel |
| Nelson Kidd | Intel |
| John Ritchie | Intel |
| Mark Walker | Intel |
| Keith Miller | Intel |
| Sungjoon Ahn | LG Electronics |
| Changhyun Kim | LG Electronics |
| Seung R. Yang (Co-Chair) | LG Electronics |
| Masatomo Hori | Matsushita Electric (Panasonic) |
| Matthew Ma | Matsushita Electric (Panasonic) |
| Anders Klemets | Microsoft |
| Jack Unverferth | Microsoft |
| Keith Miller (Co-Chair) | Nokia |
| Christian Gran | PacketVideo |
| Wouter van der Beek | Philips |
| Wim Bronnenberg | Philips |
| Jeffrey Kang | Philips |
| Geert Knapen | Philips |
| Russell Berkoff | Pioneer |
| Irene Shen | Pioneer |
| Russell Berkoff (vice-Chair) | Samsung |
| Richard Bardini | Sony |
| Norifumi Kikkawa | Sony |
| Jonathan Tourzan | Sony |
| Yasuhiro Morioka | Toshiba |
| Nicholas Frame | TP Vision |
| Jeffrey Kang | TP Vision |
| <p>Note: The UPnP Forum in no way guarantees the accuracy or completeness of this author list and in no way implies any rights for or support from those members listed. This list is not the specifications' contributor list that is kept on the UPnP Forum's website.</p> | |

CONTENTS

| | | |
|----------|---|-----------|
| 1 | Scope | 10 |
| 2 | Normative references | 10 |
| 3 | Terms, definitions, symbols and abbreviations | 13 |
| 3.1 | Provisioning terms..... | 13 |
| 3.2 | Symbols | 14 |
| 4 | Notations and Conventions..... | 14 |
| 4.1 | Notation | 14 |
| 4.1.1 | Data Types..... | 14 |
| 4.1.2 | Strings Embedded in Other Strings..... | 14 |
| 4.1.3 | Extended Backus-Naur Form | 15 |
| 4.2 | Derived Data Types..... | 15 |
| 4.2.1 | Summary..... | 15 |
| 4.2.2 | CSV Lists | 15 |
| 4.3 | Management of XML Namespaces in Standardized DCPs | 17 |
| 4.3.1 | Namespace Prefix Requirements..... | 21 |
| 4.3.2 | Namespace Names, Namespace Versioning and Schema Versioning | 22 |
| 4.3.3 | Namespace Usage Examples | 24 |
| 4.4 | Vendor-defined Extensions..... | 24 |
| 4.4.1 | Vendor-defined Action Names | 24 |
| 4.4.2 | Vendor-defined State Variable Names | 25 |
| 4.4.3 | Vendor-defined XML Elements and attributes | 25 |
| 4.4.4 | Vendor-defined Property Names | 25 |
| 5 | Service Modeling Definitions | 25 |
| 5.1 | ServiceType | 25 |
| 5.2 | State Variables..... | 25 |
| 5.2.1 | State Variable Overview | 25 |
| 5.2.2 | <u>TransportState</u> | 27 |
| 5.2.3 | <u>TransportStatus</u> | 28 |
| 5.2.4 | <u>CurrentMediaCategory</u> | 28 |
| 5.2.5 | <u>PlaybackStorageMedium</u> | 28 |
| 5.2.6 | <u>RecordStorageMedium</u> | 30 |
| 5.2.7 | <u>PossiblePlaybackStorageMedia</u> | 30 |
| 5.2.8 | <u>PossibleRecordStorageMedia</u> | 30 |
| 5.2.9 | <u>CurrentPlayMode</u> | 30 |
| 5.2.10 | <u>TransportPlaySpeed</u> | 30 |
| 5.2.11 | <u>RecordMediumWriteStatus</u> | 31 |
| 5.2.12 | <u>CurrentRecordQualityMode</u> | 31 |
| 5.2.13 | <u>PossibleRecordQualityModes</u> | 31 |
| 5.2.14 | <u>NumberOfTracks</u> | 31 |
| 5.2.15 | <u>CurrentTrack</u> | 32 |
| 5.2.16 | <u>CurrentTrackDuration</u> | 32 |
| 5.2.17 | <u>CurrentMediaDuration</u> | 33 |
| 5.2.18 | <u>CurrentTrackMetaData</u> | 33 |
| 5.2.19 | <u>CurrentTrackURI</u> | 33 |
| 5.2.20 | <u>AVTransportURI</u> | 33 |
| 5.2.21 | <u>AVTransportURIMetaData</u> | 33 |

| | | |
|--------|---|----|
| 5.2.22 | <u>NextAVTransportURI</u> | 33 |
| 5.2.23 | <u>NextAVTransportURIMetaData</u> | 34 |
| 5.2.24 | <u>RelativeTimePosition</u> | 34 |
| 5.2.25 | <u>AbsoluteTimePosition</u> | 34 |
| 5.2.26 | <u>RelativeCounterPosition</u> | 34 |
| 5.2.27 | <u>AbsoluteCounterPosition</u> | 35 |
| 5.2.28 | <u>CurrentTransportActions</u> | 35 |
| 5.2.29 | <u>LastChange</u> | 35 |
| 5.2.30 | <u>DRMState</u> | 36 |
| 5.2.31 | <u>SyncOffset</u> | 36 |
| 5.2.32 | <u>A_ARG_TYPE_SeekMode</u> | 37 |
| 5.2.33 | <u>A_ARG_TYPE_SeekTarget</u> | 37 |
| 5.2.34 | <u>A_ARG_TYPE_InstanceID</u> | 38 |
| 5.2.35 | <u>A_ARG_TYPE_DeviceUDN</u> | 38 |
| 5.2.36 | <u>A_ARG_TYPE_ServiceType</u> | 38 |
| 5.2.37 | <u>A_ARG_TYPE_ServiceID</u> | 38 |
| 5.2.38 | <u>A_ARG_TYPE_StateVariableValuePairs</u> | 38 |
| 5.2.39 | <u>A_ARG_TYPE_StateVariableList</u> | 39 |
| 5.2.40 | <u>A_ARG_TYPE_PlaylistData</u> | 39 |
| 5.2.41 | <u>A_ARG_TYPE_PlaylistDataLength</u> | 39 |
| 5.2.42 | <u>A_ARG_TYPE_PlaylistOffset</u> | 39 |
| 5.2.43 | <u>A_ARG_TYPE_PlaylistTotalLength</u> | 39 |
| 5.2.44 | <u>A_ARG_TYPE_PlaylistMIMEType</u> | 40 |
| 5.2.45 | <u>A_ARG_TYPE_PlaylistExtendedType</u> | 40 |
| 5.2.46 | <u>A_ARG_TYPE_PlaylistStep</u> | 40 |
| 5.2.47 | <u>A_ARG_TYPE_PlaylistType</u> | 40 |
| 5.2.48 | <u>A_ARG_TYPE_PlaylistInfo</u> | 41 |
| 5.2.49 | <u>A_ARG_TYPE_PlaylistStartObjID</u> | 44 |
| 5.2.50 | <u>A_ARG_TYPE_PlaylistStartGroupID</u> | 44 |
| 5.2.51 | <u>A_ARG_TYPE_SyncOffsetAdj</u> | 44 |
| 5.2.52 | <u>A_ARG_TYPE_PresentationTime</u> | 45 |
| 5.2.53 | <u>A_ARG_TYPE_ClockId</u> | 45 |
| 5.3 | Eventing and Moderation | 45 |
| 5.3.1 | Eventing and Moderation Overview | 45 |
| 5.3.2 | Event Model | 47 |
| 5.4 | Actions | 48 |
| 5.4.1 | Action Overview | 48 |
| 5.4.2 | <u>SetAVTransportURI()</u> | 50 |
| 5.4.3 | <u>SetNextAVTransportURI()</u> | 51 |
| 5.4.4 | <u>GetMediaInfo()</u> | 53 |
| 5.4.5 | <u>GetMediaInfo_Ext()</u> | 54 |
| 5.4.6 | <u>GetTransportInfo()</u> | 55 |
| 5.4.7 | <u>GetPositionInfo()</u> | 55 |
| 5.4.8 | <u>GetDeviceCapabilities()</u> | 56 |
| 5.4.9 | <u>GetTransportSettings()</u> | 56 |
| 5.4.10 | <u>Stop()</u> | 57 |
| 5.4.11 | <u>Play()</u> | 58 |
| 5.4.12 | <u>Pause()</u> | 59 |

| | | |
|------------------------------|---|------------|
| 5.4.13 | <u>Record()</u> | 60 |
| 5.4.14 | <u>Seek()</u> | 61 |
| 5.4.15 | <u>Next()</u> | 63 |
| 5.4.16 | <u>Previous()</u> | 64 |
| 5.4.17 | <u>SetPlayMode()</u> | 65 |
| 5.4.18 | <u>SetRecordQualityMode()</u> | 66 |
| 5.4.19 | <u>GetCurrentTransportActions()</u> | 66 |
| 5.4.20 | <u>GetDRMState()</u> | 67 |
| 5.4.21 | <u>GetStateVariables()</u> | 67 |
| 5.4.22 | <u>SetStateVariables()</u> | 68 |
| 5.4.23 | <u>GetSyncOffset()</u> | 69 |
| 5.4.24 | <u>SetSyncOffset()</u> | 70 |
| 5.4.25 | <u>AdjustSyncOffset()</u> | 70 |
| 5.4.26 | <u>SyncPlay()</u> | 71 |
| 5.4.27 | <u>SyncStop()</u> | 73 |
| 5.4.28 | <u>SyncPause()</u> | 74 |
| 5.4.29 | <u>SetStaticPlaylist()</u> | 75 |
| 5.4.30 | <u>SetStreamingPlaylist()</u> | 77 |
| 5.4.31 | <u>GetPlaylistInfo()</u> | 78 |
| 5.4.32 | Common Error Codes | 79 |
| 6 | XML Service Description | 81 |
| 7 | Test | 101 |
| Annex A (normative) | <u>SetAVTransportURI()</u> Protocol Specifics | 102 |
| A.1 | Application to HTTP Streaming | 102 |
| A.1.1 | <u>AVTransportURI</u> Definition | 102 |
| A.1.2 | Control Point Behavior for <u>SetAVTransportURI()</u> | 102 |
| A.1.3 | Implementation of <u>SetAVTransportURI()</u> | 102 |
| A.1.4 | Cleanup | 102 |
| A.2 | Application to RTSP/RTP/UDP Streaming | 102 |
| A.2.1 | <u>AVTransportURI</u> Definition | 102 |
| A.2.2 | Control point behavior for <u>SetAVTransportURI()</u> | 103 |
| A.2.3 | Implementation of <u>SetAVTransportURI()</u> | 103 |
| A.2.4 | Cleanup | 103 |
| A.2.5 | Implementation of Transport Controls | 104 |
| A.3 | Application to Internal Streaming | 104 |
| A.3.1 | <u>AVTransportURI</u> Definition | 104 |
| A.3.2 | Implementation of <u>SetAVTransportURI()</u> | 104 |
| A.3.3 | Cleanup | 104 |
| A.4 | Application to IEC61883 Streaming | 104 |
| A.4.1 | <u>AVTransportURI</u> Definition | 104 |
| A.4.2 | Implementation of <u>SetAVTransportURI()</u> | 104 |
| A.4.3 | Cleanup | 104 |
| A.5 | Application to Vendor-specific Streaming | 105 |
| A.5.1 | <u>AVTransportURI</u> Definition | 105 |
| A.5.2 | Implementation of <u>SetAVTransportURI()</u> | 105 |
| A.5.3 | Cleanup | 105 |
| Annex B (informative) | Theory of Operation | 106 |

| | | |
|--|---|------------|
| B.1 | TransportState Control | 106 |
| B.2 | Transport Settings | 107 |
| B.3 | Navigation | 107 |
| B.4 | AVTransportURI Concept | 107 |
| B.5 | AVTransport Abstraction | 108 |
| B.6 | Supporting Multiple Virtual Transports | 110 |
| B.7 | Playlist Playback | 111 |
| B.8 | Dynamic Playlists | 112 |
| B.8.1 | Playlist Updating..... | 112 |
| B.8.2 | Determining Playlist Rendering Capabilities..... | 112 |
| B.8.3 | Submitting a Streaming Playlist | 113 |
| B.8.4 | Modifying a Current Playlist | 114 |
| B.8.5 | Submitting a Static Playlist | 115 |
| B.8.6 | Retrieving a Current Static Playlist | 116 |
| B.9 | <i>CLOCKSYNC</i> feature: Synchronized Playback | 117 |
| B.9.2 | Example of <u>SyncOffset</u> State Variable | 118 |
| B.9.3 | Example Scenarios for <u>SyncPlay()</u> Action | 118 |
| Annex C (informative) Bibliography..... | | 122 |

List of Tables

| | |
|--|----|
| Table 1 — EBNF Operators | 15 |
| Table 2 — CSV Examples | 16 |
| Table 3 — Namespace Definitions | 18 |
| Table 4 — Schema-related Information | 20 |
| Table 5 — Default Namespaces for the AV Specifications | 22 |
| Table 6 — State Variables | 26 |
| Table 7 — allowedValueList for TransportState | 28 |
| Table 8 — allowedValueRange for TransportStatus | 28 |
| Table 9 — allowedValueList for CurrentMediaCategory | 28 |
| Table 10 — allowedValueList for PlaybackStorageMedium | 29 |
| Table 11 — allowedValueList for CurrentPlayMode | 30 |
| Table 12 — allowedValueList for RecordMediumWriteStatus | 31 |
| Table 13 — allowedValueList for CurrentRecordQualityMode | 31 |
| Table 14 — allowedValueRange for NumberOfTracks | 32 |
| Table 15 — allowedValueRange for CurrentTrack | 32 |
| Table 16 — allowedValueList for CurrentTransportActions | 35 |
| Table 17 — allowedValueList for DRMState | 36 |
| Table 18 — allowedValueRange for SyncOffset | 37 |
| Table 19 — allowedValueList for A_ARG_TYPE_SeekMode | 37 |
| Table 20 — Format of A_ARG_TYPE_SeekTarget | 38 |
| Table 21 — allowedValueList for A_ARG_TYPE_PlaylistStep | 40 |
| Table 22 — allowedValueList for A_ARG_TYPE_PlaylistType | 40 |
| Table 23 — Event Moderation | 45 |
| Table 24 — Actions | 48 |
| Table 25 — Arguments for SetAVTransportURI() | 50 |
| Table 26 — Error Codes for SetAVTransportURI() | 51 |
| Table 27 — Arguments for SetNextAVTransportURI() | 52 |
| Table 28 — Error Codes for SetNextAVTransportURI() | 53 |
| Table 29 — Arguments for GetMediaInfo() | 53 |
| Table 30 — Error Codes for GetMediaInfo() | 54 |
| Table 31 — Arguments for GetMediaInfo_Ext() | 54 |
| Table 32 — Error Codes for GetMediaInfo_Ext() | 54 |
| Table 33 — Arguments for GetTransportInfo() | 55 |
| Table 34 — Error Codes for GetTransportInfo() | 55 |
| Table 35 — Arguments for GetPositionInfo() | 55 |
| Table 36 — Error Codes for GetPositionInfo() | 56 |
| Table 37 — Arguments for GetDeviceCapabilities() | 56 |
| Table 38 — Error Codes for GetDeviceCapabilities() | 56 |
| Table 39 — Arguments for GetTransportSettings() | 56 |
| Table 40 — Error Codes for GetTransportSettings() | 57 |
| Table 41 — Arguments for Stop() | 57 |

| | |
|--|----|
| Table 42 — Error Codes for <u>Stop()</u> | 58 |
| Table 43 — Arguments for <u>Play()</u> | 58 |
| Table 44 — Error Codes for <u>Play()</u> | 59 |
| Table 45 — Arguments for <u>Pause()</u> | 60 |
| Table 46 — Error Codes for <u>Pause()</u> | 60 |
| Table 47 — Arguments for <u>Record()</u> | 60 |
| Table 48 — Error Codes for <u>Record()</u> | 61 |
| Table 49 — Arguments for <u>Seek()</u> | 62 |
| Table 50 — Error Codes for <u>Seek()</u> | 63 |
| Table 51 — Arguments for <u>Next()</u> | 63 |
| Table 52 — Error Codes for <u>Next()</u> | 64 |
| Table 53 — Arguments for <u>Previous()</u> | 64 |
| Table 54 — Error Codes for <u>Previous()</u> | 65 |
| Table 55 — Arguments for <u>SetPlayMode()</u> | 65 |
| Table 56 — Error Codes for <u>SetPlayMode()</u> | 66 |
| Table 57 — Arguments for <u>SetRecordQualityMode()</u> | 66 |
| Table 58 — Error Codes for <u>SetRecordQualityMode()</u> | 66 |
| Table 59 — Arguments for <u>GetCurrentTransportActions()</u> | 67 |
| Table 60 — Error Codes for <u>GetCurrentTransportActions()</u> | 67 |
| Table 61 — Arguments for <u>GetDRMState()</u> | 67 |
| Table 62 — Error Codes for <u>GetDRMState()</u> | 67 |
| Table 63 — Arguments for <u>GetStateVariables()</u> | 68 |
| Table 64 — Error Codes for <u>GetStateVariables()</u> | 68 |
| Table 65 — Arguments for <u>SetStateVariables()</u> | 69 |
| Table 66 — Error Codes for <u>SetStateVariables()</u> | 69 |
| Table 67 — Arguments for <u>GetSyncOffset()</u> | 69 |
| Table 68 — Error Codes for <u>GetSyncOffset()</u> | 70 |
| Table 69 — Arguments for <u>SetSyncOffset()</u> | 70 |
| Table 70 — Error Codes for <u>SetSyncOffset()</u> | 70 |
| Table 71 — Arguments for <u>AdjustSyncOffset()</u> | 71 |
| Table 72 — Error Codes for <u>AdjustSyncOffset()</u> | 71 |
| Table 73 — Arguments for <u>SyncPlay()</u> | 72 |
| Table 74 — Error Codes for <u>SyncPlay()</u> | 73 |
| Table 75 — Arguments for <u>SyncStop()</u> | 74 |
| Table 76 — Error Codes for <u>SyncStop()</u> | 74 |
| Table 77 — Arguments for <u>SyncPause()</u> | 75 |
| Table 78 — Error Codes for <u>SyncPause()</u> | 75 |
| Table 79 — Arguments for <u>SetStaticPlaylist()</u> | 76 |
| Table 80 — Error Codes for <u>SetStaticPlaylist()</u> | 77 |
| Table 81 — Arguments for <u>SetStreamingPlaylist()</u> | 77 |
| Table 82 — Error Codes for <u>SetStreamingPlaylist()</u> | 78 |
| Table 83 — Arguments for <u>GetPlaylistInfo()</u> | 78 |

Table 84 — Error Codes for GetPlaylistInfo() 79

Table 85 — Common Error Codes 80

Table B.1 — Allowed AVTransportURIs 108

Table B.2 — Example mappings of resources type to track sequences 109

Table B.3 — Example seek modes, play modes and transport actions, per resource
type 110

List of Figures

Figure B.1 — TransportState Transitions - INFORMATIVE 106

Figure B.2 — SyncPlay() with past presentation time 119

Figure B.3 — SyncPlay() with future presentation time..... 120

Figure B.4 — SyncPlay() with future presentation time and reference positioning 121

1 Scope

This service definition is compliant with UPnP Device Architecture version 1.0 [14].

This service type enables control over the transport of audio and video streams. The service type defines a common model for A/V transport control suitable for a generic user interface. It can be used to control a wide variety of disc, tape and solid-state based media devices such as CD players, VCRs and MP3 players. A minimal implementation of this service can be used to control tuners.

The service type is related to the ConnectionManager service type, which describes A/V connection setup procedures, and the ContentDirectory service, which offers meta-information about the resource stored on the media. AVTransport also offers an action to retrieve any metadata embedded in the resource itself.

This service type does not offer scheduled recording.

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/rii-v1-20101231.xsd>.
Latest version available at: <http://www.upnp.org/schemas/av/rii.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] – *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).

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

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

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

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

[41] – *IETF RFC 3550, RTP: A Transport Protocol for Real-Time Applications*, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, July 2003.
Available at: <http://www.ietf.org/rfc/rfc3550.txt>.

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 |
|----------|---|
| ::= | definition – the non-terminal symbol on the left is defined by one or more alternative sequences of terminals and/or non-terminals to its right. |
| | alternative separator – separates sequences on the right that are independently allowed definitions for the non-terminal on the left. |
| * | null repetition – means the expression to its left may occur zero or more times. |
| + | non-null repetition – means the expression to its left shall occur at least once and may occur more times. |
| [] | optional – the expression between the brackets is allowed. |
| () | grouping – groups the expressions between the parentheses. |
| - | 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 (string) or CSV (xsd:string) | +artist,-date | List of 2 property sort criteria. |
| CSV (int) or CSV (xsd:integer) | “1,-5,006,0,+7” | List of 5 integers. |
| CSV (boolean) or CSV (xsd:Boolean) | “0,1,1,0” | List of 4 booleans |
| CSV (string) or CSV (xsd:string) | “Smith\, Fred,Jones\, Davey” | List of 2 names, “Smith, Fred” and “Jones, Davey” |
| CSV (i4, string, ui2) or CSV (xsd:int, xsd:string, xsd:unsignedShort) | “-29837, string with leading blanks,0” | Note that the second value is “ string with leading blanks” |
| CSV (i4) or CSV (xsd:int) | “3, 4” | Illegal CSV. White space is not allowed as part of an integer value. |
| CSV (string) or CSV (xsd:string) | “,,” | List of 3 empty string values |
| CSV (heterogeneous) | “Alice,Marketing,5,Sue,R&D,21,Dave,Finance,7” | List of unspecified number of people and associated attributes. Each person is described by 3 elements: a name string , a department string and years-of-service ui2 or a name xsd:string, a department xsd:string and years-of-service xsd:unsignedShort. |

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] |
| 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] |
| rcs-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] |

| Standard Name-space Prefix | Namespace Name | Namespace Description | Normative Definition Document Reference |
|--------------------------------------|---|-------------------------------------|--|
| <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-yyyyymmdd.xsd AllowedTransformSetting s-vn.xsd AllowedTransformSetting s.xsd | <TransformList> | [1] |
| av | av-vn-yyyyymmdd.xsd av-vn.xsd av.xsd | n/a | [3] |
| avdt | avdt-vn-yyyyymmdd.xsd avdt-vn.xsd avdt.xsd | <AVDT> | [2] |
| avs | avs-vn-yyyyymmdd.xsd avs-vn.xsd avs.xsd | <Capabilities> <Features> <stateVariableValuePairs> | [4] |
| avt-event | avt-event-vn- yyyyymmdd.xsd avt-event-vn.xsd avt-event.xsd | <Event> | [6] |
| cds-event | cds-event-vn- yyyyymmdd.xsd cds-event-vn.xsd cds-event.xsd | <StateEvent> | [8] |
| cm-dciu | cm- deviceClockInfoUpdates- vn-yyyyymmdd.xsd cm- deviceClockInfoUpdates -vn.xsd cm- deviceClockInfoUpdates. xsd | <DeviceClockInfoUpdates> | [10] |
| cm-ftrlst | cm-featureList-vn- yyyyymmdd.xsd cm-featureList-vn.xsd cm-featureList.xsd | <Features> | [11] |
| didl-lite | didl-lite-vn- yyyyymmdd.xsd didl-lite-vn.xsd didl-lite.xsd | <DIDL-Lite> | [15] |
| dmo | dmo-vn-yyyyymmdd.xsd dmo-vn.xsd dmo.xsd | <DeviceMode> | [16] |
| dmor | dmor-vn-yyyyymmdd.xsd dmor-vn.xsd dmor.xsd | <DeviceModeRequest> | [17] |
| dmos | dmos-vn-yyyyymmdd.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] |
| <i>Externally Defined Namespaces</i> | | | |
| dc | <i>Absolute URL:</i> 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 | rcs-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](http://www.upnp.org/schemas/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](http://www.upnp.org/schemas/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](http://www.upnp.org/schemas/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](http://www.upnp.org/schemas/av/rcs-event.xsd)” was linked to the schema that is otherwise known as “[.../av/rcs-event-v1-20020625.xsd](http://www.upnp.org/schemas/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](http://www.upnp.org/schemas/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 ServiceType

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

urn:schemas-upnp-org:service:AVTransport:3

5.2 State Variables

5.2.1 State Variable Overview

Table 6 — State Variables

| Variable Name | R/A ^a | Data Type | Allowed Value | Default Value | Eng. Units |
|---|------------------------|-------------------------------|---|--------------------|------------|
| <u>TransportState</u> | <i>R</i> | <u>string</u> | See 5.2.2 | | |
| <u>TransportStatus</u> | <i>R</i> | <u>string</u> | See 5.2.3 | | |
| <u>CurrentMediaCategory</u> | <i>R</i> | <u>string</u> | See 5.2.4 | | |
| <u>PlaybackStorageMedium</u> | <i>R</i> | <u>string</u> | See 5.2.5 | | |
| <u>RecordStorageMedium</u> | <i>R</i> | <u>string</u> | See 5.2.6 | | |
| <u>PossiblePlaybackStorageMedia</u> | <i>R</i> | <u>string</u> | CSV ^b (<u>string</u>) See 5.2.7 | | |
| <u>PossibleRecordStorageMedia</u> | <i>R</i> | <u>string</u> | CSV (<u>string</u>) See 5.2.8 | | |
| <u>CurrentPlayMode</u> | <i>R</i> | <u>string</u> | See 5.2.9 | " <i>NORMAL</i> " | |
| <u>TransportPlaySpeed</u> | <i>R</i> | <u>string</u> | See 5.2.10 | " <i>1</i> " | |
| <u>RecordMediumWriteStatus</u> | <i>R</i> | <u>string</u> | See 5.2.11 | | |
| <u>CurrentRecordQualityMode</u> | <i>R</i> | <u>string</u> | See 5.2.12 | | |
| <u>PossibleRecordQualityModes</u> | <i>R</i> | <u>string</u> | CSV (<u>string</u>) See 5.2.13 | | |
| <u>NumberOfTracks</u> | <i>R</i> | <u>ui4</u> | See 5.2.14 | | |
| <u>CurrentTrack</u> | <i>R</i> | <u>ui4</u> | See 5.2.15 | | |
| <u>CurrentTrackDuration</u> | <i>R</i> | <u>string</u> | See 5.2.16 | | |
| <u>CurrentMediaDuration</u> | <i>R</i> | <u>string</u> | See 5.2.17 | | |
| <u>CurrentTrackMetaData</u> | <i>R</i> | <u>string</u> | See 5.2.18 | | |
| <u>CurrentTrackURI</u> | <i>R</i> | <u>string</u> | See 5.2.19 | | |
| <u>AVTransportURI</u> | <i>R</i> | <u>string</u> | See 5.2.20 | | |
| <u>AVTransportURIMetaData</u> | <i>R</i> | <u>string</u> | See 5.2.21 | | |
| <u>NextAVTransportURI</u> | <i>R</i> | <u>string</u> | See 5.2.22 | | |
| <u>NextAVTransportURIMetaData</u> | <i>R</i> | <u>string</u> | See 5.2.23 | | |
| <u>RelativeTimePosition</u> | <i>R</i> | <u>string</u> | See 5.2.24 | | |
| <u>AbsoluteTimePosition</u> | <i>R</i> | <u>string</u> | See 5.2.25 | | |
| <u>RelativeCounterPosition</u> | <i>R</i> | <u>i4</u> | See 5.2.26 | | |
| <u>AbsoluteCounterPosition</u> | <i>R</i> | <u>ui4</u> | See 5.2.27 | | |
| <u>CurrentTransportActions</u> | <i>CR</i> ^c | <u>string</u> | CSV (<u>string</u>) See 5.2.28 | | |
| <u>LastChange</u> | <i>R</i> | <u>string</u> | See 5.2.29 | | |
| <u>DRMState</u> | <i>CR</i> ^c | <u>string</u> | See 5.2.30 | " <i>UNKNOWN</i> " | |
| <u>SyncOffset</u> | <i>CR</i> ^c | <u>string</u> | See 5.2.31 | | |
| <u>A_ARG_TYPE_SeekMode</u> | <i>R</i> | <u>string</u> | See 5.2.32 | | |
| <u>A_ARG_TYPE_SeekTarget</u> | <i>R</i> | <u>string</u> | See 5.2.33 | | |
| <u>A_ARG_TYPE_InstanceID</u> | <i>R</i> | <u>ui4</u> | See 5.2.34 | | |
| <u>A_ARG_TYPE_DeviceUDN</u> | <i>CR</i> ^c | <u>string</u> | See 5.2.35 | | |
| <u>A_ARG_TYPE_ServiceType</u> | <i>CR</i> ^c | <u>string</u> | See 5.2.36 | | |
| <u>A_ARG_TYPE_ServiceID</u> | <i>CR</i> ^c | <u>string</u> | See 5.2.37 | | |
| <u>A_ARG_TYPE_StateVariableValuePairs</u> | <i>CR</i> ^c | <u>string</u> | See 5.2.38 | | |

| Variable Name | R/A ^a | Data Type | Allowed Value | Default Value | Eng. Units |
|--|------------------------|---------------|---|---------------|------------|
| A_ARG_TYPE_StateVariableList | <u>CR</u> ^c | <u>string</u> | CSV (<u>string</u>) See 5.2.39 | | |
| A_ARG_TYPE_PlaylistData | <u>CR</u> ^c | <u>string</u> | See 5.2.40 | | |
| A_ARG_TYPE_PlaylistDataLength | <u>CR</u> ^c | <u>ui4</u> | See 5.2.41 | | |
| A_ARG_TYPE_PlaylistOffset | <u>CR</u> ^c | <u>ui4</u> | See 5.2.42 | | |
| A_ARG_TYPE_PlaylistTotalLength | <u>CR</u> ^c | <u>ui4</u> | See 5.2.43 | | |
| A_ARG_TYPE_PlaylistMIMETYPE | <u>CR</u> ^c | <u>string</u> | 3 rd field of res@protocolInfo See 5.2.44 | | |
| A_ARG_TYPE_PlaylistExtendedType | <u>CR</u> ^c | <u>string</u> | 4 th field of res@protocolInfo See 5.2.45 | | |
| A_ARG_TYPE_PlaylistStep | <u>CR</u> ^c | <u>string</u> | See 5.2.46 | | |
| A_ARG_TYPE_PlaylistType | <u>CR</u> ^c | <u>string</u> | See 5.2.47 | | |
| A_ARG_TYPE_PlaylistInfo | <u>CR</u> ^c | <u>string</u> | See 5.2.48 | | |
| A_ARG_TYPE_PlaylistStartObjID | <u>CR</u> ^c | <u>string</u> | See 5.2.49 | | |
| A_ARG_TYPE_PlaylistStartGroupID | <u>CR</u> ^c | <u>string</u> | See 5.2.50 | | |
| A_ARG_TYPE_SyncOffsetAdj | <u>CR</u> ^c | <u>string</u> | See 5.2.51 | | |
| A_ARG_TYPE_PresentationTime | <u>CR</u> ^c | <u>string</u> | See 5.2.52 | | |
| A_ARG_TYPE_ClockId | <u>CR</u> ^c | <u>string</u> | See 5.2.53 | | |
| <i>Non-standard state variables implemented by a UPnP vendor go here</i> | <u>X</u> | <i>TBD</i> | <i>TBD</i> | <i>TBD</i> | <i>TBD</i> |

^a For a device this column indicates whether the state variable shall be implemented or not, where R = required, A = allowed, CR = conditionally required, CA = conditionally allowed, X = non-standard, add -D when deprecated (e.g., R-D, A-D).

^b CSV stands for Comma-Separated Value list. The type between brackets denotes the UPnP data type used for the elements inside the list. CSV is defined more formally in the ContentDirectory service template.

^c See referenced subclause for conditions under which implementation of this state variable is required.

5.2.2 TransportState

This required state variable forms the core of the AVTransport service. It defines the conceptually top-level state of the transport, for example, whether it is playing, recording, etc. Device vendors do not need to implement all allowed values of this variable, for example, non-recordable media will not implement the "RECORDING" state.

The "PAUSED_RECORDING" state is different from the "STOPPED" state in the sense that the transport is already prepared for recording and can respond faster or more accurately. The "PAUSED_PLAYBACK" state is different from the "PAUSED_RECORDING" state in the sense that in case the media contains video, it indicates output of a still image. The other TransportState values are self explanatory.

Note that *dubbing* of media at various speeds is not supported in this version of the AVTransport, mainly because there are no standards for cross-device dubbing speeds.

Device vendors are allowed to implement additional vendor-defined transport states. However, since the semantic meaning of these transport states is not specified, control points that find a AVTransport service in a transport state that they do not understand are encouraged to refrain from interacting with that AVTransport service (for example, forcing the service into the "STOPPED" state). Rather, they are encouraged to wait until the service transits back into a transport state that they understand.

Table 7 — allowedValueList for TransportState

| Value | R/A |
|--------------------|---|
| "STOPPED" | <u>R</u> |
| "PLAYING" | <u>R</u> |
| "TRANSITIONING" | <u>R</u> |
| "PAUSED_PLAYBACK" | <u>CR</u> , required if <u>Pause()</u> action is implemented |
| "PAUSED_RECORDING" | <u>CR</u> , required if both <u>Pause()</u> and <u>Record()</u> actions are implemented |
| "RECORDING" | <u>CR</u> , required if <u>Record()</u> action is implemented |
| "NO_MEDIA_PRESENT" | <u>A</u> |
| Vendor-defined | <u>X</u> |

5.2.3 TransportStatus

This required state variable is used to indicate if asynchronous errors have occurred, during operation of the AVTransport service, that cannot be returned by a normal action. For example, some time after playback of a stream has been started (via SetAVTransportURI() and Play() actions), there can be network congestion or server problems causing hiccups in the rendered media. These types of situations can be signaled to control points by setting this state variable to value "ERROR_OCCURRED". More specific error descriptions may also be used as vendor extensions. The value of TransportState after an error has occurred is implementation-dependent; some implementations may go to "STOPPED" while other implementations may be able to continue playing after an error. The time at which this state variable returns to "OK" after an error situation is also implementation dependent.

Table 8 — allowedValueRange for TransportStatus

| Value | R/A |
|------------------|----------|
| "OK" | <u>R</u> |
| "ERROR_OCCURRED" | <u>R</u> |
| Vendor-defined | <u>X</u> |

5.2.4 CurrentMediaCategory

This required state variable indicates whether the current media is track-aware (both single and multi-track) or track-unaware (e.g. VHS-tape). The semantics of state variables RelativeTimePosition, AbsoluteTimePosition, RelativeCounterPosition, AbsoluteCounterPosition and of the Seek() action change, depending on this state variable.

Table 9 — allowedValueList for CurrentMediaCategory

| Value | R/A |
|-----------------|----------|
| "NO_MEDIA" | <u>R</u> |
| "TRACK_AWARE" | <u>R</u> |
| "TRACK_UNAWARE" | <u>R</u> |

5.2.5 PlaybackStorageMedium

This required state variable indicates the storage medium of the resource specified by AVTransportURI. If no resource is specified, then the state variable is set to "NONE". If AVTransportURI refers to a resource received from the UPnP network, the state variable is set to "NETWORK". Device vendors may extend the specified allowed value list of this variable. For example, various types of solid-state media formats can be added in a vendor-specific way.

Note that this variable is not intended for signal- or content-formats such as MPEG2. Such type of information is exposed by the ConnectionManager service associated with this service.

Table 10 — allowedValueList for *PlaybackStorageMedium*

| Value | R/A | Description |
|----------------------------|-----|--|
| " <u>UNKNOWN</u> " | A | Unknown medium |
| " <u>DV</u> " | A | Digital Video Tape medium |
| " <u>MINI-DV</u> " | A | Mini Digital Video Tape medium |
| " <u>VHS</u> " | A | VHS Tape medium |
| " <u>W-VHS</u> " | A | W-VHS Tape medium |
| " <u>S-VHS</u> " | A | Super VHS Tape medium |
| " <u>D-VHS</u> " | A | Digital VHS Tape medium |
| " <u>VHSC</u> " | A | Compact VHS medium |
| " <u>VIDEO8</u> " | A | 8 mm Video Tape medium |
| " <u>HI8</u> " | A | High resolution 8 mm Video Tape medium |
| " <u>CD-ROM</u> " | A | Compact Disc-Read Only Memory medium |
| " <u>CD-DA</u> " | A | Compact Disc-Digital Audio medium |
| " <u>CD-R</u> " | A | Compact Disc-Recordable medium |
| " <u>CD-RW</u> " | A | Compact Disc-Rewritable medium |
| " <u>VIDEO-CD</u> " | A | Video Compact Disc medium |
| " <u>SACD</u> " | A | Super Audio Compact Disc medium |
| " <u>MD-AUDIO</u> " | A | Mini Disc Audio medium |
| " <u>MD-PICTURE</u> " | A | Mini Disc Picture medium |
| " <u>DVD-ROM</u> " | A | DVD Read Only medium |
| " <u>DVD-VIDEO</u> " | A | DVD Video medium |
| " <u>DVD+R</u> " | A | DVD Recordable medium |
| " <u>DVD-R</u> " | A | DVD Recordable medium |
| " <u>DVD+RW</u> " | A | DVD Rewritable medium |
| " <u>DVD-RW</u> " | A | DVD Rewritable medium |
| " <u>DVD-RAM</u> " | A | DVD RAM medium |
| " <u>DVD-AUDIO</u> " | A | DVD Audio medium |
| " <u>DAT</u> " | A | Digital Audio Tape medium |
| " <u>LD</u> " | A | Laser Disk medium |
| " <u>HDD</u> " | A | Hard Disk Drive medium |
| " <u>MICRO-MV</u> " | A | Micro MV Tape medium |
| " <u>NETWORK</u> " | A | Network Interface medium |
| " <u>NONE</u> " | A | No medium present |
| " <u>NOT_IMPLEMENTED</u> " | A | Medium type discovery is not implemented |
| " <u>SD</u> " | A | SD (Secure Digital) Memory Card medium |
| " <u>PC-CARD</u> " | A | PC Card medium |
| " <u>MMC</u> " | A | MultimediaCard medium |
| " <u>CF</u> " | A | Compact Flash medium |
| " <u>BD</u> " | A | Blu-ray Disc medium |
| " <u>MS</u> " | A | Memory Stick medium |
| " <u>HD_DVD</u> " | A | HD DVD medium |

| Value | R/A | Description |
|-----------------------|----------|-------------|
| <i>Vendor-defined</i> | <i>X</i> | |

5.2.6 RecordStorageMedium

This required state variable indicates the storage medium where the resource specified by AVTransportURI will be recorded when a Record action is issued. If no resource is specified, then the state variable is set to “NONE”. Device vendors may extend the allowed value list of this variable. For example, various types of solid-state media formats can be added in a vendor-specific way.

Note that this variable is not intended for signal- or content-formats such as MPEG2. Such type of information is exposed by the ConnectionManager service associated with this service. If the service implementation does not support recording, then this state variable shall be set to “NOT_IMPLEMENTED”. The allowed values for this state variable are the same as the PlaybackStorageMedium state variable.

5.2.7 PossiblePlaybackStorageMedia

This required state variable contains a static, comma-separated list of storage media that the device can play. Recommended values are defined in the allowed value list for state variable PlaybackStorageMedium.

5.2.8 PossibleRecordStorageMedia

This required state variable contains a static, comma-separated list of storage media onto which the device can record. Recommended values are defined in the allowed value list for state variable RecordStorageMedium. If the service implementation does not support recording, then this state variable shall be set to “NOT_IMPLEMENTED”.

5.2.9 CurrentPlayMode

This required state variable indicates the current play mode (for example, random play, repeated play, etc.). This notion is typical for CD-based audio media, but is generally not supported by tape-based media. Value “DIRECT_1” indicates playing a single track and then stop (don’t play the next track). Value “INTRO” indicates playing a short sample (typically 10 seconds or so) of each track on the media. Other play mode values are self explanatory.

Table 11 — allowedValueList for CurrentPlayMode

| Value | R/A |
|-----------------------|----------|
| “ <u>NORMAL</u> ” | <u>R</u> |
| “ <u>SHUFFLE</u> ” | <u>A</u> |
| “ <u>REPEAT_ONE</u> ” | <u>A</u> |
| “ <u>REPEAT_ALL</u> ” | <u>A</u> |
| “ <u>RANDOM</u> ” | <u>A</u> |
| “ <u>DIRECT_1</u> ” | <u>A</u> |
| “ <u>INTRO</u> ” | <u>A</u> |
| <i>Vendor-defined</i> | <i>X</i> |

5.2.10 TransportPlaySpeed

This required state variable is a string representation of a rational fraction that indicates the speed relative to normal speed. Example values are “1”, “1/2”, “2”, “-1”, “1/10”, etc. Value “1” is required, value “0” is not allowed. Device vendors may support additional play speeds. Negative values indicate reverse playback. Actually supported speeds can be retrieved from the allowed value list of this state variable in the AVTransport service description.

5.2.11 RecordMediumWriteStatus

This required state variable reflects the write protection status of the currently loaded media. “NOT WRITABLE” indicates an inherent *read-only* media (for example, a DVD-ROM disc) or the device doesn’t support recording on the current media. “PROTECTED” indicates a writable media that is currently write-protected (for example, a protected VHS tape). If no media is loaded, the write status will be “UNKNOWN”. If the service implementation does not support recording, then this state variable shall be set to “NOT_IMPLEMENTED”.

Table 12 — allowedValueList for RecordMediumWriteStatus

| Value | R/A |
|----------------------------|----------|
| “ <u>WRITABLE</u> ” | <u>A</u> |
| “ <u>PROTECTED</u> ” | <u>A</u> |
| “ <u>NOT WRITABLE</u> ” | <u>A</u> |
| “ <u>UNKNOWN</u> ” | <u>A</u> |
| “ <u>NOT_IMPLEMENTED</u> ” | <u>A</u> |
| <i>Vendor-defined</i> | <u>X</u> |

5.2.12 CurrentRecordQualityMode

This required state variable indicates the currently set record quality mode. Such a setting takes the form of “Quality Ordinal:label”. The Quality Ordinal indicates a particular relative quality level available in the device, from 0 (lowest quality) to n (highest quality). The label associated with the ordinal provides a human-readable indication of the ordinal’s meaning. If the service implementation does not support recording, then this state variable shall be set to “NOT_IMPLEMENTED”.

Table 13 — allowedValueList for CurrentRecordQualityMode

| Value | R/A |
|----------------------------|----------|
| “ <u>0:EP</u> ” | <u>A</u> |
| “ <u>1:LP</u> ” | <u>A</u> |
| “ <u>2:SP</u> ” | <u>A</u> |
| “ <u>0:BASIC</u> ” | <u>A</u> |
| “ <u>1:MEDIUM</u> ” | <u>A</u> |
| “ <u>2:HIGH</u> ” | <u>A</u> |
| “ <u>NOT_IMPLEMENTED</u> ” | <u>A</u> |
| <i>Vendor-defined</i> | <u>X</u> |

5.2.13 PossibleRecordQualityModes

This required state variable contains a static, comma-separated list of recording quality modes that the device supports. For example, for an analog VHS recorder the string would be “0:EP,1:LP,2:SP”, while for a PVR the string would be “0:BASIC,1:MEDIUM,2:HIGH”. The string specifics depend on the type of device containing the AVTransport. Note that record quality modes are independent of the *content-format* that may be exposed to the network through a ConnectionManager service. If the service implementation does not support recording, then this state variable shall be set to “NOT_IMPLEMENTED”.

5.2.14 NumberOfTracks

This required state variable contains the number of tracks controlled by the AVTransport instance. If no resource is associated with the AVTransport instance (via SetAVTransportURI()), and there is no *default* resource (for example, a loaded disc) then NumberOfTracks shall be 0. Also, if the implementation is never able to determine the number of tracks in the currently selected media, NumberOfTracks shall be set to 0. Otherwise, it shall be 1 or higher. In some cases, for example, large playlist, it can take a long time to determine

the exact number of tracks. Until the exact number is determined, the value of the state variable is implementation dependent, for example, keeping it to 1 until determined or updating the value periodically. Note that in any case, the AVTransport service shall generate a [LastChange](#) event with defined moderation period when the exposed value is updated.

For track-unaware media, this state variable will always be set to 1. For LD and DVD media, a track is defined as a chapter number. For Tuners that provide an indexed list of channels, a track is defined as an index number in such a list. This state variable has to be consistent with the resource identified by [AVTransportURI](#). For example, if [AVTransportURI](#) points to a single MP3 file, then [NumberOfTracks](#) shall be set to 1. However, if [AVTransportURI](#) points to a playlist file, then [NumberOfTracks](#) shall be equal to the number of entries in the playlist.

Table 14 — allowedValueRange for [NumberOfTracks](#)

| | Value | R/A |
|-------------------------|----------------|-----|
| minimum | 0 | R |
| maximum | vendor-defined | R |

5.2.15 [CurrentTrack](#)

This is a required state variable. If [NumberOfTracks](#) is 0, then [CurrentTrack](#) will be 0. Otherwise, this state variable contains the sequence number of the currently selected track, starting at value 1, up to and including [NumberOfTracks](#). For track-unaware media, this state variable is always 1. For LD and DVD media, the notion of track equals the notion of chapter number. For Tuners that provide an indexed list of channels, the current track is defined as the current index number in such a list.

Table 15 — allowedValueRange for [CurrentTrack](#)

| | Value | R/A |
|-------------------------|----------------|-----|
| minimum | 0 | R |
| maximum | vendor-defined | R |
| step | 1 | R |

5.2.16 [CurrentTrackDuration](#)

This required state variable contains the duration of the current track, specified as a string of the following form:

H+:MM:SS[.F+] or H+:MM:SS[.F0/F1]

where:

- H+: one or more digits to indicate elapsed hours,
- MM: exactly 2 digits to indicate minutes (00 to 59),
- SS: exactly 2 digits to indicate seconds (00 to 59),
- F+: one or more digits to indicate fractions of seconds,
- F0/F1: a fraction, with F0 and F1 at least one digit long, and F0 < F1.

The string may be preceded by an allowed "+" or "-" sign, and the decimal point itself shall be omitted if there are no fractional second digits. This variable does not apply to Tuners. If the implementation is never able to determine the duration of the current track, [CurrentTrackDuration](#) shall be set to "00:00:00". If the optional fractional components are included, they shall be set to either "0" or "0/<F1>". In some cases, it can take a long time to determine the exact duration of tracks. Until the exact duration is determined, the value of the state variable is implementation dependent, for example, keeping it to "00:00:00" or updating the value periodically. Note that in any case, the AVTransport service shall generate a [LastChange](#) event with defined moderation period when the exposed value is updated. If the

service implementation does not support track duration information then this state variable shall be set to "NOT_IMPLEMENTED".

5.2.17 CurrentMediaDuration

This required state variable contains the duration of the media, as identified by state variable AVTransportURI. In case the AVTransportURI represents only 1 track, this state variable is equal to CurrentTrackDuration. The format of this variable is the same as the format for CurrentTrackDuration, described above in 5.2.16. If no content is associated with the AVTransport instance (via SetAVTransportURI()), and there is no default content (for example, a loaded disc) then CurrentMediaDuration shall be set to "00:00:00". Also, if the implementation is never able to determine the duration of the currently selected media, CurrentMediaDuration shall be set to "00:00:00". If the optional fractional components are included, they shall be set to either "0" or "0/<F1>". In some cases, it can take a long time to determine the exact duration of the media. Until the exact duration is determined, the value of the state variable is implementation dependent, for example, keeping it to "00:00:00" or updating the value periodically. Note that in any cases, the AVTransport service shall generate a LastChange event with defined moderation period when the exposed value is updated. If the service implementation does not support media duration information, then this state variable shall be set to "NOT_IMPLEMENTED".

5.2.18 CurrentTrackMetaData

This required state variable contains the metadata, in the form of a *DIDL-Lite XML Fragment* (defined in the ContentDirectory service template), associated with the resource pointed to by state variable CurrentTrackURI. The metadata could have been extracted from state variable AVTransportURIMetaData, or extracted from the resource binary itself (for example, embedded ID3 tags for MP3 audio). This is implementation dependent. If the service implementation does not support this feature, then this state variable shall be set to "NOT_IMPLEMENTED".

5.2.19 CurrentTrackURI

This required state variable contains a reference, in the form of a URI, to the current track. The URI can enable a control point to retrieve any meta-data associated with the current track, such as title and author information, via the ContentDirectory service Browse() and/or Search() action. In case the media does contain multi-track content, but there is no separate URI associated with each track, CurrentTrackURI shall be set equal to AVTransportURI.

5.2.20 AVTransportURI

This required state variable contains a reference, in the form of a URI, to the resource controlled by the AVTransport instance. This URI can refer to a single item (for example, a song) or to a collection of items (for example, a playlist). In the *single item* case, the AVTransport will have 1 track and AVTransportURI is equal to CurrentTrackURI. In the *collection of items* case, the AVTransport will have multiple tracks, and AVTransportURI will remain constant during track changes. The URI enables a control point to retrieve any meta-data associated with the AVTransport instance, such as title and author information, via the ContentDirectory service.

5.2.21 AVTransportURIMetaData

This required state variable contains the meta-data, in the form of a *DIDL-Lite XML Fragment* (defined in the ContentDirectory service template), associated with the resource pointed to by state variable AVTransportURI. See the ContentDirectory service specification [7] for details. If the service implementation does not support this feature, then this state variable shall be set to "NOT_IMPLEMENTED".

5.2.22 NextAVTransportURI

This required state variable contains the AVTransportURI value to be played when the playback of the current AVTransportURI finishes. Setting this variable ahead of time (via action SetNextAVTransportURI()) enables a device to provide seamless transitions between resources for certain data transfer protocols that need buffering (for example, HTTP). If the

service implementation does not support this feature, then this state variable shall be set to “NOT_IMPLEMENTED”.

Do not confuse transitions between AVTransportURI and NextAVTransportURI with track transitions. When AVTransportURI is set to a playlist, NextAVTransportURI will be played when the whole playlist finishes, not when the current playlist entry (CurrentTrackURI) finishes.

5.2.23 NextAVTransportURIMetaData

This required state variable contains the meta-data, in the form of a *DIDL-Lite XML Fragment* (defined in the ContentDirectory service template), associated with the resource pointed to by state variable NextAVTransportURI. See the ContentDirectory service specification [7] for details. If the service implementation does not support this feature then this state variable shall be set to “NOT_IMPLEMENTED”.

5.2.24 RelativeTimePosition

For track-aware media, this required state variable contains the current position in the current track, in terms of time, measured from the beginning of the current track. The range for this state variable is from “00:00:00” to the duration of the current track as indicated by the CurrentTrackDuration state variable. For track-aware media, this state variable always contains a positive value.

For track-unaware media (e.g. a single tape), this state variable contains the position, in terms of time, measured from a *zero reference point* on the media. The range for this state variable is from the beginning of the media, measured from the zero reference point to the end of the media, also measured from the zero reference point. For track-unaware media, this state variable can be negative. Indeed, when the zero reference point does not coincide with the beginning of the media, all positions before the zero reference point are expressed as negative values.

The time format used for the RelativeTimePosition state variable is the same as for state variable CurrentTrackDuration. If the service implementation does not support relative time-based position information, then this state variable shall be set to “NOT_IMPLEMENTED”.

5.2.25 AbsoluteTimePosition

This required state variable contains the current position, in terms of time, measured from the beginning of the media. The time format used for the AbsoluteTimePosition state variable is the same as for state variable CurrentTrackDuration. The range for this state variable is from “00:00:00” to the duration of the current media as indicated by the CurrentMediaDuration state variable. This state variable always contains a positive value.

If the service implementation does not support any kind of position information, then this state variable shall be set to “NOT_IMPLEMENTED”. Devices that do not have time position information, but are able to detect whether they are at the end of the media shall use special value “END_OF_MEDIA” when actually at the end, and the value “NOT_IMPLEMENTED” otherwise.

5.2.26 RelativeCounterPosition

For track-aware media, this required state variable contains the current position in the current track, in terms of a dimensionless counter, measured from the beginning of the current track. The range for this state variable is from 0 to the counter value that corresponds to the end of the current track. For track-aware media, this state variable always contains a positive value.

For track-unaware media (e.g. a single tape), this state variable contains the position, in terms of a dimensionless counter, measured from a *zero reference point* on the media. The range for this state variable is from the counter value that corresponds to the beginning of the media, measured from the zero reference point to the counter value that corresponds to the end of the media, also measured from the zero reference point. For track-unaware media, this state variable can be negative. Indeed, when the zero reference point does not coincide with

the beginning of the media, all positions before the zero reference point are expressed as negative values.

For devices that support media with addressable ranges that equal or exceed the allowed range of this counter, the AVTransport service shall scale actual media addresses to counter values to fit within the range allowed for this counter.

If the service implementation does not support relative count-based position information, then this state variable shall be set to the maximum value of the [i4](#) data type.

5.2.27 AbsoluteCounterPosition

This required state variable contains the current position, in terms of a dimensionless counter, measured from the beginning of the loaded media. The allowed range for this variable is [0, 2147483646]. For devices that support media with addressable ranges that equal or exceed the allowed range of this counter, the AVTransport service shall scale actual media addresses to counter values to fit within the range allowed for this counter. If the service implementation does not support absolute count-based position information, then this state variable shall be set to the value 2147483647.

Note: Although the data type for state variable [AbsoluteCounterPosition](#) is [ui4](#), the range is restricted to [0, Max([i4](#))] for backwards compatibility reasons.

5.2.28 CurrentTransportActions

This conditionally required state variable shall be supported if the AVTransport service implements the [GetCurrentTransportActions\(\)](#) action, otherwise it is not allowed. This state variable contains a comma-separated list of transport-controlling actions that can be successfully invoked for the current resource at this specific point in time. The list shall contain a subset (including the empty set) of the following action names: [“Play”](#), [“Stop”](#), [“Pause”](#), [“Seek”](#), [“Next”](#), [“Previous”](#) and [“Record”](#). In addition, the list may be augmented by a subset of vendor-defined transport-controlling action names. For example, when a live stream from the Internet is being controlled, the variable can be only [“Play,Stop”](#). When a local audio CD is being controlled, the variable can be [“Play,Stop,Pause,Seek,Next,Previous”](#). This information can be used, for example, to dynamically enable or disable play, stop, and pause buttons, etc., on a user interface.

Table 16 — allowedValueList for [CurrentTransportActions](#)

| Value | R/A |
|--------------------------------|---|
| “PLAY” | R |
| “STOP” | R |
| “PAUSE” | CR , required if Pause() action is implemented. Not allowed otherwise. |
| “SEEK” | R |
| “NEXT” | R |
| “PREVIOUS” | R |
| “RECORD” | CR , required if Record() action is implemented. Not allowed otherwise. |
| Vendor-defined | X |

5.2.29 LastChange

This required state variable is used for eventing purposes to enable clients to receive meaningful event notifications whenever the state of the AVTransport changes. Logically, it contains a list of pairs, one element being an AVTransport instance ID and the second element the name and new value of the state variable for that instance. The format of the [LastChange](#) state variable is defined in [6]. The [LastChange](#) state variable follows the behavior of the [LastChange](#) state variable as described in the RenderingControl service specification [21], subclause 5.2.1.

5.2.30 DRMState

This conditionally required state variable shall be supported if the AVTransport service implements the [GetDRMState\(\)](#) action and the AVTransport service supports controlling of the transport for DRM-controlled content, otherwise it is not allowed.

The DRMState state variable is used by instances of the AVTransport service to inform control points about process failures and other AVTransport instance state changes that can occur independently of AVTransport actions.

Table 17 below details the allowed values for the DRMState state variable:

Table 17 — allowedValueList for DRMState

| Allowed Value | R/A | Description |
|--|-------------------|---|
| " OK " | R | This setting indicates that DRM related processing has completed successfully. This setting also applies, to items which do not have DRM protection applied. |
| " UNKNOWN " | A | This setting indicates that the state of the DRM subsystem is not known. For example, this would be the case when the DRM system is first initialized and the content-binary location has not yet been specified. |
| " PROCESSING_CONTENT_KEY " ^a | A | This setting indicates that the DRM system is currently deriving a decryption key to decrypt a content-binary. |
| " CONTENT_KEY_FAILURE " | A | This setting indicates that a content key needed to start or continue media transport was either not received or has failed verification. |
| " ATTEMPTING_AUTHENTICATION " ^a | A | This setting indicates that the authentication process is currently in progress, but has not yet completed. |
| " FAILED_AUTHENTICATION " | A | This setting indicates that an attempted authentication process has failed. |
| " NOT_AUTHENTICATED " | A | This setting indicates that authentication has not yet taken place or that a previously successful authentication has transitioned to a non-authenticated state for example due to a timeout or other condition. |
| " DEVICE_REVOCATION " | A | This setting indicates that the DRM system has detected that this device has been revoked, i.e. the device has been explicitly prohibited from accessing any DRM protected content on this server. |
| " DRM_SYSTEM_NOT_SUPPORTED " | A | This setting indicates that the device cannot decrypt the content-binary since it does not support the DRM technology used to encode this content. |
| " LICENSE_DENIED " | A | This setting indicates that this device is not able to obtain any license for this content-binary. |
| " LICENSE_EXPIRED " | A | This setting indicates that a previously valid license obtained by this device has expired. |
| " LICENSE_INSUFFICIENT " | A | This setting indicates that a license granted to the device does not permit an attempted operation on the content-binary. |
| ^a This setting indicates a transient DRM state. As DRM processing continues, this state would be expected to transition to a non-transient state. | | |

5.2.31 SyncOffset

This conditionally required state variable shall be supported if the AVTransport service implements [GetSyncOffset\(\)](#) and [SetSyncOffset\(\)](#) actions, otherwise it is not allowed. Note that if either action is implemented, both shall be implemented. This state variable indicates a high-precision time offset that is used to adjust the actual timing of the ConnectionManager *CLOCKSYNC* feature for a specific instance. Its value is used to automatically and uniformly shift all of the presentation time values that are associated with the ConnectionManager *CLOCKSYNC* feature. Some examples include the [RelativePresentationTime](#) input argument of the [SyncPlay\(\)](#) action or the presentation timestamps associated with a content stream.

A positive value indicates that the relevant time-of-day value(s) shall be increased by the specified amount, thus, causing a slight delay. Conversely, a negative value indicates that the relevant time-of-day value(s) shall be decreased by the specified amount, thus, causing the associated effect to occur sooner than would have otherwise occurred.

The SyncOffset state variable is of type string and has the following format:

```

duration      ::= ['-']P' time
time          ::= HH ':' MM ':' SS '.' MilliS MicroS NanoS
HH            ::= 2DIGIT (* 00-23 *)
MM            ::= 2DIGIT (* 00-59 *)
SS            ::= 2DIGIT (* 00-59 *)
MilliS        ::= 3DIGIT
MicroS        ::= 3DIGIT
NanoS         ::= 3DIGIT
    
```

Table 18 — allowedValueRange for SyncOffset

| | Value | R/A |
|----------------|----------------|----------|
| <u>minimum</u> | >= P00:00:00 | <u>R</u> |
| <u>maximum</u> | Vendor defined | <u>R</u> |
| <u>step</u> | Vendor defined | <u>R</u> |

5.2.32 A_ARG_TYPE SeekMode

This required state variable is introduced to provide type information for the Unit argument in action Seek(). It indicates the allowed units in which the amount of seeking to be performed is specified. It can be specified as a time (relative or absolute), a count (relative or absolute), a track number, a tape-index (for example, for tapes with an indexing facility; relative or absolute) or even a video frame (relative or absolute). A device vendor may implement a subset of the allowed value list of this state variable. Only the value "TRACK_NR" is required.

Table 19 — allowedValueList for A_ARG_TYPE SeekMode

| Value | R/A |
|---------------------------|----------|
| " <u>TRACK_NR</u> " | <u>R</u> |
| " <u>ABS_TIME</u> " | <u>A</u> |
| " <u>REL_TIME</u> " | <u>A</u> |
| " <u>ABS_COUNT</u> " | <u>A</u> |
| " <u>REL_COUNT</u> " | <u>A</u> |
| " <u>CHANNEL_FREQ</u> " | <u>A</u> |
| " <u>TAPE-INDEX</u> " | <u>A</u> |
| " <u>REL_TAPE-INDEX</u> " | <u>A</u> |
| " <u>FRAME</u> " | <u>A</u> |
| " <u>REL_FRAME</u> " | <u>A</u> |
| Vendor-defined | <u>X</u> |

5.2.33 A_ARG_TYPE SeekTarget

This required state variable is introduced to provide type information for the Target argument in action Seek(). It indicates the target position of the Seek() action, in terms of units defined by state variable A_ARG_TYPE SeekMode. The data type of this variable is **string**. However, depending on the actual seek mode used, it shall contain string representations of values as defined in Table 20 below:

Table 20 — Format of **A_ARG_TYPE SeekTarget**

| Value of <u>A_ARG_TYPE SeekMode</u> | Format of <u>A_ARG_TYPE SeekTarget</u> |
|--|---|
| " <u>TRACK_NR</u> " | <u>ui4</u> |
| " <u>ABS_TIME</u> " | Formatted as specified in subclause 5.2.16 |
| " <u>REL_TIME</u> " | Formatted as specified in subclause 5.2.16 |
| " <u>ABS_COUNT</u> " | <u>ui4</u> |
| " <u>REL_COUNT</u> " | <u>i4</u> |
| " <u>CHANNEL_FREQ</u> " | <u>float</u> , expressed in Hz. |
| " <u>TAPE-INDEX</u> " | <u>ui4</u> |
| " <u>REL_TAPE-INDEX</u> " | <u>i4</u> |
| " <u>FRAME</u> " | <u>ui4</u> |
| " <u>REL_FRAME</u> " | <u>i4</u> |

Supported ranges of these integer, time or float values are device-dependent.

5.2.34 **A_ARG_TYPE InstanceID**

This required state variable is introduced to provide type information for the **InstanceID** input argument present in all AVTransport actions. It identifies the virtual instance of the AVTransport service to which the action applies. A valid **InstanceID** is obtained from a *factory* method in the ConnectionManager service: the **ConnectionManager::PrepareForConnection()** action.

If the device's ConnectionManager does not implement the optional **ConnectionManager::PrepareForConnection()** action, special value "**0**" shall be used for the **InstanceID** input argument. In such a case, the device implements a single static AVTransport instance, and only one stream can be controlled and sent (or received) at any time.

5.2.35 **A_ARG_TYPE DeviceUDN**

This conditionally required state variable shall be supported if the AVTransport service implements the **SetStateVariables()** action, otherwise it is not allowed. The state variable is introduced to provide type information for the **AVTransportUDN** argument in that action. It is a **string** value containing the UDN of the device.

5.2.36 **A_ARG_TYPE ServiceType**

This conditionally required state variable shall be supported if the AVTransport service implements the **SetStateVariables()** action, otherwise it is not allowed. The state variable is introduced to provide type information for the **ServiceType** argument in that action. It is a **string** value containing the service type and version number of a service such as "AVTransport:3".

5.2.37 **A_ARG_TYPE ServiceID**

This conditionally required state variable shall be supported if the AVTransport service implements the **SetStateVariables()** action, otherwise it is not allowed. The state variable is introduced to provide type information for the **ServiceID** argument in that action. It is a **string** value containing the service ID of a service.

5.2.38 **A_ARG_TYPE StateVariableValuePairs**

This conditionally required state variable shall be supported if the AVTransport service implements the **GetStateVariables()** and **SetStateVariables()** actions, otherwise it is not allowed. Note that if either action is implemented, both shall be implemented. The state variable is introduced to provide type information for the **StateVariableValuePairs** argument in that action. This state variable contains a list of state variable names and their values. The list of state variables whose name/value pair is requested is given by another argument to the action. The structure of the **StateVariableValuePairs** argument is defined in [4].

The following *stateVariableValuePairs* XML Document illustrates a typical example of the schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<stateVariableValuePairs
  xmlns="urn:schemas-upnp-org:av:avs"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:schemas-upnp-org:av:avs
    http://www.upnp.org/schemas/av/avs.xsd">
  <stateVariable variableName="CurrentPlayMode">
    NORMAL
  </stateVariable>
  <stateVariable variableName="CurrentTrack">
    3
  </stateVariable>
  <!-- More state variable value pairs can be inserted here -->
</stateVariableValuePairs>
```

The relevant variable names shall be either all or a subset (as required) of the defined AVTransport state variables except for *LastChange* and any *A_ARG_TYPE_xxx* state variables.

5.2.39 *A_ARG_TYPE_StateVariableList*

This conditionally required state variable shall be supported if the AVTransport service implements the *GetStateVariables()* and *SetStateVariables()* actions, otherwise it is not allowed. Note that if either action is implemented, both shall be implemented. The state variable is introduced to provide type information for the *StateVariableList* argument in that action. It is a CSV list of state variable names. This variable may contain one or more (as required) of the defined AVTransport state variable names except *LastChange* and any *A_ARG_TYPE_xxx* state variable names. The asterisk ("*") can be specified to indicate all relevant variable names (excluding *LastChange* and any *A_ARG_TYPE_xxx* state variables.)

5.2.40 *A_ARG_TYPE_PlaylistData*

This conditionally required state variable shall be supported if the AVTransport service implements either the *SetStaticPlaylist()* or *SetStreamingPlaylist()* actions, otherwise it is not allowed. This state variable is introduced to provide a chunk of a playlist document to the device. It is a **string** value.

5.2.41 *A_ARG_TYPE_PlaylistDataLength*

This conditionally required state variable shall be supported if the AVTransport service implements either the *SetStaticPlaylist()* or *SetStreamingPlaylist()* actions, otherwise it is not allowed. This state variable is introduced to indicate the chunk's length of the playlist document. It is a **ui4** value. The data length shall match the XML-escaped representation of the associated **string** value.

5.2.42 *A_ARG_TYPE_PlaylistOffset*

This conditionally required state variable shall be supported if the AVTransport service implements the *SetStaticPlaylist()* action, otherwise it is not allowed. This state variable is introduced to provide a zero-based offset into the playlist document being passed to the renderer. The state variable contains a **ui4** value. The offset value shall correspond to the the XML-escaped representation of the aggregate playlist.

5.2.43 *A_ARG_TYPE_PlaylistTotalLength*

This conditionally required state variable shall be supported if the AVTransport service implements the *SetStaticPlaylist()* action, otherwise it is not allowed. This state variable is introduced to provide the total length of the entire playlist document. The state variable contains a **ui4** value. The total length shall match the XML-escaped representation of the aggregate playlist.

5.2.44 A_ARG_TYPE PlaylistMIMEType

This conditionally required state variable shall be supported if the AVTransport service implements either the [SetStaticPlaylist\(\)](#) or [SetStreamingPlaylist\(\)](#) actions, otherwise it is not allowed. This state variable is introduced to provide the MIME type of the playlist provided to the device. The value of this argument corresponds to the contents of the [res@protocolInfo](#) property 3rd field. The state variable contains a [string](#) value.

5.2.45 A_ARG_TYPE PlaylistExtendedType

This conditionally required state variable shall be supported if the AVTransport service implements either the [SetStaticPlaylist\(\)](#) or [SetStreamingPlaylist\(\)](#) actions, otherwise it is not allowed. This state variable is introduced to provide extended type information of the playlist provided to the device. The value of this argument corresponds to the contents of the [res@protocolInfo](#) property 4th field. The state variable contains a [string](#) value.

5.2.46 A_ARG_TYPE PlaylistStep

This conditionally required state variable shall be supported if the AVTransport service implements the [SetStreamingPlaylist\(\)](#) action, otherwise it is not allowed. This state variable is introduced to provide step information for a streaming playlist operation. The state variable contains a [string](#) value.

Table 21 — allowedValueList for A_ARG_TYPE PlaylistStep

| Value | R/A | Description |
|----------------------------|-------------------|--|
| "Initial" | R | Indicates that this is the start of streaming playlist operation. |
| "Continue" | R | Indicates that this is a continuation of a streaming playlist operation. |
| "Stop" | R | Indicates that the current streaming playlist operation will end when all pending playlist data at the device is consumed. |
| "Reset" | R | Indicates that processing of the current streaming playlist ends immediately. Any pending playlist data for the streaming playlist is discarded. |
| "Replace" | A | Indicates the current streaming playlist contents be replaced with the contents provided by this operation. If the playlist is being actively rendered, then the current playback selection shall not be interrupted. Processing of the replacement playlist tracks shall begin at the next track transition. Additional playlist entries may be delivered using the "Continue" operation. |

5.2.47 A_ARG_TYPE PlaylistType

This conditionally required state variable shall be supported if the AVTransport service implements the [GetPlaylistInfo\(\)](#) action, otherwise it is not allowed. This state variable describes the playlist types supported by the implementation. The state variable contains a [string](#) value.

Table 22 — allowedValueList for A_ARG_TYPE PlaylistType

| Value | R/A |
|---------------------------------------|--|
| "Static" | CR , required if the information returned by the GetPlaylistInfo() action refers to a static (non-streaming) playlist. Not allowed otherwise. If the SetStaticPlaylist() action is supported, then this allowed value shall be provided. |
| "StaticPIContents" | CA , allowed if the information returned by the GetPlaylistInfo() action refers to a static (non-streaming) playlist. Otherwise not allowed. If the SetStaticPlaylist() action is supported, then this allowed value may be provided. If this allowed value is implemented then the <playlistContents> element of the A_ARG_TYPE PlaylistInfo state variable shall be supported. |
| "Streaming" | CR , required if the information returned by the GetPlaylistInfo() action refers to a streaming playlist. Not allowed otherwise. If the SetStreamingPlaylist() action is supported, then this allowed value shall be provided. |
| "StreamingPIContents" | CA , allowed if the information returned by the GetPlaylistInfo() action refers to a streaming playlist. Otherwise not allowed. If the SetStreamingPlaylist() action is supported, then this allowed value may be provided. If this allowed value is implemented then the <playlistContents> element of the A_ARG_TYPE PlaylistInfo state variable shall be supported. |

5.2.48 A_ARG_TYPE_PlaylistInfo

This conditionally required state variable shall be supported if the AVTransport service implements the GetPlaylistInfo() action, otherwise it is not allowed. This state variable is a document detailing whether the implementation can play the indicated item formats. The state variable contains a **string** value. The structure of the document is as follows:

A_ARG_TYPE_PlaylistInfo state variable for Streaming Playlist operation:

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

  <streamingPlaylistInfo>
    <playlistState>
      Idle | Ready | Active
    </playlistState>

    <playlistChunkLengthMax>
      maximum allowed length of a playlist chunk, in its XML-escaped form
    </playlistChunkLengthMax>

    <playlistDataLengthUsed>
      length of playlist data currently stored at device, in its XML-escaped form
    </playlistDataLengthUsed>

    <playlistTotalLengthAvail>
      length of playlist storage currently available on device, in its XML-escaped form
    </playlistTotalLengthAvail>

    <playlistTrackMin>
      current minimum available track number for this playlist
    </playlistTrackMin>

    <playlistTrackMax>
      current maximum available track number for this playlist
    </playlistTrackMax>

    <playlistCurrentFormat>
      <contentType
        MIMEType="playlist MIME type"
        extendedType="res@protocolInfo 4th field value" />
    </playlistCurrentFormat>

    <playlistAllowedFormats>
      <contentType
        MIMEType="playlist MIME type"
        extendedType="res@protocolInfo 4th field value" />
      <!-- Additional supported streaming <contentType> elements -->
    </playlistAllowedFormats>

    <!-- PlaylistType == StreamingPlContents -->
    <!-- See GetPlaylistInfo() and A_ARG_TYPE_PlaylistInfo -->

    <playlistContents currentTrack="current track number">
      streaming playlist contents
    </playlistContents>

  </streamingPlaylistInfo>
</playlistInfo>
```

A_ARG_TYPE_PlaylistInfo state variable for Static Playlist operation:

```

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

  <staticPlaylistInfo>

    <playlistState>
      Idle | Incomplete | Ready | Active
    </playlistState>

    <playlistChunkLengthMax>
      maximum allowed length of a playlist chunk
    </playlistChunkLengthMax>

    <playlistTotalLengthMax>
      length of playlist storage currently available on device
    </playlistTotalLengthMax>

    <playlistCurrentFormat>
      <contentType
        MIMEType="playlist MIME type"
        extendedType="res@protocolInfo 4th field value" />
    </playlistCurrentFormat>

    <playlistAllowedFormats>
      <contentType
        MIMEType="playlist MIME type"
        extendedType="res@protocolInfo 4th field value" />
      <!-- Additional supported static <contentType> elements -->
    </playlistAllowedFormats>

    <!-- PlaylistType == StaticPlContents -->
    <!-- See GetPlaylistInfo() and A_ARG_TYPE_PlaylistInfo -->

    <playlistContents currentObjectID="current playlist item identifier">
      Static (XML) playlist contents
    </playlistContents>

  </staticPlaylistInfo>
</playlistInfo>

```

xml

Allowed. Case sensitive.

playlistInfo

Required. Shall include a namespace declaration for the PlaylistInfo schema ("urn:schemas-upnp-org:av:rpl"). Shall include zero or one of the following elements. This namespace defines the following elements and attributes:

streamingPlaylistInfo

Required. <XML>, Provides information about a streaming playlist operation. It consists of a series of dependent elements that describe various aspects of a streaming playlist operation.

playlistState

Required. xsd:string, Provides state information for a streaming playlist operation. The possible states for the playlist state are:

Idle – The device instance is not currently processing a streaming playlist.

Ready - A streaming playlist operation is in progress and sufficient playlist data has been sent to the device instance to start playback. However, a *Play()* action has not been invoked.

Active – The device instance is currently playing a streaming playlist.

playlistChunkLengthMax

Required. xsd:unsignedInt, Provides the maximum allowable length for a playlist data chunk.

playlistDataLengthUsed

Required. xsd:unsignedInt, Provides the number of playlist bytes currently available (stored) at the device for this instance.

playlistTotalLengthAvail

Required. xsd:unsignedInt, Provides the maximum number of playlist bytes that will currently be accepted by the device for this instance.

playlistTrackMin

Required. xsd:unsignedInt, Provides the current minimum track number that the media renderer has in its playlist buffer for this instance

playlistTrackMax

Required. xsd:unsignedint, Provides the current maximum track number that the media renderer has in its playlist buffer for this instance.

playlistCurrentFormat

Required. <XML>, Provides MIME type and extended type information for the current playlist. If the device is not processing a playlist, this element shall be empty.

contentType

Allowed. <XML>, Identifies the MIME type and extended type information for the playlist.

@MIMETYPE

Required. xsd:string, Provides the MIME type for the playlist.

@extendedType

Required. xsd:string, Provides the extended type information for the playlist. This information should contain information typically provided by a [res@protocolInfo](#) property 4th field element.

playlistAllowedFormats

Required. <XML>, Provides zero or more <contentType> elements indicating streaming playlist types supported by the device.

contentType

Allowed. <XML>, See <contentType> element as defined above.

playlistContents

Conditionally required. xsd:string, Element shall be present if the [PlaylistType](#) argument of the [GetPlaylistInfo\(\)](#) action is set to "[StreamingPILContents](#)" (see subclause 5.2.47 for further details), otherwise not allowed. This element value is a copy of the device streaming playlist. If the current <playlistState> element value is not "[Ready](#)" or "[Active](#)", this element shall be empty. Playlist data returned by this element's value shall be XML escaped.

@currentTrack

Required. xsd:unsignedint, Provides the current track number for the streaming playlist. The track number reported shall be within the track number range reported by the <playlistTrackMin> and <playlistTrackMax> elements.

staticPlaylistInfo

Required. <XML>, Provides information about a static playlist operation. It consists of a series of dependent elements that describe various aspects of a static playlist operation.

playlistState

Required. xsd:string, Provides state information for a static playlist operation. The possible states for the playlist state are:

[Idle](#) – The device instance is not currently processing a static playlist.

[Incomplete](#) – A static playlist operation has been started on the device instance but the full playlist has not been sent to the device instance.

[Ready](#) - A static playlist operation is in progress and the full playlist has been sent and accepted by the device. However, a [Play\(\)](#) action has not been invoked.

[Active](#) – The device instance is currently playing a static playlist.

playlistChunkLengthMax

Required. xsd:unsignedInt, Provides the maximum allowable length for a playlist data chunk for this device instance.

playlistTotalLengthMax

Required. xsd:unsignedInt, Provides the maximum number of playlist bytes that will currently be accepted by the device for this instance.

playlistCurrentFormat

Required. <XML>, Provides MIME type and extended type information for the current playlist. If the device is not processing a playlist, this element shall be empty.

contentType

Allowed. <XML>, See <contentType> element as defined above.

playlistAllowedFormats

Required. <XML>, Provides zero or more <contentType> elements indicating static playlist types supported by the device.

contentType

Allowed. <XML>, See <contentType> element as defined above.

playlistContents

Conditionally required. xsd:string, Element shall be present if the PlaylistType argument of the GetPlaylistInfo() action is set to "StaticPlContents" (see subclause 5.2.47 for further details), otherwise not allowed. This element value is a copy of the device static playlist. If the current <playlistState> element value is not "Ready" or "Active", this element shall be empty. Playlist data returned by this element's value shall be XML escaped.

@currentObjID

Required. xsd:string, Provides an identifier for the currently playing playlist object. For XML formatted playlists (DIDL-Lite) the identifier value corresponds to the @id property. Identifier values for non-DIDL-Lite formats are playlist format specific.

5.2.49 A ARG TYPE PlaylistStartObjID

This conditionally required state variable shall be supported if the AVTransport service implements the SetStaticPlaylist() action, otherwise it is not allowed. This argument provides a starting object @id property value for playlists which employ object linking properties. The state variable contains a string value. For playlists that do not employ object linking properties this state variable should be set to "".

5.2.50 A ARG TYPE PlaylistStartGroupID

This conditionally required state variable shall be supported if the AVTransport service implements the SetStaticPlaylist() action, otherwise it is not allowed. This argument provides a starting target group ID objectLink@groupID value for playlists which employ object linking properties. The state variable contains a string value. For playlists that do not employ object linking properties this state variable should be set to "".

5.2.51 A ARG TYPE SyncOffsetAdj

This conditionally required state variable shall be supported if the AVTransport service implements the AdjustSyncOffset() action, otherwise it is not allowed. This state variable indicates a high-precision time offset that is used to adjust the actual timing of the ConnectionManager CLOCKSYNC feature for a specific instance. Its value is used to automatically and uniformly shift all of the presentation time values that are associated with the ConnectionManager CLOCKSYNC feature. Some examples include the RelativePresentationTime input argument of the SyncPlay() action or the presentation timestamps associated with a content stream.

A positive value indicates that the relevant time-of-day value(s) shall be increased by the specified amount, thus, causing a slight delay. Conversely, a negative value indicates that the relevant time-of-day value(s) shall be decreased by the specified amount, thus, causing the associated effect to occur sooner than would have otherwise occurred.

The A ARG TYPE SyncOffsetAdj state variable is of type string and has the following format:

```
duration      ::= ['-']'P' time
time          ::= HH ':' MM ':' SS '.' Millis MicroS NanoS
HH           ::= 2DIGIT (* 00-23 *)
MM           ::= 2DIGIT (* 00-59 *)
```

```

SS          ::= 2DIGIT (* 00-59 *)
MilliS     ::= 3DIGIT
MicroS     ::= 3DIGIT
NanoS      ::= 3DIGIT
    
```

5.2.52 A ARG TYPE PresentationTime

This conditionally required state variable shall be implemented if the AVTransport service implements the [SyncPlay\(\)](#), [SyncStop\(\)](#), or [SyncPause\(\)](#) actions, otherwise it is not allowed. This state variable is introduced to provide type information for the [ReferencePresentationTime](#) and other similar input arguments for AVTransport actions related to *CLOCKSYNC feature*. It represents a high-precision point in time (corresponding to a specific time on a specific day) that is used to designate the exact time when certain time-sensitive operations are to be performed.

The [A ARG TYPE PresentationTime](#) state variable is of type **string** and represents a point in time as specified by [36]. The format is as follows:

```

date-time      ::= yyyy '-' mm '-' dd T-labeled-time
T-labeled-time ::= 'T' time 'Z'
time           ::= HH ':' MM ':' SS '.' MilliS MicroS NanoS
HH            ::= 2DIGIT (* 00-23 *)
MM            ::= 2DIGIT (* 00-59 *)
SS            ::= 2DIGIT (* 00-59 *)
MilliS        ::= 3DIGIT
MicroS        ::= 3DIGIT
NanoS         ::= 3DIGIT
    
```

5.2.53 A ARG TYPE ClockId

This conditionally required state variable shall be implemented if the AVTransport service implements the [SyncPlay\(\)](#), [SyncStop\(\)](#), or [SyncPause\(\)](#) actions, otherwise it is not allowed. This state variable is introduced to provide type information for the [ReferenceClockId](#) input argument for AVTransport actions related to *CLOCKSYNC feature*. It represents a unique **string** identifier for the <deviceClockInfo> instance which in turn identifies the clock sync protocol, master clock, clock accuracy and supported timestamp mechanisms (if any). The [A ARG TYPE ClockId](#) value shall be one of the deviceClockInfo@id values, which are declared in the [ConnectionManager::FeatureList](#) state variable. In this case the referenced ConnectionManager service has the same parent device as this AVTransport service.

5.3 Eventing and Moderation

5.3.1 Eventing and Moderation Overview

Table 23 — Event Moderation

| Variable Name | Evented | Moderated Event | Min Event Interval ^a (seconds) | Logical Combination | Min Delta per Event ^b |
|--|--------------------|--------------------|---|---------------------|----------------------------------|
| TransportState | NO | NO | | | |
| TransportStatus | NO | NO | | | |
| CurrentMediaCategory | NO | NO | | | |
| PlaybackStorageMedium | NO | NO | | | |
| RecordStorageMedium | NO | NO | | | |
| PossiblePlaybackStorageMedia | NO | NO | | | |
| PossibleRecordStorageMedia | NO | NO | | | |
| CurrentPlayMode | NO | NO | | | |
| TransportPlaySpeed | NO | NO | | | |
| RecordMediumWriteStatus | NO | NO | | | |
| PossibleRecordQualityModes | NO | NO | | | |

| Variable Name | Evented | Moderated Event | Min Event Interval ^a (seconds) | Logical Combination | Min Delta per Event ^b |
|---|----------------------------|----------------------------|---|---------------------|----------------------------------|
| <u>CurrentRecordQualityMode</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>NumberOfTracks</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>CurrentTrack</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>CurrentTrackDuration</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>CurrentMediaDuration</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>CurrentTrackURI</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>CurrentTrackMetaData</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>AVTransportURI</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>AVTransportURIMetaData</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>NextAVTransportURI</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>NextAVTransportURIMetaData</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>RelativeTimePosition</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>AbsoluteTimePosition</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>RelativeCounterPosition</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>AbsoluteCounterPosition</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>CurrentTransportActions</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>LastChange</u> | <u>YES</u> | <u>YES</u> | 0.2 | | |
| <u>DRMState</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>SyncOffset</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_SeekMode</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_SeekTarget</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_InstanceID</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_DeviceUDN</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_ServiceType</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_ServiceID</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_StateVariableValuePairs</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistData</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistDataLength</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistOffset</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistTotalLength</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistMIMEType</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistExtendedType</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistStep</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistType</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistInfo</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistStartObjID</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PlaylistStartGroupID</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_StateVariableList</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_PresentationTime</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_ClockID</u> | <u>NO</u> | <u>NO</u> | | | |
| <u>A_ARG_TYPE_SyncOffsetAdj</u> | <u>NO</u> | <u>NO</u> | | | |

| Variable Name | Evented | Moderated Event | Min Event Interval ^a (seconds) | Logical Combination | Min Delta per Event ^b |
|--|------------|-----------------|---|---------------------|----------------------------------|
| <i>Non-standard state variables implemented by a UPnP vendor go here</i> | <i>TBD</i> | <i>TBD</i> | <i>TBD</i> | <i>TBD</i> | <i>TBD</i> |
| ^a Max event rate is determined by N , where $Rate = 1/N$, where N is the Min Event Interval in seconds. ^b $(N) * (allowedValueRange Step)$. | | | | | |

Note that non-standard state variables shall also be evented through the LastChange event mechanism.

5.3.2 Event Model

Since the AVTransport service supports multiple virtual instances (via the InstanceID argument included in each action), the traditional UPnP eventing model is unable to differentiate between multiple instances of the same state variable. Therefore, the AVTransport service event model defines a specialized state variable (LastChange) that is used exclusively for eventing individual state changes. In this model, the LastChange state change is the only variable that is evented using the standard UPnP event mechanism. All other state variables, except the position-related state variables listed as a) to d) below, are indirectly evented via the LastChange state variable. (Note: A_ARG_TYPE_ state variables are not evented, either directly or indirectly). More details about the LastChange-based event mechanism can be found in the RenderingControl service specification [21], subclause 5.3.1.

The AVTransport service contains various state variables that, during certain transport states, change almost continuously. The following variables are therefore not evented via LastChange:

- a) RelativeTimePosition
- b) AbsoluteTimePosition
- c) RelativeCounterPosition
- d) AbsoluteCounterPosition

Each control point can poll for these values at a rate appropriate for their application, whenever they need to. For example, a control point can invoke GetPositionInfo() every second when the TransportState is “PLAYING”, “RECORDING” or “TRANSITIONING”. This is more efficient and flexible than requiring event notifications to be sent to all subscribing control points, in all cases.

Evented state variables shall only be evented if their value actually changes. Writing the same value to a state variable does not generate an event. For example, a transition from the state “PLAYING” to the state “PLAYING” with a different speed does not generate an event for state variable TransportState (“PLAYING” → “PLAYING”). However, this transition will generate an event for the state variable TransportPlaySpeed. If a moderated state variable is evented and it returns the same value, this means that within the moderation time, its value has actually changed and then changed back to its previous value.

5.4 Actions

5.4.1 Action Overview

Table 24 — Actions

| Name | R/A ^a | Control Point R/A ^b |
|--|--|---------------------------------------|
| <u>SetAVTransportURI()</u> | <u>R</u> | <u>R</u> |
| <u>SetNextAVTransportURI()</u> | <u>A</u> | <u>A</u> |
| <u>GetMediaInfo()</u> | <u>R</u> | <u>A</u> |
| <u>GetMediaInfo_Ext()</u> | <u>R</u> | <u>A</u> |
| <u>GetTransportInfo()</u> | <u>R</u> | <u>A</u> |
| <u>GetPositionInfo()</u> | <u>R</u> | <u>A</u> |
| <u>GetDeviceCapabilities()</u> | <u>R</u> | <u>A</u> |
| <u>GetTransportSettings()</u> | <u>R</u> | <u>A</u> |
| <u>Stop()</u> | <u>R</u> | <u>R</u> ^c |
| <u>Play()</u> | <u>R</u> | <u>R</u> ^c |
| <u>Pause()</u> | <u>A</u> | <u>A</u> |
| <u>Record()</u> | <u>A</u> | <u>A</u> |
| <u>Seek()</u> | <u>R</u> | <u>A</u> |
| <u>Next()</u> | <u>R</u> | <u>A</u> |
| <u>Previous()</u> | <u>R</u> | <u>A</u> |
| <u>SetPlayMode()</u> | <u>A</u> | <u>A</u> |
| <u>SetRecordQualityMode()</u> | <u>CA</u> ^d | <u>A</u> |
| <u>GetCurrentTransportActions()</u> | <u>A</u> | <u>A</u> |
| <u>GetDRMState()</u> | <u>CR</u> ^e | <u>A</u> |
| <u>GetStateVariables()</u> | <u>CR</u> ^e | <u>A</u> |
| <u>SetStateVariables()</u> | <u>CR</u> ^e | <u>A</u> |
| <u>GetSyncOffset()</u> | <u>CR</u> ^e | <u>A</u> |
| <u>AdjustSyncOffset()</u> | <u>CR</u> ^e | <u>A</u> |
| <u>SetSyncOffset()</u> | <u>CR</u> ^e | <u>A</u> |
| <u>SyncPlay()</u> | <u>CR</u> ^e | <u>A</u> |
| <u>SyncStop()</u> | <u>CR</u> ^e | <u>A</u> |
| <u>SyncPause()</u> | <u>CA</u> ^d | <u>A</u> |
| <u>SetStaticPlaylist()</u> | <u>A</u> | <u>A</u> |
| <u>SetStreamingPlaylist()</u> | <u>A</u> | <u>A</u> |
| <u>GetPlaylistInfo()</u> | <u>CR</u> ^e | <u>A</u> |
| <i>Non-standard actions implemented by a UPnP vendor go here</i> | <u>X</u> | <u>X</u> |

| Name | R/A ^a | Control Point R/A ^b |
|---|------------------|--------------------------------|
| <p>^a For a device this column indicates whether the action shall be implemented or not, where <u>R</u> = required, <u>A</u> = allowed, <u>CR</u> = conditionally required, <u>CA</u> = conditionally allowed, <u>X</u> = non-standard, add <u>-D</u> when deprecated (e.g., <u>R-D</u>, <u>A-D</u>).</p> <p>^b For a control point this column indicates whether a control point shall be capable of invoking this action, where <u>R</u> = required, <u>A</u> = allowed, <u>CR</u> = conditionally required, <u>CA</u> = conditionally allowed, <u>X</u> = non-standard, add <u>-D</u> when deprecated (e.g., <u>R-D</u>, <u>A-D</u>).</p> <p>^c Required only if the control point implements interaction with the AVTransport service.</p> <p>^d See action description for conditions under which implementation of this action is allowed. If the condition is not met implementation of this action is not allowed.</p> <p>^e See action description for conditions under which implementation of this action is required.</p> | | |

Note that non-standard actions shall be implemented in such a way that they do not interfere with the basic operation of the AVTransport service; that is: these actions shall be allowed, and shall not need to be invoked for the AVTransport service to operate normally.

5.4.2 SetAVTransportURI()

This required action specifies the URI of the resource to be controlled by the specified AVTransport instance. It is recommended that the AVTransport service checks the MIME-type of the specified resource when executing this action. For AVTransport instances that control the transport of DRM-controlled content, the authentication process is also recommended to start as a result of executing this action. The SetAVTransportURI() action is successful even when a necessary authentication or revocation check cannot be completed before the expiration of time allotted for the completion of the SetAVTransportURI() action. In the case of AVTransport instances that control the transport of DRM-controlled content, the subsequent detection of conditions that need to be communicated to the control point, like an authentication failure, a revocation condition, etc. are indicated via the DRMState state variable. A control point can supply metadata associated with the specified resource, using a DIDL-Lite XML Fragment (defined in the ContentDirectory service specification), in argument CurrentURIMetaData. If supported by the AVTransport instance, this metadata is stored in a state variable, and returned as output argument as part of the GetMediaInfo() action and the GetMediaInfo_Ext() action. If a control point does not want to use this feature it can supply the empty string for the CurrentURIMetaData argument.

A resource may have descriptions of the embedded media components associated with it, as indicated by the metadata property upnp:resExt::componentInfo (see the ContentDirectory service specification). This description can be used by the device implementation to provide an extended user experience (for example, by offering a choice between different subtitle languages). Some associated components can have their own resource (e.g. URI), these resources are known as secondary resources. A control point can supply the metadata including the secondary resources in the CurrentURIMetadata argument. The AVTransport service implementation can use this information to offer a selection of alternative playback components to a control point.

If a playback component is a secondary resource, the device shall support synchronized playback of these resources upon invocation of the Play() action. Similarly, actions such as Stop() and Pause() shall affect all the secondary resources as well. If the AVTransport service implementation does not support this feature, or if the control point does not supply the metadata, then only the URI of the primary resource will be played back. As all secondary resources are synchronized with the primary resource during playback, implementations can implement retrieval of the state information of an AVTransport instance (via the actions GetMediaInfo(), GetMediaInfo_Ext(), GetTransportInfo() and GetPositionInfo()) based on the state of the primary resource.

Note: The time resolution of what is considered synchronized playback is dependent on the application.

5.4.2.1 Arguments

Table 25 — Arguments for SetAVTransportURI()

| Argument | Direction | relatedStateVariable |
|---------------------------|-----------|-------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE InstanceID</u> |
| <u>CurrentURI</u> | <u>IN</u> | <u>AVTransportURI</u> |
| <u>CurrentURIMetaData</u> | <u>IN</u> | <u>AVTransportURIMetaData</u> |

5.4.2.2 Dependency on State

None.

5.4.2.3 Effect on State

Depending on the URI, the number of tracks available on this instance could have changed. For example, if the URI points to a single audio file, state variable NumberOfTracks changes

to 1. However, if the URI points to an audio playlist, state variable *NumberOfTracks* changes to the number of entries in the playlist.

If the renderer fails to locate or download the resource at the URI the *TransportState* shall change to “*STOPPED*”. If the current transport state is “*PLAYING*”, and it would take a noticeable amount of time before a human user would actually see or hear the media at the new URI playing, the AVTransport may temporarily go to the “*TRANSITIONING*” state before going back to “*PLAYING*”. This might be appropriate for devices that need to start buffering or completely download the media before playback can start. If the current transport state is “*NO MEDIA PRESENT*” the transport state changes to “*STOPPED*”. In all other cases, this action does not change the transport state of the specified instance.

5.4.2.4 Errors

Table 26 — Error Codes for *SetAVTransportURI()*

| ErrorCode | errorDescription | Description |
|-----------|-------------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 714 | Illegal MIME-type | The specified resource has a MIME-type which is not supported by the AVTransport service. |
| 715 | Content ‘BUSY’ | This indicates that the resource is already in use at this time. |
| 716 | Resource not found | The specified resource cannot be found in the network. |
| 718 | Invalid <i>InstanceID</i> | The specified <i>InstanceID</i> is invalid for this AVTransport. |
| 719 | DRM error | The action failed because an unspecified DRM error occurred. |
| 720 | Expired content | The action failed because the content use validity interval has expired. |
| 721 | Non-allowed use | The action failed because the requested content use is disallowed. |
| 722 | Can’t determine allowed uses | The action failed because the allowed content uses cannot be verified. |
| 723 | Exhausted allowed use | The action failed because the number of times this content has been used as requested has reached the maximum allowed number of uses. |
| 724 | Device authentication failure | The action failed because of a device authentication failure between the media source device and the media sink device. |
| 725 | Device revocation | The action failed because either the media source device or the media sink device has been revoked. |
| 737 | No DNS Server | The DNS Server is not available (HTTP error 503). |
| 738 | Bad Domain Name | Unable to resolve the Fully Qualified Domain Name (HTTP error 502). |
| 739 | Server Error | The server that hosts the resource is unreachable or unresponsive (HTTP error 404/410). |

5.4.3 *SetNextAVTransportURI()*

This allowed action specifies the URI of the resource to be controlled when the playback of the current resource (set earlier via *SetAVTransportURI()*) finishes. This action enables a device to *prefetch* the data to be played next, in order to provide a seamless transition between resources. This type of prefetching or buffering is particularly useful for protocols such as HTTP, where the data is usually buffered before playback. It is recommended that the AVTransport service checks the MIME-type of the specified resource when executing this action. For AVTransport instances that control the transport of DRM-controlled content, the authentication process is also recommended to start as a result of executing this action. The *SetNextAVTransportURI()* action is successful even when a necessary authentication or revocation check cannot be completed before the expiration of time allotted for the completion of the *SetNextAVTransportURI()* action. In the case of AVTransport instances that control the transport of DRM-controlled content, the subsequent detection of conditions that need to be communicated to the control point, like an authentication failure, a revocation condition, etc. are indicated via the *DRMState* state variable.

A control point can supply metadata, using a *DIDL-Lite XML Fragment* (defined in the ContentDirectory service specification), via argument [NextURIMetaData](#). If supported by the AVTransport service, this metadata is stored in a state variable, and returned as an output argument as part of actions [GetMediaInfo\(\)](#) and [GetMediaInfo_Ext\(\)](#). If a control point does not want to use this feature it can supply the empty string for the [NextURIMetaData](#) argument.

A resource may have descriptions of the embedded media components associated with it, as indicated by the metadata property [upnp:resExt::componentInfo](#) (see the ContentDirectory service specification). This description can be used by the device implementation to provide an extended user experience (for example, by offering a choice between different subtitle languages). Some associated components can have their own resource (e.g. URI), these resources are known as secondary resources. A control point can supply the metadata including the secondary resources in the [NextURIMetadata](#) argument. The AVTransport service implementation can use this information to offer a selection of alternative playback components to a control point.

If a playback component is a secondary resource, the device shall support synchronized playback of these resources upon invocation of the [Play\(\)](#) action. Similarly, actions such as [Stop\(\)](#) and [Pause\(\)](#) shall affect all the secondary resources as well. If the AVTransport service implementation does not support this feature, or if the control point does not supply the metadata, then only the URI of the primary resource will be played back. As all secondary resources are synchronized with the primary resource during playback, implementations can implement retrieval of the state information of an AVTransport instance (via the actions [GetMediaInfo\(\)](#), [GetMediaInfo_Ext\(\)](#), [GetTransportInfo\(\)](#) and [GetPositionInfo\(\)](#)) based on the state of the primary resource.

Note: The time resolution of what is considered synchronized playback is dependent on the application.

5.4.3.1 Arguments

Table 27 — Arguments for [SetNextAVTransportURI\(\)](#)

| Argument | Direction | relatedStateVariable |
|---------------------------------|--------------------|--|
| InstanceID | IN | A_ARG_TYPE_InstanceID |
| NextURI | IN | NextAVTransportURI |
| NextURIMetaData | IN | NextAVTransportURIMetaData |

5.4.3.2 Dependency on State

None.

5.4.3.3 Effect on State

This action does **not** change the transport state of the specified instance. In case that the next URI buffer exists (that is: a legal URI which will be rendered next has been located), when the playback of the current resource finishes, state variable [AVTransportURI](#) changes to the value of state variable [NextAVTransportURI](#). The same holds for [AVTransportURIMetaData](#) and [NextAVTransportURIMetaData](#). The process repeats itself until there is no more URI to be rendered. In such case, the state variable [NextAVTransportURI](#) will be set to the empty string.

If an illegal URI is used for the [SetNextAVTransportURI\(\)](#) action, which is detected immediately and most likely while the current URI is still being rendered, the current transport state shall be kept. After the current URI finishes playing, the transition to that illegal URI cannot be made and [TransportState](#) shall be set to "[STOPPED](#)".

5.4.3.4 Errors

Table 28 — Error Codes for SetNextAVTransportURI()

| ErrorCode | errorDescription | Description |
|-----------|-------------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 714 | Illegal MIME-type | The specified resource has a MIME-type which is not supported by the AVTransport service. |
| 715 | Content 'BUSY' | This indicates that the resource is already in use at this time. |
| 716 | Resource not found | The specified resource cannot be found in the network. |
| 718 | Invalid InstanceID | The specified InstanceID is invalid for this AVTransport. |
| 719 | DRM error | The action failed because an unspecified DRM error occurred. |
| 720 | Expired content | The action failed because the content use validity interval has expired. |
| 721 | Non-allowed use | The action failed because the requested content use is disallowed. |
| 722 | Can't determine allowed uses | The action failed because the allowed content uses cannot be verified. |
| 723 | Exhausted allowed use | The action failed because the number of times this content has been used as requested has reached the maximum allowed number of uses. |
| 724 | Device authentication failure | The action failed because of a device authentication failure between the media source device and the media sink device. |
| 725 | Device revocation | The action failed because either the media source device or the media sink device has been revoked. |
| 737 | No DNS Server | The DNS Server is not available (HTTP error 503). |
| 738 | Bad Domain Name | Unable to resolve the Fully Qualified Domain Name (HTTP error 502). |
| 739 | Server Error | The server that hosts the resource is unreachable or unresponsive (HTTP error 404/410). |

5.4.4 GetMediaInfo()

This required action returns information associated with the current media of the specified instance; it has no effect on state.

5.4.4.1 Arguments

Table 29 — Arguments for GetMediaInfo()

| Argument | Direction | relatedStateVariable |
|---------------------------|------------|-----------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE InstanceID</u> |
| <u>NrTracks</u> | <u>OUT</u> | <u>NumberOfTracks</u> |
| <u>MediaDuration</u> | <u>OUT</u> | <u>CurrentMediaDuration</u> |
| <u>CurrentURI</u> | <u>OUT</u> | <u>AVTransportURI</u> |
| <u>CurrentURIMetaData</u> | <u>OUT</u> | <u>AVTransportURIMetaData</u> |
| <u>NextURI</u> | <u>OUT</u> | <u>NextAVTransportURI</u> |
| <u>NextURIMetaData</u> | <u>OUT</u> | <u>NextAVTransportURIMetaData</u> |
| <u>PlayMedium</u> | <u>OUT</u> | <u>PlaybackStorageMedium</u> |
| <u>RecordMedium</u> | <u>OUT</u> | <u>RecordStorageMedium</u> |
| <u>WriteStatus</u> | <u>OUT</u> | <u>RecordMediumWriteStatus</u> |

5.4.4.2 Dependency on State

None.

5.4.4.3 Effect on State

None.

5.4.4.4 Errors

Table 30 — Error Codes for GetMediaInfo()

| ErrorCode | errorDescription | Description |
|-----------|---------------------------|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |

5.4.5 GetMediaInfo_Ext()

This required action returns information associated with the current media of the specified instance; it has no effect on state. The information returned is identical to the information returned by the GetMediaInfo() action, except for the additionally returned CurrentType argument

5.4.5.1 Arguments

Table 31 — Arguments for GetMediaInfo_Ext()

| Argument | Direction | relatedStateVariable |
|---------------------------|------------|-----------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE InstanceID</u> |
| <u>CurrentType</u> | <u>OUT</u> | <u>CurrentMediaCategory</u> |
| <u>NrTracks</u> | <u>OUT</u> | <u>NumberOfTracks</u> |
| <u>MediaDuration</u> | <u>OUT</u> | <u>CurrentMediaDuration</u> |
| <u>CurrentURI</u> | <u>OUT</u> | <u>AVTransportURI</u> |
| <u>CurrentURIMetaData</u> | <u>OUT</u> | <u>AVTransportURIMetaData</u> |
| <u>NextURI</u> | <u>OUT</u> | <u>NextAVTransportURI</u> |
| <u>NextURIMetaData</u> | <u>OUT</u> | <u>NextAVTransportURIMetaData</u> |
| <u>PlayMedium</u> | <u>OUT</u> | <u>PlaybackStorageMedium</u> |
| <u>RecordMedium</u> | <u>OUT</u> | <u>RecordStorageMedium</u> |
| <u>WriteStatus</u> | <u>OUT</u> | <u>RecordMediumWriteStatus</u> |

5.4.5.2 Dependency on State

None.

5.4.5.3 Effect on State

None.

5.4.5.4 Errors

Table 32 — Error Codes for GetMediaInfo_Ext()

| ErrorCode | errorDescription | Description |
|-----------|---------------------------|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |

5.4.6 GetTransportInfo()

This required action returns information associated with the current transport state of the specified instance; it has no effect on state.

5.4.6.1 Arguments

Table 33 — Arguments for GetTransportInfo()

| Argument | Direction | relatedStateVariable |
|-------------------------------|------------|------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE InstanceID</u> |
| <u>CurrentTransportState</u> | <u>OUT</u> | <u>TransportState</u> |
| <u>CurrentTransportStatus</u> | <u>OUT</u> | <u>TransportStatus</u> |
| <u>CurrentSpeed</u> | <u>OUT</u> | <u>TransportPlaySpeed</u> |

5.4.6.2 Dependency on State

None.

5.4.6.3 Effect on State

None.

5.4.6.4 Errors

Table 34 — Error Codes for GetTransportInfo()

| ErrorCode | errorDescription | Description |
|-----------|---------------------------|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |

5.4.7 GetPositionInfo()

This required action returns information associated with the current position of the transport of the specified instance; it has no effect on state.

5.4.7.1 Arguments

Table 35 — Arguments for GetPositionInfo()

| Argument | Direction | relatedStateVariable |
|----------------------|------------|--------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE InstanceID</u> |
| <u>Track</u> | <u>OUT</u> | <u>CurrentTrack</u> |
| <u>TrackDuration</u> | <u>OUT</u> | <u>CurrentTrackDuration</u> |
| <u>TrackMetaData</u> | <u>OUT</u> | <u>CurrentTrackMetaData</u> |
| <u>TrackURI</u> | <u>OUT</u> | <u>CurrentTrackURI</u> |
| <u>RelTime</u> | <u>OUT</u> | <u>RelativeTimePosition</u> |
| <u>AbsTime</u> | <u>OUT</u> | <u>AbsoluteTimePosition</u> |
| <u>RelCount</u> | <u>OUT</u> | <u>RelativeCounterPosition</u> |
| <u>AbsCount</u> | <u>OUT</u> | <u>AbsoluteCounterPosition</u> |

5.4.7.2 Dependency on State

None.

5.4.7.3 Effect on State

None.

5.4.7.4 Errors

Table 36 — Error Codes for ***GetPositionInfo()***

| ErrorCode | errorDescription | Description |
|-----------|---------------------------|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |

5.4.8 ***GetDeviceCapabilities()***

This required action returns information on device capabilities of the specified instance, such as the supported playback and recording formats, and the supported quality levels for recording. This action has no effect on state.

5.4.8.1 Arguments

Table 37 — Arguments for ***GetDeviceCapabilities()***

| Argument | Direction | relatedStateVariable |
|------------------------|------------|-------------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE_InstanceID</u> |
| <u>PlayMedia</u> | <u>OUT</u> | <u>PossiblePlaybackStorageMedia</u> |
| <u>RecMedia</u> | <u>OUT</u> | <u>PossibleRecordStorageMedia</u> |
| <u>RecQualityModes</u> | <u>OUT</u> | <u>PossibleRecordQualityModes</u> |

5.4.8.2 Dependency on State

None.

5.4.8.3 Effect on State

None.

5.4.8.4 Errors

Table 38 — Error Codes for ***GetDeviceCapabilities()***

| ErrorCode | errorDescription | Description |
|-----------|---------------------------|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |

5.4.9 ***GetTransportSettings()***

This required action returns information on various settings of the specified instance, such as the current play mode and the current recording quality mode. This action has no effect on state.

5.4.9.1 Arguments

Table 39 — Arguments for ***GetTransportSettings()***

| Argument | Direction | relatedStateVariable |
|-----------------------|------------|---------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE_InstanceID</u> |
| <u>PlayMode</u> | <u>OUT</u> | <u>CurrentPlayMode</u> |
| <u>RecQualityMode</u> | <u>OUT</u> | <u>CurrentRecordQualityMode</u> |

5.4.9.2 Dependency on State

None.

5.4.9.3 Effect on State

None.

5.4.9.4 Errors

Table 40 — Error Codes for GetTransportSettings()

| ErrorCode | errorDescription | Description |
|-----------|---------------------------|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |

5.4.10 Stop()

This required action stops the progression of the current resource that is associated with the specified instance. Additionally, it is recommended that the *output of the device* (defined below) should change to something other than the current snippet of resource. Although the exact nature of this change varies from device to device, a common behavior is to immediately cease all output from the device. Nevertheless, the exact behavior is defined by the manufacturer of the device.

On some devices, the current position on the transport changes as a result of the Stop() action. This can be detected by control points via event notification of state variable CurrentTrack. Alternatively, a control point can poll using the GetPositionInfo() action.

Output of a device: In this context, the term *output of the device* (used above) has different semantics depending on the type of device that has implemented this AVTransport service. Some devices (for example, MediaServer devices) output media content to the network while other devices (for example, a MediaRenderer) output a visual and/or audio representation of media content that was received from the network.

5.4.10.1 Arguments

Table 41 — Arguments for Stop()

| Argument | Direction | relatedStateVariable |
|-------------------|-----------|------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE InstanceID</u> |

5.4.10.2 Dependency on State

This action is allowed in all transport states except in state "NO_MEDIA_PRESENT".

5.4.10.3 Effect on State

This action changes TransportState to "STOPPED". If it would take a noticeable amount of time before a human user would actually see or hear the media playback has stopped, the AVTransport may temporarily go to the "TRANSITIONING" state before going to "STOPPED".

5.4.10.4 Errors

Table 42 — Error Codes for [Stop\(\)](#)

| ErrorCode | errorDescription | Description |
|-----------|------------------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (the transport is hold locked), the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 718 | Invalid InstanceID | The specified InstanceID is invalid for this AVTransport. |

5.4.11 [Play\(\)](#)

This required action starts playing the resource of the specified instance, at the specified speed, starting at the current position, according to the current play mode. Playing shall continue until the resource ends or the transport state is changed via actions [Stop\(\)](#) or [Pause\(\)](#). The device shall do a *best effort* to match the specified play speed. Actually supported speeds can be retrieved from the allowed value list of the [TransportPlaySpeed](#) state variable in the AVTransport service description.

If no [AVTransportURI](#) is set, the resource being played is device-dependent.

5.4.11.1 Arguments

Table 43 — Arguments for [Play\(\)](#)

| Argument | Direction | relatedStateVariable |
|----------------------------|-----------|------------------------------|
| InstanceID | <i>IN</i> | <i>A_ARG_TYPE InstanceID</i> |
| Speed | <i>IN</i> | <i>TransportPlaySpeed</i> |

5.4.11.2 Dependency on State

This action is allowed in the “[STOPPED](#)”, “[PLAYING](#)”, and “[PAUSED PLAYBACK](#)” transport states. In other states the action may also succeed or it may fail with error code 701.

5.4.11.3 Effect on State

This action changes [TransportState](#) to “[PLAYING](#)” and [TransportPlaySpeed](#) to the value specified in the [Speed](#) argument of the [Play\(\)](#) action. If it would take a noticeable amount of time before a human user would actually see or hear the media playing, the AVTransport may temporarily go to the “[TRANSITIONING](#)” state before going to “[PLAYING](#)”. This might be appropriate, for example, for devices that need to start buffering or completely download the media before playback can start.

5.4.11.4 Errors

Table 44 — Error Codes for *Play()*

| errorCode | errorDescription | Description |
|-----------|-----------------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 702 | No contents | The media does not contain any contents that can be played. |
| 703 | Read error | The media cannot be read (for example, because of dust or a scratch). |
| 704 | Format not supported for playback | The storage format of the currently loaded media is not supported for playback by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (the transport is hold locked), the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 714 | Illegal MIME-type | The resource to be played has a MIME-type which is not supported by the AVTransport service. |
| 715 | Content 'BUSY' | This indicates that the resource is already in use at this time. |
| 716 | Resource not found | The resource to be played cannot be found in the network. |
| 717 | Play speed not supported | The specified playback speed is not supported by the AVTransport service. |
| 718 | Invalid InstanceID | The specified <i>InstanceID</i> is invalid for this AVTransport. |
| 719 | DRM error | The action failed because an unspecified DRM error occurred. |
| 720 | Expired content | The action failed because the content use validity interval has expired. |
| 721 | Non-allowed use | The action failed because the requested content use is disallowed. |
| 722 | Can't determine allowed uses | The action failed because the allowed content uses cannot be verified. |
| 723 | Exhausted allowed use | The action failed because the number of times this content has been used as requested has reached the maximum allowed number of uses. |
| 724 | Device authentication failure | The action failed because of a device authentication failure between the media source device and the media sink device. |
| 725 | Device revocation | The action failed because either the media source device or the media sink device has been revoked. |

5.4.12 *Pause()*

This is an allowed action. While the device is in a playing state, that is: *TransportState* is "*PLAYING*", this action halts the progression of the resource that is associated with the specified *InstanceID*. Any visual representation of the resource should remain displayed in a static manner (for example, the last frame of video remains displayed). Any audio representation of the resource should be muted. The difference between *Pause()* and *Stop()* is that *Pause()* shall remain at the current position within the resource and the current resource shall persist as described above (for example, the current video resource continues to be transmitted/displayed).

When the device is recording, that is: the *TransportState* is "*RECORDING*", the device shall maintain its current recording position, but will not accept any more data to record. Any data received after the *Pause()* action and before the next *Record()* action will be lost.

5.4.12.1 Arguments

Table 45 — Arguments for *Pause()*

| Argument | Direction | relatedStateVariable |
|-------------------|-----------|------------------------------|
| <i>InstanceID</i> | <i>IN</i> | <i>A_ARG_TYPE_InstanceID</i> |

5.4.12.2 Dependency on State

This action is always allowed while playing or recording. In other cases, the action may fail with error code 701.

5.4.12.3 Effect on State

When recording, this action changes *TransportState* to “*PAUSED_RECORDING*”. When playing, this action changes *TransportState* to “*PAUSED_PLAYBACK*”. The *Pause()* action does not operate as a toggle.

If it would take a noticeable amount of time before a human user would actually see or hear the media is paused, the AVTransport may temporarily go to the “*TRANSITIONING*” state before going to “*PAUSED_PLAYBACK*”.

Similarly, if it would take a noticeable amount of time before recording is paused, the AVTransport may temporarily go to the “*TRANSITIONING*” state before going to “*PAUSED_RECORDING*”.

5.4.12.4 Errors

Table 46 — Error Codes for *Pause()*

| ErrorCode | errorDescription | Description |
|-----------|---------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (the transport is hold locked), the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 718 | Invalid <i>InstanceID</i> | The specified <i>InstanceID</i> is invalid for this AVTransport. |

5.4.13 *Record()*

This allowed action starts recording on the specified transport instance, at the current position on the media, according to the currently specified recording quality, and returns immediately. If *AVTransportURI* is set (differs from the empty string) then that resource will be recorded. If no *AVTransportURI* is set (equals the empty string), then the source of the content being recorded is device-dependent. In both cases, whether the device outputs the resource to a screen or speakers while recording is device-dependent. If the device implementing the *Record()* action also has a ContentDirectory service, then recorded content will be added to this ContentDirectory in a device-dependent way. Specifically, there is no UPnP mechanism to specify the location of the recorded content in the ContentDirectory hierarchy.

5.4.13.1 Arguments

Table 47 — Arguments for *Record()*

| Argument | Direction | relatedStateVariable |
|-------------------|-----------|------------------------------|
| <i>InstanceID</i> | <i>IN</i> | <i>A_ARG_TYPE_InstanceID</i> |

5.4.13.2 Dependency on State

This action is allowed in the “*STOPPED*” or “*PAUSED_RECORDING*” transport states. In other states the action may fail with error code 701.

5.4.13.3 Effect on State

This action changes *TransportState* to “*RECORDING*”. If it would take a noticeable amount of time before recording starts, the AVTransport may temporarily go to the “*TRANSITIONING*” state before going to “*RECORDING*”.

5.4.13.4 Errors

Table 48 — Error Codes for *Record()*

| errorCode | errorDescription | Description |
|-----------|------------------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (the transport is hold locked), the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 706 | Write error | The media cannot be written. (for example, because of dust or a scratch) |
| 707 | Media is protected or not writable | The media is write-protected or is of a not writable type. |
| 708 | Format not supported for recording | The storage format of the currently loaded media is not supported for recording by this device. |
| 709 | Media is full | There is no free space left on the loaded media. |
| 718 | Invalid <i>InstanceID</i> | The specified <i>InstanceID</i> is invalid for this AVTransport. |

5.4.14 *Seek()*

This required action starts seeking through the resource controlled by the specified instance - as fast as possible - to the position, specified in the *Target* argument. The value in the *Unit* argument indicates how the *Target* argument needs to be interpreted.

Unit value “*TRACK_NR*” indicates seeking to the beginning of a particular track number. For track-unaware media (such as VCRs), *Seek(InstanceID, “TRACK_NR”, “1”)* is equivalent to the common *FastReverse* VCR functionality. Special track number “0” is used to indicate the end of the media. Hence, *Seek(InstanceID, “TRACK_NR”, “0”)* is equivalent to the common *FastForward* VCR functionality.

For *Unit* values “*ABS_TIME*”, “*REL_TIME*”, “*ABS_COUNT*”, and “*REL_COUNT*”, the semantics defined by the corresponding state variables shall be respected. After the *Seek()* action completes, the appropriate state variable shall contain the value, specified in the *Target* argument. For example, if the *RelativeTimePosition* state variable contains the value “00:05:30” before the *Seek()* action, then *Seek(InstanceID, “REL_TIME”, “00:00:10”)* will move the current position to 10 seconds from the beginning of the track and the *RelativeTimePosition* state variable will contain the value “00:00:10” after the *Seek()* action is completed.

For *Unit* value “*REL_FRAME*”, the semantics of the *Target* argument is defined as follows:

- For track-aware media, the *Target* argument contains the desired position in the current track, in terms of frames, measured from the beginning of the current track. The range for the *Target* argument is from “0” to the duration of the current track,

measured in number of frames. For track-aware media, the *Target* argument shall always contain a positive value.

- For track-unaware media (e.g. a single tape), the *Target* argument contains the desired position, in terms of frames, measured from a *zero reference point* on the media. The range for the *Target* argument is from the beginning of the media, measured from the zero reference point to the end of the media, also measured from the zero reference point. For track-unaware media, the *Target* argument can be negative. Indeed, when the zero reference point does not coincide with the beginning of the media, all positions before the zero reference point are expressed as negative values.

For *Unit* value “*FRAME*”, the *Target* argument contains the desired position, in terms of frames, measured from the beginning of the media. The range for the *Target* argument is from “0” to the total duration of the current media, expressed in frames. The *Target* argument shall always contain a positive value.

The *Unit* values “*TAPE-INDEX*” and “*REL TAPE-INDEX*” only apply for track-unaware media. It is assumed that the media contains a set of subsequent ‘marks’ that indicate some relevant position on the media (a scene change in a video, for instance). The position of these marks and how these marks are inserted on the media is completely device dependent. However, it is further assumed that these marks are sequentially numbered from one to the total number of marks on the media. Furthermore, the first mark is always assumed to be present at the beginning of the media and the last mark is always assumed to be present at the end of the media.

For *Unit* value “*REL TAPE-INDEX*”, the *Target* argument contains the desired position, in terms of tape index marks, measured from the current position on the media. The range for the *Target* argument is the *i4* data type range. If a value is specified that is outside the range of available tape index marks, then the resulting position will be either the first tape mark position (*Target* < 0) i.e. the beginning of the media, or the last tape mark position (*Target* > 0) i.e. the end of the media.

For *Unit* value “*TAPE-INDEX*”, the *Target* argument contains the desired position, in terms of tape index marks, measured from the beginning of the media. The range for the *Target* argument is from “1” (the first mark on the tape, at the beginning of the tape) to the total number of tape marks on the media. The *Target* argument shall always contain a positive value.

5.4.14.1 Arguments

Table 49 — Arguments for *Seek()*

| Argument | Direction | relatedStateVariable |
|-------------------|-----------|------------------------------|
| <i>InstanceID</i> | <i>IN</i> | <i>A_ARG_TYPE_InstanceID</i> |
| <i>Unit</i> | <i>IN</i> | <i>A_ARG_TYPE_SeekMode</i> |
| <i>Target</i> | <i>IN</i> | <i>A_ARG_TYPE_SeekTarget</i> |

5.4.14.2 Dependency on State

This action is allowed in the “*STOPPED*” and “*PLAYING*” transport states, in other states the action may fail with error code 701.

5.4.14.3 Effect on State

This action changes *TransportState* to “*TRANSITIONING*” and then returns immediately. When the desired position is reached, *TransportState* will return to the previous transport state (typically “*STOPPED*” or “*PLAYING*”). Note that the new transport state can be detected through the eventing mechanism.

5.4.14.4 Errors

Table 50 — Error Codes for Seek()

| errorCode | errorDescription | Description |
|-----------|-------------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (the transport is hold locked), the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 710 | Seek mode not supported | The specified seek mode is not supported by the device. |
| 711 | Illegal seek target | The specified seek target is not present on the media or is not specified in terms of the seek mode. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |
| 719 | DRM error | The action failed because an unspecified DRM error occurred. |
| 720 | Expired content | The action failed because the content use validity interval has expired. |
| 721 | Non-allowed use | The action failed because the requested content use is disallowed. |
| 722 | Can't determine allowed uses | The action failed because the allowed content uses cannot be verified. |
| 723 | Exhausted allowed use | The action failed because the number of times this content has been used as requested has reached the maximum allowed number of uses. |
| 724 | Device authentication failure | The action failed because of a device authentication failure between the media source device and the media sink device. |
| 725 | Device revocation | The action failed because either the media source device or the media sink device has been revoked. |

5.4.15 Next()

This required action is used to advance to the next track. This action is functionally equivalent to `Seek("TRACK_NR", "CurrentTrackNr+1")`. This action does not *cycle back* to the first track.

5.4.15.1 Arguments

Table 51 — Arguments for Next()

| Argument | Direction | relatedStateVariable |
|-------------------|-----------|------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE InstanceID</u> |

5.4.15.2 Dependency on State

This action is allowed in the "STOPPED" and "PLAYING" transport states, in other states the action may succeed or it may fail with error code 701.

5.4.15.3 Effect on State

This action changes TransportState to "TRANSITIONING" and then returns immediately. When the desired position is reached, TransportState will return to the previous transport state (typically "STOPPED"). Note that this can be detected through the eventing mechanism.

5.4.15.4 Errors

Table 52 — Error Codes for **Next()**

| errorCode | errorDescription | Description |
|-----------|-------------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (he transport is hold locked) the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 711 | Illegal seek target | The specified seek target is not present on the media. |
| 718 | Invalid InstanceID | The specified <i>InstanceID</i> is invalid for this AVTransport. |
| 719 | DRM error | The action failed because an unspecified DRM error occurred. |
| 720 | Expired content | The action failed because the content use validity interval has expired. |
| 721 | Non-allowed use | The action failed because the requested content use is disallowed. |
| 722 | Can't determine allowed uses | The action failed because the allowed content uses cannot be verified. |
| 723 | Exhausted allowed use | The action failed because the number of times this content has been used as requested has reached the maximum allowed number of uses. |
| 724 | Device authentication failure | The action failed because of a device authentication failure between the media source device and the media sink device. |
| 725 | Device revocation | The action failed because either the media source device or the media sink device has been revoked. |

5.4.16 **Previous()**

This required action is used to advance to the previous track. This action is functionally equivalent to `Seek("TRACK_NR", "CurrentTrackNr-1")`. This action does not *cycle back* to the last track.

5.4.16.1 Arguments

Table 53 — Arguments for **Previous()**

| Argument | Direction | relatedStateVariable |
|-------------------|-----------|------------------------------|
| <i>InstanceID</i> | <i>IN</i> | <i>A_ARG_TYPE_InstanceID</i> |

5.4.16.2 Dependency on State

This action is allowed in the "**STOPPED**" and "**PLAYING**" transport states, in other states the action may succeed or it may fail with error code 701.

5.4.16.3 Effect on State

This action changes *TransportState* to "**TRANSITIONING**" and then returns immediately. When the desired position is reached, *TransportState* will return to the previous transport state (typically "**STOPPED**"). Note that this can be detected through the eventing mechanism.

5.4.16.4 Errors

Table 54 — Error Codes for *Previous()*

| errorCode | errorDescription | Description |
|-----------|-------------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (he transport is hold locked) the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 711 | Illegal seek target | The specified seek target is not present on the media. |
| 718 | Invalid <i>InstanceID</i> | The specified <i>InstanceID</i> is invalid for this AVTransport. |
| 719 | DRM error | The action failed because an unspecified DRM error occurred. |
| 720 | Expired content | The action failed because the content use validity interval has expired. |
| 721 | Non-allowed use | The action failed because the requested content use is disallowed. |
| 722 | Can't determine allowed uses | The action failed because the allowed content uses cannot be verified. |
| 723 | Exhausted allowed use | The action failed because the number of times this content has been used as requested has reached the maximum allowed number of uses. |
| 724 | Device authentication failure | The action failed because of a device authentication failure between the media source device and the media sink device. |
| 725 | Device revocation | The action failed because either the media source device or the media sink device has been revoked. |

5.4.17 *SetPlayMode()*

This allowed action sets the play mode of the specified AVTransport instance.

5.4.17.1 Arguments

Table 55 — Arguments for *SetPlayMode()*

| Argument | Direction | relatedStateVariable |
|--------------------|-----------|------------------------------|
| <i>InstanceID</i> | <i>IN</i> | <i>A_ARG_TYPE_InstanceID</i> |
| <i>NewPlayMode</i> | <i>IN</i> | <i>CurrentPlayMode</i> |

5.4.17.2 Dependency on State

None.

5.4.17.3 Effect on State

This action sets the play mode of the specified instance to the specified value. A subsequent *Play()* action for this instance will behave according to the set play mode.

5.4.17.4 Errors

Table 56 — Error Codes for **SetPlayMode()**

| ErrorCode | errorDescription | Description |
|-----------|-------------------------|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 712 | Play mode not supported | The specified play mode is not supported by the device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (the transport is hold locked) the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 718 | Invalid InstanceID | The specified <i>InstanceID</i> is invalid for this AVTransport. |

5.4.18 **SetRecordQualityMode()**

This conditionally allowed action may be implemented if the AVTransport service implements the **Record()** action. Otherwise, implementing the action is not allowed. This action sets the record quality mode of the specified AVTransport instance.

5.4.18.1 Arguments

Table 57 — Arguments for **SetRecordQualityMode()**

| Argument | Direction | relatedStateVariable |
|-----------------------------|-----------|---------------------------------|
| <i>InstanceID</i> | <i>IN</i> | <i>A_ARG_TYPE InstanceID</i> |
| <i>NewRecordQualityMode</i> | <i>IN</i> | <i>CurrentRecordQualityMode</i> |

5.4.18.2 Dependency on State

None.

5.4.18.3 Effect on State

This action sets *CurrentRecordQualityMode* of the specified instance to the specified record quality mode. A subsequent **Record()** action will behave according to the specified record quality mode. This action does not change any ongoing recordings.

5.4.18.4 Errors

Table 58 — Error Codes for **SetRecordQualityMode()**

| ErrorCode | errorDescription | Description |
|-----------|------------------------------|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 713 | Record quality not supported | The specified record quality is not supported by the device. |
| 718 | Invalid InstanceID | The specified <i>InstanceID</i> is invalid for this AVTransport. |

5.4.19 **GetCurrentTransportActions()**

This allowed action returns the *CurrentTransportActions* state variable for the specified instance.

5.4.19.1 Arguments

Table 59 — Arguments for [GetCurrentTransportActions\(\)](#)

| Argument | Direction | relatedStateVariable |
|-----------------------------------|----------------------------|--|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE_InstanceID</u> |
| <u>Actions</u> | <u>OUT</u> | <u>CurrentTransportActions</u> |

5.4.19.2 Dependency on State

None.

5.4.19.3 Effect on State

None.

5.4.19.4 Errors

Table 60 — Error Codes for [GetCurrentTransportActions\(\)](#)

| errorCode | errorDescription | Description |
|-----------|---|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |

5.4.20 [GetDRMState\(\)](#)

This conditionally required action shall be implemented if the AVTransport service implements the [DRMState](#) state variable, otherwise it is not allowed. This action returns information associated with the current DRM state of the specified instance. It has no effect on state.

5.4.20.1 Arguments

Table 61 — Arguments for [GetDRMState\(\)](#)

| Argument | Direction | relatedStateVariable |
|--|----------------------------|--|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE_InstanceID</u> |
| <u>CurrentDRMState</u> | <u>OUT</u> | <u>DRMState</u> |

5.4.20.2 Dependency on State

None.

5.4.20.3 Effect on State

None.

5.4.20.4 Errors

Table 62 — Error Codes for [GetDRMState\(\)](#)

| errorCode | errorDescription | Description |
|-----------|---|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |

5.4.21 [GetStateVariables\(\)](#)

This conditionally required action shall be supported if the AVTransport service supports bookmarks (see *BOOKMARK feature, as defined by the ContentDirectory service specification*), otherwise it is not allowed. Note that the service shall always implement

[GetStateVariables\(\)](#) and [SetStateVariables\(\)](#) as a pair; either the service includes or omits both actions in the same implementation. This action returns the current collection of AVTransport state variable names and their respective values that are associated with the AVTransport instance indicated by the input argument [InstanceID](#). The [StateVariableList](#) argument specifies which state variables are captured. Vendor-extended state variables can be specified in this argument as well. If the value of the [StateVariableList](#) argument is set to “*”, the action shall return all the supported state variables of the service, including the vendor-extended state variables except for [LastChange](#) and any [A_ARG_TYPE_xxx](#) variables. When the action fails and the error code indicates “invalid StateVariableList”, the control point is encouraged to inspect the list or invoke successive [Getxxx\(\)](#) actions for each of the state variables instead. AVTransport service implementations that want to participate in scenarios that use bookmarks shall implement this action. Furthermore, when creating or manipulating bookmarks, control points are encouraged to set the [StateVariableList](#) argument to “*” when invoking this action. This ensures that the maximum available set of state information is stored within the bookmark item.

5.4.21.1 Arguments

Table 63 — Arguments for [GetStateVariables\(\)](#)

| Argument | Direction | relatedStateVariable |
|---|---------------------|--|
| InstanceID | IN | A_ARG_TYPE_InstanceID |
| StateVariableList | IN | A_ARG_TYPE_StateVariableList |
| StateVariableValuePairs | OUT | A_ARG_TYPE_StateVariableValuePairs |

5.4.21.2 Dependency on State

None.

5.4.21.3 Effect on State

None.

5.4.21.4 Errors

Table 64 — Error Codes for [GetStateVariables\(\)](#)

| errorCode | errorDescription | Description |
|-----------|---|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid InstanceID | The specified InstanceID is invalid for this AVTransport. |
| 726 | Invalid StateVariableList | Some of the variables are invalid. |
| 727 | Ill-formed CSV List | The CSV list is not well formed. |

5.4.22 [SetStateVariables\(\)](#)

This conditionally required action shall be supported if the AVTransport service supports bookmarks (see *BOOKMARK feature, as defined by the ContentDirectory service specification*), otherwise it is not allowed. Note that the service shall always implement [GetStateVariables\(\)](#) and [SetStateVariables\(\)](#) as a pair; either the service includes or omits both actions in the same implementation. This action extracts the values from the [StateVariableValuePairs](#) IN argument and copies these values to the corresponding AVTransport state variables associated with the AVTransport instance indicated by the input argument [InstanceID](#). The [AVTransportUDN](#), [ServiceType](#) and [ServiceId](#) argument values are used for compatibility checking by the device. If this action is invoked to replace all of the state variable values, the device shall check whether the [AVTransportUDN](#), [ServiceType](#) and [ServiceId](#) input arguments match those of the device. If this is the case, all state variable values will be replaced. Otherwise, the device only sets the state variable values that are relevant. The [StateVariableList](#) argument is a CSV list of state variable names that were

accepted by the AVTransport service. AVTransport service implementations that want to participate in scenarios that use bookmarks shall implement this action.

5.4.22.1 Arguments

Table 65 — Arguments for SetStateVariables()

| Argument | Direction | relatedStateVariable |
|--------------------------------|------------|---|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE InstanceID</u> |
| <u>AVTransportUDN</u> | <u>IN</u> | <u>A_ARG_TYPE DeviceUDN</u> |
| <u>ServiceType</u> | <u>IN</u> | <u>A_ARG_TYPE ServiceType</u> |
| <u>ServiceId</u> | <u>IN</u> | <u>A_ARG_TYPE ServiceID</u> |
| <u>StateVariableValuePairs</u> | <u>IN</u> | <u>A_ARG_TYPE StateVariableValuePairs</u> |
| <u>StateVariableList</u> | <u>OUT</u> | <u>A_ARG_TYPE StateVariableList</u> |

5.4.22.2 Dependency on State

None.

5.4.22.3 Effect on State

None.

5.4.22.4 Errors

Table 66 — Error Codes for SetStateVariables()

| errorCode | errorDescription | Description |
|-----------|-------------------------------------|--|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |
| 728 | Invalid <u>State Variable Value</u> | One of the StateVariableValuePairs contains an invalid value. |
| 729 | Invalid Service Type | The specified <u>ServiceType</u> is invalid. |
| 730 | Invalid Service Id | The specified <u>ServiceId</u> is invalid. |

5.4.23 GetSyncOffset()

This conditionally required action shall be implemented if the AVTransport service supports the ConnectionManager *CLOCKSYNC* feature, otherwise it is not allowed. If this action is implemented, then the AVTransport service shall implement all of these actions: GetSyncOffset(), SetSyncOffset(), and AdjustSyncOffset(). This action is used to retrieve the current value of the SyncOffset state variable. See subclass 5.2.31 for details.

5.4.23.1 Arguments

Table 67 — Arguments for GetSyncOffset()

| Argument | Direction | relatedStateVariable |
|--------------------------|------------|------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE InstanceID</u> |
| <u>CurrentSyncOffset</u> | <u>OUT</u> | <u>SyncOffset</u> |

5.4.23.2 Dependency on State

None.

5.4.23.3 Effect on State

None.

5.4.23.4 Errors

Table 68 — Error Codes for [GetSyncOffset\(\)](#)

| errorCode | errorDescription | Description |
|-----------|------------------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid InstanceID | The specified InstanceID is invalid for this AVTransport. |

5.4.24 [SetSyncOffset\(\)](#)

This conditionally required action shall be implemented if the AVTransport service supports the ConnectionManager *CLOCKSYNC feature*, otherwise it is not allowed. If this action is implemented, then the AVTransport service shall implement all of these actions: [GetSyncOffset\(\)](#), [SetSyncOffset\(\)](#), and [AdjustSyncOffset\(\)](#). This action is used to set the value of the [SyncOffset](#) state variable. See subclause 5.2.31 for details.

5.4.24.1 Arguments

Table 69 — Arguments for [SetSyncOffset\(\)](#)

| Argument | Direction | relatedStateVariable |
|-------------------------------|-----------|------------------------------|
| InstanceID | <i>IN</i> | <i>A_ARG_TYPE InstanceID</i> |
| NewSyncOffset | <i>IN</i> | SyncOffset |

5.4.24.2 Dependency on State

None.

5.4.24.3 Effect on State

As a result of this action the [SyncOffset](#) state variable is updated consistent with the value of the [NewSyncOffset](#) input argument.

5.4.24.4 Errors

Table 70 — Error Codes for [SetSyncOffset\(\)](#)

| errorCode | errorDescription | Description |
|-----------|---|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid InstanceID | The specified InstanceID is invalid for this AVTransport. |
| 731 | Invalid time, offset, or position value | The action failed because the supplied time, offset, or position value for an argument was not valid. |

5.4.25 [AdjustSyncOffset\(\)](#)

This conditionally required action shall be implemented if the AVTransport service supports the ConnectionManager *CLOCKSYNC feature*, otherwise it is not allowed. If this action is implemented, then the AVTransport service shall implement all of these actions: [GetSyncOffset\(\)](#), [SetSyncOffset\(\)](#), and [AdjustSyncOffset\(\)](#). This action is used to adjust the current value of the [SyncOffset](#) state variable. Successive invocations of this action have a cumulative effect. In other words, an invocation of this action changes the current value of the [SyncOffset](#) state variable by the specified amount. For example, assuming the current value of the [SyncOffset](#) state variable is P00:00:00.020, an invocation of this action with an adjustment value of P00:00:00.010 followed by another invocation of this action with an adjustment value of - P00:00:00.005 will result in a final [SyncOffset](#) value of P00:00:025.0. See subclause 5.2.31 for details.

If the [SyncOffset](#) state variable has been implemented, then the [GetSyncOffset\(\)](#), [SetSyncOffset\(\)](#), and [AdjustSyncOffset\(\)](#) actions shall also be implemented.

5.4.25.1 Arguments

Table 71 — Arguments for [AdjustSyncOffset\(\)](#)

| Argument | Direction | relatedStateVariable |
|----------------------------|-----------|---------------------------------|
| InstanceID | <i>IN</i> | <i>A_ARG_TYPE InstanceID</i> |
| Adjustment | <i>IN</i> | <i>A_ARG_TYPE SyncOffsetAdj</i> |

5.4.25.2 Dependency on State

None.

5.4.25.3 Effect on State

As a result of this action the [SyncOffset](#) state variable is updated consistent with the value of the [Adjustment](#) input argument.

5.4.25.4 Errors

Table 72 — Error Codes for [AdjustSyncOffset\(\)](#)

| errorCode | errorDescription | Description |
|-----------|---|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 718 | Invalid InstanceID | The specified InstanceID is invalid for this AVTransport. |
| 731 | Invalid time, offset, or position value | The action failed because the supplied time, offset, or position value for an argument was not valid. |

5.4.26 [SyncPlay\(\)](#)

This conditionally required action shall be implemented if the AVTransport service supports the ConnectionManager *CLOCKSYNC* feature, otherwise it is not allowed. This action behaves the same as the [Play\(\)](#) action except that the playback of the current content binary shall be synchronized with the device's internal time-of-day clock as specified by the [ReferencePosition](#) and [ReferencePresentationTime](#) input arguments. [ReferenceClockId](#) identifies the <deviceClockInfo> element of the device. The renderer shall start rendering the content at the [ReferencePresentationTime](#) argument at an offset inside the content specified by [ReferencePosition](#) argument. The [ReferencePositionUnits](#) argument identifies the format of the [ReferencePosition](#) argument, for example, time vs. frame count, etc. The [ReferencePosition](#) argument identifies a specific location within the content binary, for example, 1 hour, 10 minutes, and 34 seconds from the beginning. The [ReferencePresentationTime](#) argument identifies a precise time of day at which the [ReferencePosition](#) is to be played for example, June 6, 2009 at 21 hours, 14 minutes, and 36.152 seconds (past midnight). As described below, there are three distinct scenarios, depending on how the presentation time, specified by the [ReferencePresentationTime](#) argument, relates to the point in time that this action is invoked.

Upon receipt of [SyncPlay\(\)](#) action, the MediaRenderer compares the MIME type of the requested content item with its local listing of <supportedTimestamps> elements to determine the applicable <supportedTimestamps> element for that particular MIME type.

The combination of the protocol attribute and the format attribute shall select one and only one <supportedTimestamps> element for a given MIME type and transport protocol.

In the simple case (when the specified presentation time occurs in the near future), the device goes to the *TRANSITIONING* state, prepares itself to render the specified position within the content, then waits for the specified presentation time to occur before the content is actually presented. The content then progresses at the specified speed.

In a more complicated case, the specified ReferencePresentationTime argument refers to a time-of-day well into the future (for example hours, days, or even months). In this case, the device goes to the TRANSITIONING state, prepares itself to render the specified position within the content, then waits for the specified presentation time to occur before the content is actually presented. The content then progresses at the specified speed.

Lastly, it is possible that the ReferencePresentationTime identifies a point in time that has already occurred. Similar to the scenario above, the ReferencePosition and ReferencePresentationTime represent a “time-anchor” for the entire content. For a given time-anchor, each and every fragment of the content corresponds to an exact predetermined presentation time such that if the content were played (uninterrupted) at the specified speed, then the specified ReferencePosition would be presented exactly at the specified ReferencePresentationTime. In order to accommodate presentation times in the past, the device shall calculate the correct position within the content that corresponds to the time-of-day when the action is invoked. In other words, the device shall begin presenting the portion of the content that would have been presented to the end-user (at the time the action was invoked) if device had begun rendering the specified ReferencePosition at the specified ReferencePresentationTime (assuming the content progressed at specified playback speed).

In all cases, the net result shall be that after the content begins to play, each fragment of content is presented to the end-user at the exact point in time according to the specified ReferencePosition and ReferencePresentationTime (in other words, according to the time-anchor). In a time-based seek, if the exact instant requested by ReferencePosition falls inside a frame, the renderer might need to round it off. In this case, the renderer shall start rendering the content at the earliest instant possible. If the device is unable to achieve the requested playback behavior, the device shall return an error code 732 (“Unable to calculate sync point”). Additionally, if the device is unable to render the content at the specified speed, then this action shall return error code 717 (“Play speed not supported”).

5.4.26.1 Arguments

Table 73 — Arguments for SyncPlay()

| Argument | | |
|----------------------------------|-----------|------------------------------------|
| <u>InstanceID</u> | <u>IN</u> | <u>A_ARG_TYPE_InstanceID</u> |
| <u>Speed</u> | <u>IN</u> | <u>TransportPlaySpeed</u> |
| <u>ReferencePositionUnits</u> | <u>IN</u> | <u>A_ARG_TYPE_SeekMode</u> |
| <u>ReferencePosition</u> | <u>IN</u> | <u>A_ARG_TYPE_SeekTarget</u> |
| <u>ReferencePresentationTime</u> | <u>IN</u> | <u>A_ARG_TYPE_PresentationTime</u> |
| <u>ReferenceClockId</u> | <u>IN</u> | <u>A_ARG_TYPE_ClockId</u> |

5.4.26.2 Dependency on State

This action is allowed in the “STOPPED”, “PLAYING”, and “PAUSED_PLAYBACK” transport states. In other states the action may also succeed or it may fail with error code 701.

5.4.26.3 Effect on State

This action changes TransportState to “PLAYING” and TransportPlaySpeed to the value specified in the Speed argument of the Play() action. If it would take a noticeable amount of time before a human user would actually see or hear the media playing, the AVTransport may temporarily go to the “TRANSITIONING” state before going to “PLAYING”. This might be appropriate, for example, for devices that need to start buffering or completely download the media before playback can start.

5.4.26.4 Errors

Table 74 — Error Codes for *SyncPlay()*

| errorCode | errorDescription | Description |
|-----------|---|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 702 | No contents | The media does not contain any contents that can be played. |
| 703 | Read error | The media cannot be read (for example, because of dust or a scratch). |
| 704 | Format not supported for playback | The storage format of the currently loaded media is not supported for playback by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (the transport is hold locked), the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 710 | Seek mode not supported | The specified seek mode is not supported by the device. |
| 711 | Illegal seek target | The specified seek target is not present on the media or is not specified in terms of the seek mode. |
| 714 | Illegal MIME-type | The resource to be played has a MIME-type which is not supported by the AVTransport service. |
| 715 | Content 'BUSY' | This indicates that the resource is already in use at this time. |
| 716 | Resource not found | The resource to be played cannot be found in the network. |
| 717 | Play speed not supported | The specified playback speed is not supported by the AVTransport service. |
| 718 | Invalid <i>InstanceID</i> | The specified <i>InstanceID</i> is invalid for this AVTransport. |
| 719 | DRM error | The action failed because an unspecified DRM error occurred. |
| 720 | Expired content | The action failed because the content use validity interval has expired. |
| 721 | Non-allowed use | The action failed because the requested content use is disallowed. |
| 722 | Can't determine allowed uses | The action failed because the allowed content uses cannot be verified. |
| 723 | Exhausted allowed use | The action failed because the number of times this content has been used as requested has reached the maximum allowed number of uses. |
| 724 | Device authentication failure | The action failed because of a device authentication failure between the media source device and the media sink device. |
| 725 | Device revocation | The action failed because either the media source device or the media sink device has been revoked. |
| 731 | Invalid time, offset, or position value | The action failed because the supplied time, offset, or position value for an argument was not valid. |
| 732 | Unable to calculate sync point | The action failed because the system was not able to calculate a synchronization point using the supplied time, offset, or position information. |

5.4.27 *SyncStop()*

This conditionally required action shall be implemented if the AVTransport service supports the ConnectionManager *CLOCKSYNC* feature, otherwise it is not allowed. The *SyncStop()* action behaves the same as the *Stop()* action except that the content is stopped at the specified time as indicated by the *StopTime* input argument. If the device is not able to stop at the exact time specified (for example, the device can only stop at the beginning of the next

video frame), the device shall stop the content as quickly as possible but after the specified stop time.

If the specified stop time has already passed (for example, if the action request is received after the specified stop time), the [SyncStop\(\)](#) action shall stop the playback of content as soon as possible.

5.4.27.1 Arguments

Table 75 — Arguments for [SyncStop\(\)](#)

| Argument | Direction | relatedStateVariable |
|----------------------------------|--------------------|---|
| InstanceID | IN | A_ARG_TYPE_InstanceID |
| StopTime | IN | A_ARG_TYPE_PresentationTime |
| ReferenceClockId | IN | A_ARG_TYPE_ClockId |

5.4.27.2 Dependency on State

This action is allowed in all transport states except in state "[NO_MEDIA_PRESENT](#)".

5.4.27.3 Effect on State

This action changes [TransportState](#) to "[STOPPED](#)". If it would take a noticeable amount of time before a human user would actually see or hear the media has stopped, the AVTransport may temporarily go to the "[TRANSITIONING](#)" state before going to "[STOPPED](#)".

5.4.27.4 Errors

Table 76 — Error Codes for [SyncStop\(\)](#)

| errorCode | errorDescription | Description |
|-----------|--|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (the transport is hold locked), the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 718 | Invalid InstanceID | The specified InstanceID is invalid for this AVTransport. |
| 731 | Invalid time, offset, or position value | The action failed because the supplied time, offset, or position value for an argument was not valid. |
| 733 | Sync, position, or offset too early or small | The action failed because the specified or calculated synchronization point, time, or position occurred too quickly (or in the past) for the device to complete the action. |

5.4.28 [SyncPause\(\)](#)

This conditionally allowed action may be implemented if the AVTransport service supports the ConnectionManager [CLOCKSYNC](#) feature. Otherwise it is not allowed. The [SyncPause\(\)](#) action behaves the same as the [Pause\(\)](#) action except that the content is paused at the specified time as indicated by the [PauseTime](#) input argument. If the device is not able to pause at the exact time specified (for example, the device can only pause at the beginning of the next video frame), the device shall pause the content as quickly as possible but after the specified pause time.

If the adjusted pause time has already passed (for example, if the action request is received after the adjusted pause time), the [SyncPause\(\)](#) action shall pause as soon as possible.

5.4.28.1 Arguments

Table 77 — Arguments for [SyncPause\(\)](#)

| Argument | Direction | relatedStateVariable |
|----------------------------------|-----------|---|
| InstanceID | <i>IN</i> | A_ARG_TYPE_InstanceID |
| PauseTime | <i>IN</i> | A_ARG_TYPE_PresentationTime |
| ReferenceClockId | <i>IN</i> | A_ARG_TYPE_ClockId |

5.4.28.2 Dependency on State

This action is always allowed while playing. In other cases, the action may fail with error code 701.

5.4.28.3 Effect on State

When playing, this action changes [TransportState](#) to “[PAUSED_PLAYBACK](#)”. The [SyncPause\(\)](#) action does not operate as a toggle. If it would take a noticeable amount of time before a human user would actually see or hear the media is paused, the AVTransport may temporarily go to the “[TRANSITIONING](#)” state before going to “[PAUSED_PLAYBACK](#)”.

5.4.28.4 Errors

Table 78 — Error Codes for [SyncPause\(\)](#)

| errorCode | errorDescription | Description |
|-----------|--|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (the transport is hold locked), the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 718 | Invalid InstanceID | The specific InstanceID is invalid for the AVTransport. |
| 731 | Invalid time, offset, or position value | The action failed because the supplied time, offset, or position value for an argument was not valid. |
| 733 | Sync, position, or offset too early or small | The action failed because the specified or calculated synchronization point, time, or position occurred too quickly (or in the past) for the device to complete the action. |

5.4.29 [SetStaticPlaylist\(\)](#)

This allowed action passes a static playlist document to the device. The playlist may be delivered using multiple invocations of the [SetStaticPlaylist\(\)](#) action to conform to any limitations on SOAP packet sizes.

The value in the [PlaylistData](#) argument provides a chunk of a playlist. Terminating characters shall not be appended to the [PlaylistData](#) value unless these characters appear in the original playlist. The value in the [PlaylistDataLength](#) argument identifies the length of the [PlaylistData](#) argument value provided.

The value in the [PlaylistOffset](#) argument provides the offset into the playlist document being passed to the renderer. The initial invocation of the [SetStaticPlaylist\(\)](#) action the [PlaylistOffset](#) argument shall be zero. If multiple invocations of the [SetStaticPlaylist\(\)](#) action are needed, the [PlaylistOffset](#) argument shall be increased by the [PlaylistDataLength](#) argument from the last invocation.

The value in the [PlaylistTotalLength](#) argument identifies the aggregate total length of playlist data using a series one or more invocations of the [SetStaticPlaylist\(\)](#) action. The series

begins with a [SetStaticPlaylist\(\)](#) with a [PlaylistOffset](#) argument value of zero and ends when the total [PlaylistDataLength](#) is reached. When the length of playlist data indicated by the [PlaylistTotalLength](#) argument is provided, syntactic and/or semantic checks (see note) on the playlist should be performed. Playlist data provided in excess of [PlaylistDataLength](#) shall be considered an error.

A [SetStaticPlaylist\(\)](#) action with a [PlaylistTotalLength](#) argument of 0 shall reset playlist processing, including discarding any pending playlist data. In this case, the action argument values other than [InstanceID](#) shall be ignored.

If a device implements the [SetStaticPlaylist\(\)](#) action, then the [GetPlaylistInfo\(\)](#) action shall also be implemented and shall support the <staticPlaylistInfo> element of the returned [A_ARG_TYPE_PlaylistInfo](#) XML document. In addition, the [PlaylistType](#) argument shall support the [Static](#) allowed value.

The [PlaylistStartObj](#) and [PlaylistStartGroup](#) arguments provide a starting object [@id](#) and starting group ID for playlists which employ object linking properties. If these arguments are non-empty, then the device should process playlist elements in the order specified by the [objectLink@nextObjID](#) and [objectLink@prevObjID](#) elements for the indicated object linking GroupID. For playlists that do not employ object linking properties, these arguments should be set to "". See the ContentDirectory service specification [7] for further details on object linking metadata properties.

Note that syntactic/semantic checks should be restricted to validation that can be performed using the playlist data. These checks can include whether an XML based playlist is well-formed, and/or that all the items referenced by an object linking playlist were included. However, checks should not be performed that involve retrieving media objects associated with the playlist.

5.4.29.1 Arguments

Table 79 — Arguments for [SetStaticPlaylist\(\)](#)

| Argument | Direction | relatedStateVariable |
|--------------------------------------|--------------------|---|
| InstanceID | IN | A_ARG_TYPE_InstanceID |
| PlaylistData | IN | A_ARG_TYPE_PlaylistData |
| PlaylistDataLength | IN | A_ARG_TYPE_PlaylistDataLength |
| PlaylistOffset | IN | A_ARG_TYPE_PlaylistOffset |
| PlaylistTotalLength | IN | A_ARG_TYPE_PlaylistTotalLength |
| PlaylistMIMEType | IN | A_ARG_TYPE_PlaylistMIMEType |
| PlaylistExtendedType | IN | A_ARG_TYPE_PlaylistExtendedType |
| PlaylistStartObj | IN | A_ARG_TYPE_PlaylistStartObjID |
| PlaylistStartGroup | IN | A_ARG_TYPE_PlaylistStartGroupID |

5.4.29.2 Dependency on State

None.

5.4.29.3 Effect on State

When a playlist delivery operation is initiated but has not completed, the renderer shall set its TransportState state variable to the value "[TRANSITIONING](#)". When the playlist deliver operation completes the renderer should transition to its previously established state, i.e. "[STOPPED](#)" or "[PLAYING](#)".

Static playlists are considered track-aware media. Specifically the [Next\(\)](#) and [Previous\(\)](#) actions should be supported. However, since static playlists do not use numeric track numbers, the [CurrentTrack](#) state variable shall be set to 0.

The AVTransport service shall set the [CurrentTrackDuration](#), [CurrentTrackURI](#) and [CurrentTrackMetadata](#) state variables to the values corresponding to the current playlist XML element when processing static playlists. The [GetPositionInfo\(\)](#) action can be used to obtain current values for these state variables during static playlist operations.

The AVTransport service shall generate a [LastChange](#) event after modifications to the [CurrentTrackDuration](#), [CurrentTrackURI](#) or [CurrentTrackMetadata](#) state variables.

5.4.29.4 Errors

Table 80 — Error Codes for [SetStaticPlaylist\(\)](#)

| errorCode | errorDescription | Description |
|-----------|--|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 714 | Illegal MIME-type | The specified resource has a MIME-type which is not supported by the AVTransport service. |
| 718 | Invalid InstanceID | The specified InstanceID is invalid for this AVTransport. |
| 734 | Illegal PlaylistOffset | The PlaylistOffset specified would result in a missing section of a playlist. The PlaylistOffset for an initial invocation of the SetStaticPlaylist() action is non-zero. |
| 735 | Incorrect Playlist length | A playlist section as defined by PlaylistOffset and PlaylistDataLength exceeds PlaylistTotalLength . The PlaylistDataLength argument value is zero or negative. The PlaylistTotalLength parameter is invalid (negative). The PlaylistTotalLength parameter changed during a series of this action. (Outside of a 0 value to reset static playlist processing). The device does not have sufficient memory capacity to process the playlist. An attempt to issue an operation: Play() , Next() , Prev() , Stop() on a playlist where PlaylistTotalLength was not reached. |
| 736 | Illegal Playlist | The playlist delivered failed syntactic or semantic checks. |

5.4.30 [SetStreamingPlaylist\(\)](#)

This allowed action initiates and maintains a streaming playlist operation to a device. The [PlaylistStep](#) argument indicates whether this action is a request to initiate, continue, end, or reset a streaming playlist operation.

If a device implements the [SetStreamingPlaylist\(\)](#) action, then the [GetPlaylistInfo\(\)](#) action shall be also be implemented and shall support the <streamingPlaylistInfo> element of the returned [A_ARG_TYPE_PlaylistInfo](#) XML document. In addition, the PlaylistType argument shall support the [Streaming](#) allowed value.

The device is responsible for “gracefully” handling conditions associated with renderer playlist underrun(s), i.e. the device exhausts all currently delivered playlist items.

5.4.30.1 Arguments

Table 81 — Arguments for [SetStreamingPlaylist\(\)](#)

| Argument | Direction | relatedStateVariable |
|--------------------------------------|--------------------|---|
| InstanceID | IN | A_ARG_TYPE_InstanceID |
| PlaylistData | IN | A_ARG_TYPE_PlaylistData |
| PlaylistDataLength | IN | A_ARG_TYPE_PlaylistDataLength |
| PlaylistMIMETYPE | IN | A_ARG_TYPE_PlaylistMIMETYPE |
| PlaylistExtendedType | IN | A_ARG_TYPE_PlaylistExtendedType |
| PlaylistStep | IN | A_ARG_TYPE_PlaylistStep |

5.4.30.2 Dependency on State

None.

5.4.30.3 Effect on State

When a playlist delivery operation is initiated, the renderer may set its *TransportState* state variable to the value “*TRANSITIONING*”. As soon as the renderer has determined a *CurrentTrackURI* the renderer should transition to its previously established state, i.e. “*STOPPED*” or “*PLAYING*”.

Streaming playlists are considered track-aware media. Specifically, the *Seek()*, *Next()* and *Previous()* actions should be supported.

The AVTransport service shall set the *CurrentTrack* state variable to facilitate issuing *Seek()* actions within streaming playlists. Note that it is possible some previous playlist tracks are no longer available. The *GetPlaylistInfo()* action can be used to get the accessible track ranges for streaming playlist operations.

The AVTransport service shall also set the *CurrentTrackDuration*, *CurrentTrackURI* and *CurrentTrackMetadata* state variables to values corresponding to the current playlist track when processing streaming playlists. The *GetPositionInfo()* action can also be used to obtain current values for these state variables during streaming playlist operations.

The AVTransport service shall generate a *LastChange* event after modifications to the *CurrentTrack*, *CurrentTrackDuration*, *CurrentTrackURI* or *CurrentTrackMetadata* state variables.

5.4.30.4 Errors

Table 82 — Error Codes for *SetStreamingPlaylist()*

| errorCode | errorDescription | Description |
|-----------|---------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 714 | Illegal MIME-type | The specified resource has a MIME-type which is not supported by the AVTransport service. |
| 718 | Invalid <i>InstanceID</i> | The specified <i>InstanceID</i> is invalid for this AVTransport. |
| 735 | Incorrect Playlist length | The <i>PlaylistDataLength</i> argument value is zero or negative. The device does not have sufficient memory capacity to process the playlist. |
| 736 | Illegal Playlist | The playlist delivered failed syntactic or semantic checks. |

5.4.31 *GetPlaylistInfo()*

This conditionally required action shall be implemented if the AVTransport service supports either the *SetStaticPlaylist()* or *SetStreamingPlaylist()* actions, otherwise it is not allowed. This action provides information about a streaming or static playlist operation. The *PlaylistType* argument indicates the target playlist type. The requested playlist information is returned in the *PlaylistInfo* argument as an XML document.

5.4.31.1 Arguments

Table 83 — Arguments for *GetPlaylistInfo()*

| Argument | Direction | relatedStateVariable |
|---------------------|------------|--------------------------------|
| <i>InstanceID</i> | <i>IN</i> | <i>A_ARG_TYPE InstanceID</i> |
| <i>PlaylistType</i> | <i>IN</i> | <i>A_ARG_TYPE PlaylistType</i> |
| <i>PlaylistInfo</i> | <i>OUT</i> | <i>A_ARG_TYPE PlaylistInfo</i> |

5.4.31.2 Dependency on State

None.

5.4.31.3 Effect on State

None.

5.4.31.4 Errors**Table 84 — Error Codes for GetPlaylistInfo()**

| errorCode | errorDescription | Description |
|------------------|---------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 714 | Illegal MIME-type | The specified resource has a MIME-type which is not supported by the service. |
| 718 | Invalid <u>InstanceID</u> | The specified <u>InstanceID</u> is invalid for this AVTransport. |

5.4.32 Common Error Codes

Table 85 below lists error codes common to actions for this service type. If an action results in multiple errors, the most specific error should be returned.

Table 85 — Common Error Codes

| errorCode | errorDescription | Description |
|-----------|------------------------------------|---|
| 400-499 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 500-599 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 600-699 | TBD | See clause 3 in UPnP Device Architecture [14]. |
| 701 | Transition not available | The immediate transition from current transport state to desired transport state is not supported by this device. |
| 702 | No contents | The media does not contain any contents that can be played. |
| 703 | Read error | The media cannot be read (for example, because of dust or a scratch). |
| 704 | Format not supported for playback | The storage format of the currently loaded media is not supported for playback by this device. |
| 705 | Transport is locked | The transport is <i>hold locked</i> . (Some portable mobile devices have a small mechanical toggle switch called a <i>hold lock switch</i> . While this switch is ON (he transport is hold locked) the device is guarded against operations such as accidental power on when not in use, or interruption of play or record from accidental pressing of a front panel button or a GUI button.) |
| 706 | Write error | The media cannot be written. (for example, because of dust or a scratch) |
| 707 | Media is protected or not writable | The media is write-protected or is of a not writable type. |
| 708 | Format not supported for recording | The storage format of the currently loaded media is not supported for recording by this device. |
| 709 | Media is full | There is no free space left on the loaded media. |
| 710 | Seek mode not supported | The specified seek mode is not supported by the device. |
| 711 | Illegal seek target | The specified seek target is not present on the media or is not specified in terms of the seek mode. |
| 712 | Play mode not supported | The specified play mode is not supported by the device. |
| 713 | Record quality not supported | The specified record quality is not supported by the device. |
| 714 | Illegal MIME-type | The specified resource has a MIME-type which is not supported by the AVTransport service. |
| 715 | Content 'BUSY' | This indicates that the resource is already in use at this time. |
| 716 | Resource not found | The specified resource cannot be found in the network. |
| 717 | Play speed not supported | The specified playback speed is not supported by the AVTransport service. |
| 718 | Invalid <i>InstanceID</i> | The specified <i>InstanceID</i> is invalid for this AVTransport. |
| 719 | DRM error | The action failed because an unspecified DRM error occurred. |
| 720 | Expired content | The action failed because the content use validity interval has expired. |
| 721 | Non-allowed use | The action failed because the requested content use is disallowed. |
| 722 | Can't determine allowed uses | The action failed because the allowed content uses cannot be verified. |
| 723 | Exhausted allowed use | The action failed because the number of times this content has been used as requested has reached the maximum allowed number of uses. |
| 724 | Device authentication failure | The action failed because of a device authentication failure between the media source device and the media sink device. |
| 725 | Device revocation | The action failed because either the media source device or the media sink device has been revoked. |
| 726 | Invalid <i>StateVariableList</i> | Some of the variables are invalid. |

| errorCode | errorDescription | Description |
|-----------|--|---|
| 727 | Ill-formed CSV List | The CSV list is not well formed. |
| 728 | Invalid <u>State Variable Value</u> | One of the StateVariableValuePairs contains an invalid value. |
| 729 | Invalid Service Type | The specified ServiceType is invalid. |
| 730 | Invalid Service Id | The specified ServiceId is invalid. |
| 731 | Invalid time, offset, or position value | The action failed because the supplied time, offset, or position value for an argument was not valid. |
| 732 | Unable to calculate sync point | The action failed because the system was not able to calculate a synchronization point using the supplied time, offset, or position information. |
| 733 | Sync, position, or offset too early or small | The action failed because the specified or calculated synchronization point, time, or position occurred too quickly (or in the past) for the device to complete the action. |
| 734 | Illegal PlaylistOffset | The PlaylistOffset specified would result in a missing section of a playlist. The PlaylistOffset for an initial invocation of the SetStaticPlaylist() action is non-zero. |
| 735 | Incorrect Playlist length | Playlist length is incorrect or exceeds storage capacity of device. See action description for specific error conditions. |
| 736 | Illegal Playlist | The playlist delivered failed syntactic or semantic checks. |
| 737 | No DNS Server | The DNS Server is not available (HTTP error 503). |
| 738 | Bad Domain Name | Unable to resolve the Fully Qualified Domain Name (HTTP error 502). |
| 739 | Server Error | The server that hosts the resource is unreachable or unresponsive (HTTP error 404/410). |

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

Note that 800-899 Error Codes are not permitted for standard actions. See clause 3 of the UPnP Device Architecture [14] for more details.

6 XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>SetAVTransportURI</name>
      <argumentList>
        <argument>
          <name>InstanceID</name>
          <direction>in</direction>
          <relatedStateVariable>
            A_ARG_TYPE_InstanceID
          </relatedStateVariable>
        </argument>
        <argument>
          <name>CurrentURI</name>
          <direction>in</direction>
          <relatedStateVariable>
            AVTransportURI
          </relatedStateVariable>
        </argument>
      </argumentList>
    </action>
  </actionList>
</scpd>
```

```

        <name>CurrentURIMetaData</name>
        <direction>in</direction>
        <relatedStateVariable>
            AVTransportURIMetaData
        </relatedStateVariable>
    </argument>
</argumentList>
</action>
<action>
    <name>SetNextAVTransportURI</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>NextURI</name>
            <direction>in</direction>
            <relatedStateVariable>
                NextAVTransportURI
            </relatedStateVariable>
        </argument>
        <argument>
            <name>NextURIMetaData</name>
            <direction>in</direction>
            <relatedStateVariable>
                NextAVTransportURIMetaData
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>GetMediaInfo</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>NrTracks</name>
            <direction>out</direction>
            <relatedStateVariable>
                NumberOfTracks
            </relatedStateVariable>
        </argument>
        <argument>
            <name>MediaDuration</name>
            <direction>out</direction>
            <relatedStateVariable>
                CurrentMediaDuration
            </relatedStateVariable>
        </argument>
        <argument>
            <name>CurrentURI</name>
            <direction>out</direction>
            <relatedStateVariable>
                AVTransportURI
            </relatedStateVariable>
        </argument>
        <argument>
            <name>CurrentURIMetaData</name>
            <direction>out</direction>

```

```

        <relatedStateVariable>
          AVTransportURIMetaData
        </relatedStateVariable>
      </argument>
    <argument>
      <name>NextURI</name>
      <direction>out</direction>
      <relatedStateVariable>
        NextAVTransportURI
      </relatedStateVariable>
    </argument>
    <argument>
      <name>NextURIMetaData</name>
      <direction>out</direction>
      <relatedStateVariable>
        NextAVTransportURIMetaData
      </relatedStateVariable>
    </argument>
    <argument>
      <name>PlayMedium</name>
      <direction>out</direction>
      <relatedStateVariable>
        PlaybackStorageMedium
      </relatedStateVariable>
    </argument>
    <argument>
      <name>RecordMedium</name>
      <direction>out</direction>
      <relatedStateVariable>
        RecordStorageMedium
      </relatedStateVariable>
    </argument>
    <argument>
      <name>WriteStatus</name>
      <direction>out</direction>
      <relatedStateVariable>
        RecordMediumWriteStatus
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>GetMediaInfo_Ext</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
    <argument>
      <name>CurrentType</name>
      <direction>out</direction>
      <relatedStateVariable>
        CurrentMediaCategory
      </relatedStateVariable>
    </argument>
    <argument>
      <name>NrTracks</name>
      <direction>out</direction>
      <relatedStateVariable>
        NumberOfTracks
      </relatedStateVariable>
    </argument>
    <argument>
      <name>MediaDuration</name>
      <direction>out</direction>

```

```

        <relatedStateVariable>
            CurrentMediaDuration
        </relatedStateVariable>
    </argument>
    <argument>
        <name>CurrentURI</name>
        <direction>out</direction>
        <relatedStateVariable>
            AVTransportURI
        </relatedStateVariable>
    </argument>
    <argument>
        <name>CurrentURIMetaData</name>
        <direction>out</direction>
        <relatedStateVariable>
            AVTransportURIMetaData
        </relatedStateVariable>
    </argument>
    <argument>
        <name>NextURI</name>
        <direction>out</direction>
        <relatedStateVariable>
            NextAVTransportURI
        </relatedStateVariable>
    </argument>
    <argument>
        <name>NextURIMetaData</name>
        <direction>out</direction>
        <relatedStateVariable>
            NextAVTransportURIMetaData
        </relatedStateVariable>
    </argument>
    <argument>
        <name>PlayMedium</name>
        <direction>out</direction>
        <relatedStateVariable>
            PlaybackStorageMedium
        </relatedStateVariable>
    </argument>
    <argument>
        <name>RecordMedium</name>
        <direction>out</direction>
        <relatedStateVariable>
            RecordStorageMedium
        </relatedStateVariable>
    </argument>
    <argument>
        <name>WriteStatus</name>
        <direction>out</direction>
        <relatedStateVariable>
            RecordMediumWriteStatus
        </relatedStateVariable>
    </argument>
</argumentList>
</action>
<action>
    <name>GetTransportInfo</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>CurrentTransportState</name>
            <direction>out</direction>

```

```

        <relatedStateVariable>
            TransportState
        </relatedStateVariable>
    </argument>
    <argument>
        <name>CurrentTransportStatus</name>
        <direction>out</direction>
        <relatedStateVariable>
            TransportStatus
        </relatedStateVariable>
    </argument>
    <argument>
        <name>CurrentSpeed</name>
        <direction>out</direction>
        <relatedStateVariable>
            TransportPlaySpeed
        </relatedStateVariable>
    </argument>
</argumentList>
</action>
<action>
    <name>GetPositionInfo</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>Track</name>
            <direction>out</direction>
            <relatedStateVariable>
                CurrentTrack
            </relatedStateVariable>
        </argument>
        <argument>
            <name>TrackDuration</name>
            <direction>out</direction>
            <relatedStateVariable>
                CurrentTrackDuration
            </relatedStateVariable>
        </argument>
        <argument>
            <name>TrackMetaData</name>
            <direction>out</direction>
            <relatedStateVariable>
                CurrentTrackMetaData
            </relatedStateVariable>
        </argument>
        <argument>
            <name>TrackURI</name>
            <direction>out</direction>
            <relatedStateVariable>
                CurrentTrackURI
            </relatedStateVariable>
        </argument>
        <argument>
            <name>RelTime</name>
            <direction>out</direction>
            <relatedStateVariable>
                RelativeTimePosition
            </relatedStateVariable>
        </argument>
        <argument>
            <name>AbsTime</name>
            <direction>out</direction>

```

```

        <relatedStateVariable>
            AbsoluteTimePosition
        </relatedStateVariable>
    </argument>
    <argument>
        <name>RelCount</name>
        <direction>out</direction>
        <relatedStateVariable>
            RelativeCounterPosition
        </relatedStateVariable>
    </argument>
    <argument>
        <name>AbsCount</name>
        <direction>out</direction>
        <relatedStateVariable>
            AbsoluteCounterPosition
        </relatedStateVariable>
    </argument>
</argumentList>
</action>
<action>
    <name>GetDeviceCapabilities</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>PlayMedia</name>
            <direction>out</direction>
            <relatedStateVariable>
                PossiblePlaybackStorageMedia
            </relatedStateVariable>
        </argument>
        <argument>
            <name>RecMedia</name>
            <direction>out</direction>
            <relatedStateVariable>
                PossibleRecordStorageMedia
            </relatedStateVariable>
        </argument>
        <argument>
            <name>RecQualityModes</name>
            <direction>out</direction>
            <relatedStateVariable>
                PossibleRecordQualityModes
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>GetTransportSettings</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>PlayMode</name>
            <direction>out</direction>
            <relatedStateVariable>
                CurrentPlayMode
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>

```

```

        </relatedStateVariable>
    </argument>
    <argument>
        <name>RecQualityMode</name>
        <direction>out</direction>
        <relatedStateVariable>
            CurrentRecordQualityMode
        </relatedStateVariable>
    </argument>
</argumentList>
</action>
<action>
    <name>Stop</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>Play</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>Speed</name>
            <direction>in</direction>
            <relatedStateVariable>
                TransportPlaySpeed
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>Pause</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>Record</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>Seek</name>

```

```

<argumentList>
  <argument>
    <name>InstanceID</name>
    <direction>in</direction>
    <relatedStateVariable>
      A_ARG_TYPE_InstanceID
    </relatedStateVariable>
  </argument>
  <argument>
    <name>Unit</name>
    <direction>in</direction>
    <relatedStateVariable>
      A_ARG_TYPE_SeekMode
    </relatedStateVariable>
  </argument>
  <argument>
    <name>Target</name>
    <direction>in</direction>
    <relatedStateVariable>
      A_ARG_TYPE_SeekTarget
    </relatedStateVariable>
  </argument>
</argumentList>
</action>
<action>
  <name>Next</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>Previous</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>SetPlayMode</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
    <argument>
      <name>NewPlayMode</name>
      <direction>in</direction>
      <relatedStateVariable>
        CurrentPlayMode
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>

```



```

<name>SetRecordQualityMode</name>
<argumentList>
  <argument>
    <name>InstanceID</name>
    <direction>in</direction>
    <relatedStateVariable>
      A_ARG_TYPE_InstanceID
    </relatedStateVariable>
  </argument>
  <argument>
    <name>NewRecordQualityMode</name>
    <direction>in</direction>
    <relatedStateVariable>
      CurrentRecordQualityMode
    </relatedStateVariable>
  </argument>
</argumentList>
</action>
<action>
  <name>GetCurrentTransportActions</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
    <argument>
      <name>Actions</name>
      <direction>out</direction>
      <relatedStateVariable>
        CurrentTransportActions
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>GetDRMState</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
    <argument>
      <name>CurentDRMState</name>
      <direction>out</direction>
      <relatedStateVariable>
        DRMState
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>GetStateVariables</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
    <argument>
      <name>StateVariableList</name>

```

```

        <direction>in</direction>
        <relatedStateVariable>
            A_ARG_TYPE_StateVariableList
        </relatedStateVariable>
    </argument>
    <argument>
        <name>StateVariableValuePairs</name>
        <direction>out</direction>
        <relatedStateVariable>
            A_ARG_TYPE_StateVariableValuePairs
        </relatedStateVariable>
    </argument>
</argumentList>
</action>
<action>
    <name>SetStateVariables</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>AVTransportUDN</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_DeviceUDN
            </relatedStateVariable>
        </argument>
        <argument>
            <name>ServiceType</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_ServiceType
            </relatedStateVariable>
        </argument>
        <argument>
            <name>ServiceId</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_ServiceID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>StateVariableValuePairs</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_StateVariableValuePairs
            </relatedStateVariable>
        </argument>
        <argument>
            <name>StateVariableList</name>
            <direction>out</direction>
            <relatedStateVariable>
                A_ARG_TYPE_StateVariableList
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>SetStaticPlaylist</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>

```

```

        A_ARG_TYPE_InstanceID
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistData</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistData
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistDataLength</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistDataLength
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistOffset</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistOffset
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistTotalLength</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistTotalLength
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistMIMETYPE</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistMIMETYPE
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistExtendedType</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistExtendedType
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistStartObj</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistStartObjID
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistStartGroup</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistStartGroupID
    </relatedStateVariable>
</argument>
</argumentList>
</action>
<action>
    <name>SetStreamingPlaylist</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>

```

```

        A_ARG_TYPE_InstanceID
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistData</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistData
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistDataLength</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistDataLength
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistMIMETYPE</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistMIMETYPE
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistExtendedType</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistExtendedType
    </relatedStateVariable>
</argument>
<argument>
    <name>PlaylistStep</name>
    <direction>in</direction>
    <relatedStateVariable>
        A_ARG_TYPE_PlaylistStep
    </relatedStateVariable>
</argument>
</argumentList>
</action>
<action>
    <name>GetPlaylistInfo</name>
    <argumentList>
        <argument>
            <name>InstanceID</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_InstanceID
            </relatedStateVariable>
        </argument>
        <argument>
            <name>PlaylistType</name>
            <direction>in</direction>
            <relatedStateVariable>
                A_ARG_TYPE_PlaylistType
            </relatedStateVariable>
        </argument>
        <argument>
            <name>PlaylistInfo</name>
            <direction>out</direction>
            <relatedStateVariable>
                A_ARG_TYPE_PlaylistInfo
            </relatedStateVariable>
        </argument>
    </argumentList>
</action>
<action>
    <name>AdjustSyncOffset</name>

```

```

<argumentList>
  <argument>
    <name>InstanceID</name>
    <direction>in</direction>
    <relatedStateVariable>
      A_ARG_TYPE_InstanceID
    </relatedStateVariable>
  </argument>
  <argument>
    <name>Adjustment</name>
    <direction>in</direction>
    <relatedStateVariable>
      A_ARG_TYPE_SyncOffsetAdj
    </relatedStateVariable>
  </argument>
</argumentList>
</action>
<action>
  <name>GetSyncOffset</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
    <argument>
      <name>CurrentSyncOffset</name>
      <direction>out</direction>
      <relatedStateVariable>
        SyncOffset
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>SetSyncOffset</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
    <argument>
      <name>NewSyncOffset</name>
      <direction>in</direction>
      <relatedStateVariable>
        SyncOffset
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>SyncPlay</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
    <argument>
      <name>Speed</name>
      <direction>in</direction>

```

```

        <relatedStateVariable>
          TransportPlaySpeed
        </relatedStateVariable>
      </argument>
    <argument>
      <name>ReferencePositionUnits</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_SeekMode
      </relatedStateVariable>
    </argument>
    <argument>
      <name>ReferencePosition</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_SeekTarget
      </relatedStateVariable>
    </argument>
    <argument>
      <name>ReferencePresentationTime</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_PresentationTime
      </relatedStateVariable>
    </argument>
    <argument>
      <name>ReferenceClockId</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_ClockId
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>SyncPause</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID
      </relatedStateVariable>
    </argument>
    <argument>
      <name>PauseTime</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_PresentationTime
      </relatedStateVariable>
    </argument>
    <argument>
      <name>ReferenceClockId</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_ClockId
      </relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>SyncStop</name>
  <argumentList>
    <argument>
      <name>InstanceID</name>
      <direction>in</direction>
      <relatedStateVariable>
        A_ARG_TYPE_InstanceID

```

```

    </relatedStateVariable>
  </argument>
  <argument>
    <name>StopTime</name>
    <direction>in</direction>
    <relatedStateVariable>
      A_ARG_TYPE_PresentationTime
    </relatedStateVariable>
  </argument>
  <argument>
    <name>ReferenceClockId</name>
    <direction>in</direction>
    <relatedStateVariable>
      A_ARG_TYPE_ClockId
    </relatedStateVariable>
  </argument>
</argumentList>
</action>
</actionList>
<serviceStateTable>
  <stateVariable sendEvents="no">
    <name>TransportState</name>
    <dataType>string</dataType>
    <allowedValueList>
      <allowedValue>STOPPED</allowedValue>
      <allowedValue>PAUSED_PLAYBACK</allowedValue>
      <allowedValue>PAUSED_RECORDING</allowedValue>
      <allowedValue>PLAYING</allowedValue>
      <allowedValue>RECORDING</allowedValue>
      <allowedValue>TRANSITIONING</allowedValue>
      <allowedValue>NO_MEDIA_PRESENT</allowedValue>
      <allowedValue>vendor-defined </allowedValue>
    </allowedValueList>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>TransportStatus</name>
    <dataType>string</dataType>
    <allowedValueList>
      <allowedValue>OK</allowedValue>
      <allowedValue>ERROR_OCCURRED</allowedValue>
      <allowedValue>vendor-defined </allowedValue>
    </allowedValueList>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>CurrentMediaCategory</name>
    <dataType>string</dataType>
    <allowedValueList>
      <allowedValue>NO_MEDIA</allowedValue>
      <allowedValue>TRACK_AWARE</allowedValue>
      <allowedValue>TRACK_UNAWARE</allowedValue>
    </allowedValueList>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>PlaybackStorageMedium</name>
    <dataType>string</dataType>
    <allowedValueList>
      <allowedValue>UNKNOWN</allowedValue>
      <allowedValue>DV</allowedValue>
      <allowedValue>MINI-DV</allowedValue>
      <allowedValue>VHS</allowedValue>
      <allowedValue>W-VHS</allowedValue>
      <allowedValue>S-VHS</allowedValue>
      <allowedValue>D-VHS</allowedValue>
      <allowedValue>VHSC</allowedValue>
      <allowedValue>VIDEO8</allowedValue>
      <allowedValue>HI8</allowedValue>
      <allowedValue>CD-ROM</allowedValue>
      <allowedValue>CD-DA</allowedValue>
    </allowedValueList>
  </stateVariable>

```

```

    <allowedValue>CD-R</allowedValue>
    <allowedValue>CD-RW</allowedValue>
    <allowedValue>VIDEO-CD</allowedValue>
    <allowedValue>SACD</allowedValue>
    <allowedValue>MD-AUDIO</allowedValue>
    <allowedValue>MD-PICTURE</allowedValue>
    <allowedValue>DVD-ROM</allowedValue>
    <allowedValue>DVD-VIDEO</allowedValue>
    <allowedValue>DVD+R</allowedValue>
    <allowedValue>DVD-R</allowedValue>
    <allowedValue>DVD+RW</allowedValue>
    <allowedValue>DVD-RW</allowedValue>
    <allowedValue>DVD-RAM</allowedValue>
    <allowedValue>DVD-AUDIO</allowedValue>
    <allowedValue>DAT</allowedValue>
    <allowedValue>LD</allowedValue>
    <allowedValue>HDD</allowedValue>
    <allowedValue>MICRO-MV</allowedValue>
    <allowedValue>NETWORK</allowedValue>
    <allowedValue>NONE</allowedValue>
    <allowedValue>NOT_IMPLEMENTED</allowedValue>
    <allowedValue>SD</allowedValue>
    <allowedValue>PC-CARD</allowedValue>
    <allowedValue>MMC</allowedValue>
    <allowedValue>CF</allowedValue>
    <allowedValue>BD</allowedValue>
    <allowedValue>MS</allowedValue>
    <allowedValue>HD_DVD</allowedValue>
    <allowedValue> vendor-defined </allowedValue>
  </allowedValueList>
</stateVariable>
<stateVariable sendEvents="no">
  <name>RecordStorageMedium</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>UNKNOWN</allowedValue>
    <allowedValue>DV</allowedValue>
    <allowedValue>MINI-DV</allowedValue>
    <allowedValue>VHS</allowedValue>
    <allowedValue>W-VHS</allowedValue>
    <allowedValue>S-VHS</allowedValue>
    <allowedValue>D-VHS</allowedValue>
    <allowedValue>VHSC</allowedValue>
    <allowedValue>VIDEO8</allowedValue>
    <allowedValue>HI8</allowedValue>
    <allowedValue>CD-ROM</allowedValue>
    <allowedValue>CD-DA</allowedValue>
    <allowedValue>CD-R</allowedValue>
    <allowedValue>CD-RW</allowedValue>
    <allowedValue>VIDEO-CD</allowedValue>
    <allowedValue>SACD</allowedValue>
    <allowedValue>MD-AUDIO</allowedValue>
    <allowedValue>MD-PICTURE</allowedValue>
    <allowedValue>DVD-ROM</allowedValue>
    <allowedValue>DVD-VIDEO</allowedValue>
    <allowedValue>DVD+R</allowedValue>
    <allowedValue>DVD-R</allowedValue>
    <allowedValue>DVD+RW</allowedValue>
    <allowedValue>DVD-RW</allowedValue>
    <allowedValue>DVD-RAM</allowedValue>
    <allowedValue>DVD-AUDIO</allowedValue>
    <allowedValue>DAT</allowedValue>
    <allowedValue>LD</allowedValue>
    <allowedValue>HDD</allowedValue>
    <allowedValue>MICRO-MV</allowedValue>
    <allowedValue>NETWORK</allowedValue>
    <allowedValue>NONE</allowedValue>
    <allowedValue>NOT_IMPLEMENTED</allowedValue>
  </allowedValueList>

```



```

    <allowedValue>SD</allowedValue>
    <allowedValue>PC-CARD</allowedValue>
    <allowedValue>MMC</allowedValue>
    <allowedValue>CF</allowedValue>
    <allowedValue>BD</allowedValue>
    <allowedValue>MS</allowedValue>
    <allowedValue>HD_DVD</allowedValue>
    <allowedValue> vendor-defined </allowedValue>
  </allowedValueList>
</stateVariable>
<stateVariable sendEvents="no">
  <name>PossiblePlaybackStorageMedia</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>PossibleRecordStorageMedia</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>CurrentPlayMode</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>NORMAL</allowedValue>
    <allowedValue>SHUFFLE</allowedValue>
    <allowedValue>REPEAT_ONE</allowedValue>
    <allowedValue>REPEAT_ALL</allowedValue>
    <allowedValue>RANDOM</allowedValue>
    <allowedValue>DIRECT_1</allowedValue>
    <allowedValue>INTRO</allowedValue>
    <allowedValue> vendor-defined </allowedValue>
  </allowedValueList>
  <defaultValue>NORMAL</defaultValue>
</stateVariable>
<stateVariable sendEvents="no">
  <name>TransportPlaySpeed</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>1</allowedValue>
    <allowedValue> vendor-defined </allowedValue>
  </allowedValueList>
  <defaultValue>1</defaultValue>
</stateVariable>
<stateVariable sendEvents="no">
  <name>RecordMediumWriteStatus</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>WRITABLE</allowedValue>
    <allowedValue>PROTECTED</allowedValue>
    <allowedValue>NOT_WRITABLE</allowedValue>
    <allowedValue>UNKNOWN</allowedValue>
    <allowedValue>NOT_IMPLEMENTED</allowedValue>
    <allowedValue> vendor-defined </allowedValue>
  </allowedValueList>
</stateVariable>
<stateVariable sendEvents="no">
  <name>CurrentRecordQualityMode</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>0:EP</allowedValue>
    <allowedValue>1:LP</allowedValue>
    <allowedValue>2:SP</allowedValue>
    <allowedValue>0:BASIC</allowedValue>
    <allowedValue>1:MEDIUM</allowedValue>
    <allowedValue>2:HIGH</allowedValue>
    <allowedValue>NOT_IMPLEMENTED</allowedValue>
    <allowedValue> vendor-defined </allowedValue>
  </allowedValueList>
</stateVariable>

```

```

<stateVariable sendEvents="no">
  <name>PossibleRecordQualityModes</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>NumberOfTracks</name>
  <dataType>ui4</dataType>
  <allowedValueRange>
    <minimum>0</minimum>
    <maximum> vendor-defined </maximum>
  </allowedValueRange>
</stateVariable>
<stateVariable sendEvents="no">
  <name>CurrentTrack</name>
  <dataType>ui4</dataType>
  <allowedValueRange>
    <minimum>0</minimum>
    <maximum> vendor-defined </maximum>
    <step>1</step>
  </allowedValueRange>
</stateVariable>
<stateVariable sendEvents="no">
  <name>CurrentTrackDuration</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>CurrentMediaDuration</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>CurrentTrackMetaData</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>CurrentTrackURI</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>AVTransportURI</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>AVTransportURIMetaData</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>NextAVTransportURI</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>NextAVTransportURIMetaData</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>RelativeTimePosition</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>AbsoluteTimePosition</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>RelativeCounterPosition</name>
  <dataType>i4</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>AbsoluteCounterPosition</name>
  <dataType>ui4</dataType>

```

```

</stateVariable>
<stateVariable sendEvents="no">
  <name>CurrentTransportActions</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>PLAY</allowedValue>
    <allowedValue>STOP</allowedValue>
    <allowedValue>PAUSE</allowedValue>
    <allowedValue>SEEK</allowedValue>
    <allowedValue>NEXT</allowedValue>
    <allowedValue>PREVIOUS</allowedValue>
    <allowedValue>RECORD</allowedValue>
    <allowedValue> vendor-defined </allowedValue>
  </allowedValueList>
</stateVariable>
<stateVariable sendEvents="yes">
  <name>LastChange</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="yes">
  <name>DRMState</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>OK</allowedValue>
    <allowedValue>UNKNOWN</allowedValue>
    <allowedValue>PROCESSING_CONTENT_KEY</allowedValue>
    <allowedValue>CONTENT_KEY_FAILURE</allowedValue>
    <allowedValue>ATTEMPTING_AUTHENTICATION</allowedValue>
    <allowedValue>FAILED_AUTHENTICATION</allowedValue>
    <allowedValue>NOT_AUTHENTICATED</allowedValue>
    <allowedValue>DEVICE_REVOCATION</allowedValue>
    <allowedValue>DRM_SYSTEM_NOT_SUPPORTED</allowedValue>
    <allowedValue>LICENSE_DENIED</allowedValue>
    <allowedValue>LICENSE_EXPIRED</allowedValue>
    <allowedValue>LICENSE_INSUFFICIENT</allowedValue>
  </allowedValueList>
  <defaultValue>UNKNOWN<defaultValue>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_SeekMode</name>
  <dataType>string</dataType>
  <allowedValueList>
    <allowedValue>ABS_TIME</allowedValue>
    <allowedValue>REL_TIME</allowedValue>
    <allowedValue>ABS_COUNT</allowedValue>
    <allowedValue>REL_COUNT</allowedValue>
    <allowedValue>TRACK_NR</allowedValue>
    <allowedValue>CHANNEL_FREQ</allowedValue>
    <allowedValue>TAPE_INDEX</allowedValue>
    <allowedValue>REL_TAPE_INDEX</allowedValue>
    <allowedValue>FRAME</allowedValue>
    <allowedValue>REL_FRAME</allowedValue>
    <allowedValue> vendor-defined </allowedValue>
  </allowedValueList>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_SeekTarget</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_InstanceID</name>
  <dataType>ui4</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_DeviceUDN</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">

```

```

    <name>A_ARG_TYPE_ServiceType</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_ServiceID</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_StateVariableValuePairs</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_StateVariableList</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_PlaylistData</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_PlaylistDataLength</name>
    <dataType>ui4</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_PlaylistOffset</name>
    <dataType>ui4</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_PlaylistTotalLength</name>
    <dataType>ui4</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_PlaylistMIMETYPE</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_PlaylistExtendedType</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_PlaylistStep</name>
    <dataType>string</dataType>
    <allowedValueList>
      <allowedValue>Initial</allowedValue>
      <allowedValue>Continue</allowedValue>
      <allowedValue>Stop</allowedValue>
      <allowedValue>Reset</allowedValue>
      <allowedValue>Replace</allowedValue>
    </allowedValueList>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_PlaylistType</name>
    <dataType>string</dataType>
    <allowedValueList>
      <allowedValue>Static</allowedValue>
      <allowedValue>StaticPlContents</allowedValue>
      <allowedValue>Streaming</allowedValue>
      <allowedValue>StreamingPlContents</allowedValue>
    </allowedValueList>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_PlaylistInfo</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>A_ARG_TYPE_PlaylistStartObjID</name>
    <dataType>string</dataType>
  </stateVariable>

```

```

<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_PlaylistStartGroupID</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>SyncOffset</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_SyncOffsetAdj</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_PresentationTime</name>
  <dataType>string</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>A_ARG_TYPE_ClockId</name>
  <dataType>string</dataType>
</stateVariable>
</serviceStateTable>
</scpd>

```

7 Test

No semantic tests have been specified for this service.

Annex A (normative)

SetAVTransportURI() Protocol Specifics

A.1 Application to HTTP Streaming

A.1.1 AVTransportURI Definition

URIs are well defined for the HTTP scheme in the HyperText Transport Protocol – HTTP/1.1 specification [37].

An example of a content URI for HTTP-based streaming using the GET method, in this case referring to an audio playlist resource, is:

```
http://hostname/audio-content/playlist_10.m3u
```

A.1.2 Control Point Behavior for SetAVTransportURI()

Within the context of invoking actions, control points are encouraged to use the URI that was obtained from the device. It is assumed that this URI is properly escaped, and control points are encouraged to not re-escape or un-escape the obtained URI before invoking an action on the relevant URI. In addition when producing URIs, URIs are expected to be properly escaped according to the Uniform Resource Locators (URL) specification [38].

A.1.3 Implementation of SetAVTransportURI()

While playing, this call instructs the AVTransport service to switch to a different resource, via using the HTTP GET method. It is recommended that the AVTransport checks whether the URI is properly escaped, and if not, escapes the URI itself before sending the HTTP GET request. When the AVTransport is not playing, the device can store the URL and fetch the resource at a later time (for example, when the Play() command is used), or start fetching & buffering immediately. This is device-dependent. For immediate error checking purpose it is recommended to start fetching immediately.

A.1.4 Cleanup

For HTTP connections, many of the underlying TCP/IP socket conventions for cleanup are utilized. An established connection will continue until one of the following occurs:

- The server terminates the socket at the end of a finite length media stream.
- The server terminates the socket as a result of a period of inactivity.

A.2 Application to RTSP/RTP/UDP Streaming

A.2.1 AVTransportURI Definition

The *rtsp* and *rtspu* schemes are used to refer to network resources via the RTSP protocol. This subclause defines the scheme-specific syntax and semantics for RTSP URLs.

```
rtsp_URL      := ('rtsp:'|'rtspu:') '//' host [':' port] [abs_path]
host          := A legal Internet host domain name or IP address (in
                dotted decimal form)
Port          := *DIGIT
```

An example is:

```
rtsp://hostname/video-content/birthdayparty.m2v
```

The scheme *RTSP* needs commands to be issued via a reliable protocol (within the Internet, TCP), while the scheme *RTSPU* identifies an unreliable protocol (within the Internet, UDP).

A.2.2 Control Point behavior for [SetAVTransportURI\(\)](#)

Within the context of invoking actions, control points use the URI that are obtained from the device. It is assumed that this URI is properly escaped, and control points are encouraged to not re-escape or un-escape the obtained URI before sending any RTSP commands. In addition when producing URIs, URIs are expected to be properly escaped according to the Uniform Resource Locators (URL) specification [38].

A.2.3 Implementation of [SetAVTransportURI\(\)](#)

For RTSP, the [SetAVTransportURI\(\)](#) action on the *renderer* device will initiate the creation of a RTSP session. In response to [SetAVTransportURI\(\)](#), the *renderer* sends an RTSP::SETUP message to the RTSP server identified by the URI. It is recommended that the AVTransport checks whether the URI is properly escaped, and if not, escapes the URI itself before sending any RTSP commands. The RTSP::SETUP request for a URI specifies the transport mechanism to be used for the streamed media (for example, RTP, unicast). A client can issue an RTSP::SETUP request for a stream that is already playing to change transport parameters, which a server can support. If the server does not support this, it shall respond with error “455 Method Not Valid In This State”. For the benefit of any intervening firewalls, a client shall indicate the transport parameters even if it has no influence over these parameters, for example, where the server advertises a fixed multicast address.

Since RTSP::SETUP includes all transport initialization information, firewalls and other intermediate network devices (which need this information) are spared the more arduous task of parsing the rtsp::describe response, which has been reserved for media initialization.

An example is (R=renderer, S=server):

```
R->S: SETUP rtsp://example.com/video/birthday.m2v RTSP/1.0
      CSeq: 30.3
      Transport: RTP/AVP;unicast;client_port=4588-4589
S->R: RTSP/1.0 200 OK
      CSeq: 30.3
      Date: 23 Jan 1997 15:35:06 GMT
      Session: 47112344
      Transport: RTP/AVP;unicast;
      client_port=4588-4589;server_port=6256-6257
```

There is no notion of a RTSP *connection*; instead, a server maintains a *session* labeled by an identifier. A RTSP session is in no way tied to a transport-level connection such as a TCP connection. During a RTSP session, a RTSP client can open and close many reliable transport connections to the server to issue RTSP requests. Alternatively, it can use a connectionless transport protocol such as UDP.

A.2.4 Cleanup

When a new URI is specified via [SetAVTransportURI\(\)](#), or when the playback of the current resource finishes, the *renderer* device will terminate the created RTSP session. The *renderer* will send an RTSP::TEARDOWN request to the RTSP server to stop stream delivery for the given session, freeing the resources associated with it. The *renderer* device uses the RTSP session identifier it received during the RTSP setup phase.

An example of a teardown message is (R=renderer, S=server):

```
R->S: TEARDOWN rtsp://example.com/fizzle/foo RTSP/1.0
      CSeq: 892
      Session: 12345678
S->R: RTSP/1.0 200 OK
      CSeq: 892
```

After the server responds OK to the TEARDOWN request, any RTSP session identifier associated with the session will no longer be valid.

A.2.5 Implementation of Transport Controls

Since RTSP is designed for stream control over a network, it defines methods that control the playback of content over a particular RTSP session. These methods (“PLAY”, “PAUSE”, “RECORD”) are sent from the RTSP client (the UPnP renderer device) to the RTSP server. The UPnP AVTransport service can be implemented on top of these methods. For example, when an AVTransport on the RTSP client device receives a *Play()* message (via SOAP), it will send the following message to the RTSP server (R=renderer, S=server):

```
R->S: PLAY rtsp://live.example.com/concert/audio RTSP/1.0
      CSeq: 3
      Session: 0456804596
S->R: RTSP/1.0 200 OK
      CSeq: 3
      Session: 0456804596
```

Time-based seeking can be supported by an RTSP server. The RTSP client can choose to specify a time-range when issuing a PLAY command, see [39] for details. Implementation of the AVTransport’s *Next()* and *Previous()* commands (changes to a new track) will generally need a new RTSP session. URLs for such a new session need to be known by the AVTransport implementation, via, for example, a playlist. This is similar to the HTTP GET case, where an AVTransport can be bound to a playlist URL.

A.3 Application to Internal Streaming

A.3.1 AVTransportURI Definition

An AVTransportURI for an internal protocol is largely unspecified. The only restrictions are:

- The scheme of the URI shall be set to FILE.
- The rest of the URI shall follow the rules for specifying legal URIs with this scheme (see [38]).

Examples of internal AVTransportURIs are:

- file://CD1
- file://CD1?track3
- file://Tuner

A.3.2 Implementation of SetAVTransportURI()

This is device-specific.

A.3.3 Cleanup

This is device-specific.

A.4 Application to IEC61883 Streaming

A.4.1 AVTransportURI Definition

In the case of IEC61883 streaming, the AVTransport instance will exist on the source side of the connection (it is a push protocol). In those cases, the Content URIs exposed by the ContentDirectory on the source device can be device-specific, since they are passed only between that ContentDirectory and the AVTransport service on the same device.

A.4.2 Implementation of SetAVTransportURI()

This is device-specific.

A.4.3 Cleanup

This is device-specific.

A.5 Application to Vendor-specific Streaming

A.5.1 AVTransportURI Definition

An AVTransportURI for a vendor-specific protocol is unspecified. The only restrictions are that it shall follow the rules for specifying legal URIs (see [40]).

A.5.2 Implementation of SetAVTransportURI()

This is vendor-specific.

A.5.3 Cleanup

This is vendor-specific.

Annex B (informative)

Theory of Operation

B.1 TransportState Control

The main functionality of this service is control over the *TransportState* variable. A state machine depicting the relations between AVTransport actions and *TransportState* values is shown below in Figure B.1. In case of any contradictions with the text in the descriptions of the individual actions, the text shall be considered normative.

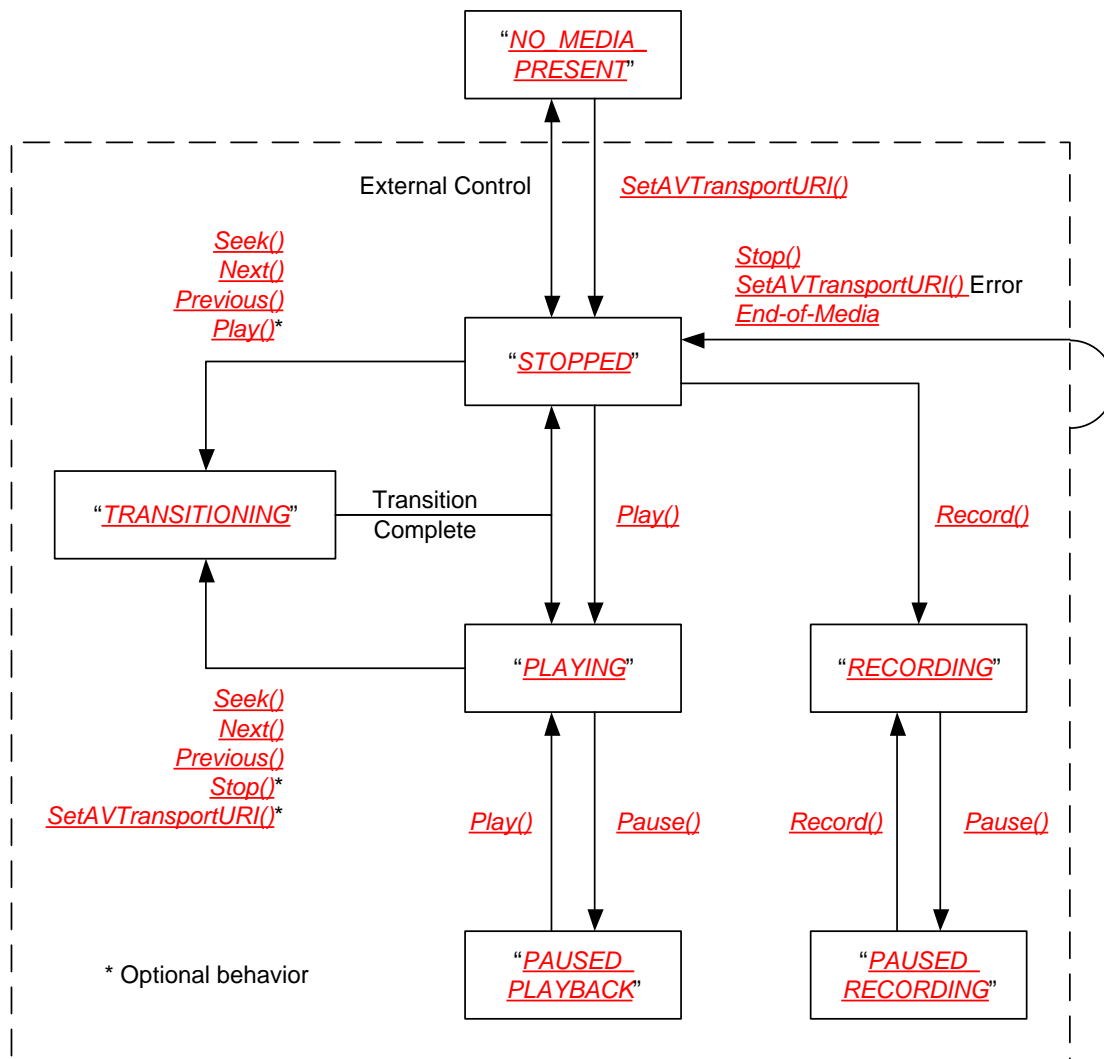


Figure B.1 — *TransportState* Transitions - INFORMATIVE

Note that the *Stop()* action is allowed in all states except *NO_MEDIA_PRESENT*, and returns the *TransportState* to the *STOPPED* value.

The state machine shows the minimal number of transitions that an AVTransport implementation shall implement. In addition, any device vendor may implement more transitions such as, for example, directly from recording to playing mode. For example, nothing prevents a control point from giving a play command during recording, but the AVTransport service simply doesn't need this transition to work, and may return error code 701 (Transition not available). Hence, in such cases, the action might succeed or might not

succeed, and a control point would be encouraged to only attempt the action if it has specific knowledge of that vendor's implementation.

In addition, a device vendor may extend the state diagram above by adding vendor-defined transport states. However, since the semantic meaning of these transport states is not specified, control points that find an AVTransport service in a transport state that they do not understand would be expected to refrain from interacting with that AVTransport service (for example, forcing the service into the "**STOPPED**" state). Rather, they would wait until the service transits back into a transport state that they understand.

Besides restrictions on state transitions that are inherent to the device, there might also be *additional* restrictions depending on the *content* whose playback or recording is being controlled. For example, a live stream coming from a broadcast tuner or Internet Radio station cannot be paused. To assist control points that want to reflect these restrictions in their user interface, an action is defined to return the currently available transport-changing actions – [*GetCurrentTransportActions\(\)*](#) – as a comma-separated list of action names. If a control point invokes a transport state changing action that is not in the list returned by [*GetCurrentTransportActions\(\)*](#), a device will return error code 701 (Transition not available).

B.2 Transport Settings

Besides control over the transport state, the AVTransport also enables control over various settings related to playback and control. These settings, such as play mode, record mode and record quality mode only take effect on subsequent [*Play\(\)*](#) or [*Record\(\)*](#) actions. In other words, they do not change the behavior of any ongoing playback or record *session*.

B.3 Navigation

The AVTransport enables two types of navigation through the media:

- navigation while producing audio and/or video: this is called *playing*.
- navigation while muting audio and/or video: this is called *seeking*.

Both types of navigation are common in the AV domain, both for audio only and for audio/video media.

Playing is allowed at various speeds. A device is required to implement normal speed (x1). Other speeds including *negative speeds* (reverse direction) are allowed.

The [*Seek\(\)*](#) action is very generic and enables a control point to specify a seek operation in various *dimensions*. For example, a control point can instruct a device to position itself 30 seconds from the current position (using the "**RELATIVE TIME**" seek mode) or to go to track number 12 (using the "**TRACK NR**" seek mode).

A device does not need to implement all seek modes described in this template (actual supported modes can be retrieved from the allowed value list of variable [*A_ARG_TYPE_SseekMode*](#)).

Bookmarking functionality could be implemented by a control point depending on the seek modes supported by the device. Action [*GetPositionInfo\(\)*](#) can be used to capture a position on the media, which can be revisited later through action [*Seek\(\)*](#).

B.4 AVTransportURI Concept

Uniform Resource Identifiers (URIs) are the Internet standard for resource identification. URIs are simply character strings which identify abstract or physical resources. The complete URI definition is available in the URI specification [40]. URIs shall be escaped according to the Uniform Resource Locators (URL) specification [38]. In particular, unsafe characters need to be escaped in a valid URI. Characters "<", ">", "\"", "#", and %" are defined unsafe, because they usually have special meanings, and "{", "}", "|", "\\", "^", "~", "[", "]", and "`" are defined unsafe, because they sometimes get corrupted in mail etc.

Every resource that can be played or recorded via an AVTransport will be modeled in the URI syntax. These resources can be *atomic* resources such as a file containing a song, or can be *non-atomic* or *collection* resources. An example of the latter is an audio playlist, an audio CD or the channel-list in a tuner.

Identifying all resources as URIs provides many advantages. First, using URIs for resource identification follows the existing Internet standard. Second, using URIs for all resources (regardless of transport medium, transport scheme, or content-type) unifies the handling for all types of resources which enables cleaner, more obvious APIs. Furthermore, using URIs provides extensibility for any transport media or content types. Finally, appending query strings to URIs confers the ability to pass variables into a dynamically created resource. The basic format is as follows:

```
[scheme]://[host]:[port]/[path]?[query string]
```

Table B.1 below lists how the [protocollInfo](#) definitions from the ConnectionManager specification relate to valid URLs. Annex A provides a more detailed explanation per protocol.

Table B.1 — Allowed AVTransportURIs

| ProtocollInfo | URI Scheme | Reference |
|--|-----------------------|-----------|
| http-get | http | A.1 |
| rtsp-rtp-udp | rtsp | A.2 |
| internal | file | A.3 |
| iec61883 | <i>Vendor-defined</i> | A.4 |
| registered ICANN domain name of vendor | <i>Vendor-defined</i> | A.5 |

B.5 AVTransport Abstraction

Via the [SetAVTransportURI\(\)](#) action an AVTransport service instance is *bound* to a content resource. Content resources are exposed by the ContentDirectory service. A content resource can represent a single atomic piece of content (for example, a single song), or a collection of contents (for example, a CD disc or playlist). Some content resources can have additional associated (secondary) resources that have to be played in conjunction; these can be bound to the same AVTransport service instance during the same invocation of the [SetAVTransportURI\(\)](#) action using the [CurrentURIMetaData](#) argument. The metadata of these associated resources is exposed by the ContentDirectory service. The *types* of content resources that can be sent or received by a device are exposed by the [GetProtocollInfo\(\)](#) action of the device's ConnectionManager service.

Once a content resource is bound to an AVTransport instance, the instance maps the resource to a flat sequence of tracks. This sequence can then be navigated via actions [Seek\(\)](#), [Next\(\)](#) and [Previous\(\)](#). For example, a resource pointing to a single audio song is mapped to 1 track, while a resource pointing to some audio playlist format is mapped to a sequence of tracks where each playlist entry maps to a single track. In case of embedded playlists, entries are mapped to tracks using a *depth-first* traversal. Playlist entries that cannot be handled by the AVTransport (for example, unknown audio formats, etc.), shall be skipped by the AVTransport. Whether those entries are eliminated immediately (not included in the number of tracks) or immediately before playback, is device-dependent.

In addition, an AVTransport might provide other means of navigation, such as time-based seeking.

The AVTransport abstracts and minimizes the differences between various specific transport media such as tapes, discs and solid-state media. Table B.2 below gives an overview on how generic AVTransport concepts such as track and [Next\(\)](#) and [Previous\(\)](#) actions apply to certain specific types of [AVTransportURIs](#). The precise mapping is implementation-dependent.

Table B.2 — Example mappings of resources type to track sequences

| <u>AVTransportURI</u> | Track concept | Number of Tracks | Current track duration | <u>Next()/Previous()</u> actions (in normal PlayMode) |
|--|------------------------------------|--|---|--|
| Audio CD | 1 track | all tracks on the CD | duration of track | Next or previous track on the CD. |
| Audio CD Changer | 1 track | all tracks of all CDs in the CD changer combined | duration of track | Next or previous track on the CD, also transition to previous or next CD in the changer if current track is the first or last one on the current disc. |
| Audio Playlist (HDD/SolidState-based player) | 1 entry in a playlist | all entries of the playlist, including entries of embedded playlists | duration of the playlist entry | Next or previous entry in the playlist; in case of embedded playlists, navigate using a <i>depth-first</i> traversal. |
| Video DVD-Volume | 1 chapter | all chapters on the DVD-Volume | duration of chapter | Next or previous chapter on the DVD. |
| Video DVD Changer | 1 volume | all volumes of all DVD discs in the DVD changer combined | duration of volume | Next or previous volume on the DVD, also transition to previous or next DVD in the changer if current volume is the first or last one on the current disc. |
| VCR (Tape) | all content on the tape | 1 | tape-length, or 0 if the tape-length is unknown | No effect. |
| List-based Tuner | 1 video channel or 1 radio station | all video channels or radio stations of the tuner channel list | 0 | Next or previous video channel or radio station in the list.; in case of major channels containing minor channels, use a <i>depth-first</i> traversal. |
| Frequency-based Tuner | 1 frequency | number of selectable frequencies | 0 | Increment or decrement frequency by device-dependent amount. |
| PVR – Tuner subsystem | 1 video channel | all live video channels of the PVR | 0 | Next of previous video channel in the list. In case of major channels containing minor channels, use a <i>depth-first</i> traversal. |
| PVR –Collection of Stored programs | 1 program | all programs of the PVR-store | duration of the program | Next or previous program in the collection. |
| PVR – Single Stored program | 1 program | 1 | duration of the program | No effect. |
| EPF | 1 image | all files of the slide show | display time of the slide in the slide show | Next of previous slide in a slide show. |

The type of resource (audio, video, image, etc.) and storage media typically affect the way the resource can be searched (seek modes), trick/play modes, and whether pausing is possible. Table B.3 below gives examples for a number of resource types.

Table B.3 — Example seek modes, play modes and transport actions, per resource type

| <u>AVTransportURI</u> | Applicable (not required) Seek modes | Applicable (not required) Play modes | Pausing possible |
|---|---|--|---------------------|
| Audio CD | TRACK_NR | NORMAL, SHUFFLE, REPEAT_ONE, REPEAT_ALL, RANDOM, DIRECT_1, INTRO | yes |
| Audio CD Changer | TRACK_NR | NORMAL, SHUFFLE, REPEAT_ONE, REPEAT_ALL, RANDOM, DIRECT_1, INTRO | yes |
| Audio Playlist (HDD/SolidState- based player) | TRACK_NR | NORMAL, SHUFFLE, REPEAT_ONE, REPEAT_ALL, RANDOM, DIRECT_1, INTRO | yes |
| Video DVD-Volume | TRACK_NR, FRAME | NORMAL | yes |
| Video DVD Changer | TRACK_NR, FRAME | NORMAL | yes |
| VCR (Tape) | TRACK_NR, ABS_TIME, REL_TIME, ABS_COUNT, REL_COUNT, TAPE_INDEX, FRAME | NORMAL | yes |
| List-based Tuner | TRACK_NR | NORMAL | no |
| Frequency-based Tuner | TRACK_NR, CHANNEL_FREQ | NORMAL | no |
| PVR – Tuner subsystem | TRACK_NR, ABS_TIME, REL_TIME, FRAME | NORMAL | yes |
| PVR – Collection of Stored programs | TRACK_NR, ABS_TIME, REL_TIME, FRAME | NORMAL | yes |
| PVR – Stored program | TRACK_NR, ABS_TIME, REL_TIME, FRAME | NORMAL | yes |
| EPF | TRACK_NR, ABS_TIME | NORMAL, SHUFFLE, REPEAT_ONE, REPEAT_ALL, RANDOM, DIRECT_1 | yes |

B.6 Supporting Multiple Virtual Transports

The number of service instances in a device shall be static, as required by the UPnP Device Architecture [14]. In certain cases it is desirable for devices to offer a dynamic number of *virtual* service instances. A control point needs to be able to control and receive events from each virtual instance individually.

In the AVTransport service case, some devices will be able to serve content to a number of clients simultaneously. For these MediaServer devices, the actual number of clients, typically renderer or recording/dubbing devices, can be fairly large, and not statically known. This service is applicable for these types of devices, as well as traditional – more static – types of devices.

A generic strategy to achieve this is as follows:

- a) Have a single static UPnP service instance in a device (hence, a single UPnP [serviceId](#)).
 - In the AVTransport case, a MediaServer device or MediaRenderer device can have a single UPnP AVTransport instance.
- b) Define the notion of a *virtual instance identifier* (this is *not* the UPnP [serviceId](#)).
 - In the AVTransport case, a [ui4](#) value.
- c) Add to all actions of the service definition an input argument that holds the virtual instance identifier to which the action applies.
 - All actions in the AVTransport service, such as [Play\(\)](#), [Stop\(\)](#), [Pause\(\)](#), have as first input argument of type [ui4](#) that identifies the instance ([InstanceID](#)).

- d) Add an evented state variable (in this case *LastChange*) to the service that holds both the instance identifier and the name and value of the latest state change of this instance. All other variables are not directly evented.
- e) Define a *factory* method, in the same service or in a related service that a control point can call to obtain a new instance identifier. This factory method shall return an error when no instances are available anymore.
 - For AVTransport, *ConnectionManager::PrepareForConnection()* serves as the factory method for obtaining a new *InstanceID* for AVTransport. In case the factory method is not present, reserved *InstanceID* value 0 can be used.
- f) Optionally, define a *cleanup* method via which a control point can indicate that the obtained instance identifier will no longer be used, and can be reused for allocation to other control points. To accommodate the situation where a control point leaves the UPnP network before calling the *cleanup* action, there needs to be an automatic cleanup mechanism implemented by the device as well. This mechanism will be device- or even vendor-specific, and needs to be described in the device template.
 - For AVTransport, *ConnectionManager::ConnectionComplete()* serves as the factory method for releasing an *InstanceID*.
- g) Optionally, define an action (and associated state variable) to retrieve the list of *currently active/allocated* instance identifiers. This is useful for control points that enter a new network and want to discover what services are currently available. It also enables control points to manually *cleanup* the whole network.
 - For AVTransport, *ConnectionManager::GetCurrentConnectionIDs()* and *ConnectionManager::GetCurrentConnectionInfo()* are used for this purpose.

The *factory* action, the *cleanup* action and the action to retrieve the *currently active/allocated* instances should normally be grouped in the same service.

To make a service that has been parameterized by instance identifiers also usable in a context (device) where no factory and cleanup actions are applicable, one or more fixed instance identifier values can be defined that a control point can directly pass in to the service actions. In the AVTransport case, special instance identifier value 0 has been defined for this purpose. The device template using such a static mechanism shall describe the semantics of that single virtual instance. For example, a vendor-specific device type that does not implement a MediaServer device but only an AVTransport service can be controlled by a control point via, for example, `AVTransport::Play(0, 1)`, `AVTransport::Stop(0)`, etc.

B.7 Playlist Playback

An important use of this service will be to control playback of an (audio) playlist. In this case, the URI of the playlist file shall be bound to the AVTransport instance via *SetAVTransportURI()*. The playlist itself only needs to be processed by the AVTransport implementation on the renderer device, the control point itself does not need to understand or parse the playlist file. Song after song of the playlist can be played without any operation needed by the control point. For example, the control point can power down, control some other devices or leave the house, without affecting the playlist playback.

When a control point has a display and wants to show meta-data of the currently playing resource it can:

- subscribe to AVTransport events so it always knows the current transport state and track information
- use *CurrentTrackURI* to obtain meta-data of the currently playing track via the *ContentDirectory::Search()* action of the ContentDirectory service
- use *CurrentTrackMetaData* to obtain any meta-data of the currently playing track from the AVTransport service directly

Whether an AVTransport implementation can deal with relative URLs that might be present inside a playlist file is device-dependent.

B.8 Dynamic Playlists

Processing of the dynamic playlists is similar to the media renderer fetching the playlist directly from a ContentDirectory service. Note, however, delivery of playlist metadata does not imply a permanent change of transport state, i.e. an explicit transport command such as [Play\(\)](#) is expected to commence playback.

B.8.1 Playlist Updating

When playlists are submitted to a renderer device, any ContentDirectory service metadata in the submitted playlist is expected to be up to date. In particular, absolute host addresses exposed in playlist metadata are expected to reflect current addresses for home network servers. For non-home network servers, host names resolved to absolute addresses need to be refreshed from the appropriate Domain Name Server (DNS) as needed.

The ContentDirectory service “Tracking Changes Option” can be used (if available) to determine if ContentDirectory service items included in a submitted playlist have changed.

Alternatively, stored instructions can be used to completely rebuild the playlist by obtaining the ContentDirectory service items prior to sending the playlist to the rendering device or an existing playlist can be updated by comparing its contents with current copies of item metadata from the ContentDirectory service.

B.8.2 Determining Playlist Rendering Capabilities

A control point can determine the types of playlists a transport instance can accept as well as state information about any currently executing playlists by issuing the [GetPlaylistInfo\(\)](#) action.

A control point requests information for both streaming and static playlists in the following examples:

Request:

```
GetPlaylistInfo(0, "Streaming")
```

Response:

```
GetPlaylistInfo(
"<?xml version="1.0" encoding="UTF-8"?>
<playlistInfo
  xmlns="urn:schemas-upnp-org:av:rpl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
urn:schemas-upnp-org:av:rpl
http://www.upnp.org/schemas/av/rpl.xsd">
  <streamingPlaylistInfo>
    <playlistState>Idle</playlistState>
    <playlistChunkLengthMax>10240</playlistChunkLengthMax>
    <playlistDataLengthUsed>0</playlistDataLengthUsed>
    <playlistTotalLengthAvail>32768</playlistTotalLengthAvail>
    <playlistTrackMin>0</playlistTrackMin>
    <playlistTrackMax>0</playlistTrackMax>
    <playlistCurrentFormat />
    <playlistAllowedFormats>
      <contentType MIMETYPE="audio/m3u" />
    </playlistAllowedFormats>
  </streamingPlaylistInfo>
</playlistInfo>")
```

Request:

```
GetPlaylistInfo(0, "Static")
```

Response:

```
GetPlaylistInfo(
"<?xml version="1.0" encoding="UTF-8"?>
<playlistInfo
  xmlns="urn:schemas-upnp-org:av:rpl"
```



```

xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
urn:schemas-upnp-org:av:rpl
http://www.upnp.org/schemas/av/rpl.xsd">
<staticPlaylistInfo>
  <playlistState>Idle</playlistState>
  <playlistChunkLengthMax>10240</playlistChunkLengthMax>
  <playlistTotalLengthMax>1048576</playlistTotalLengthMax>
  <playlistCurrentFormat />
  <playlistAllowedFormats>
    <contentType MIMEType="text/xml" extendedType="*" />
    <contentType MIMEType="text/xml" extendedType="DLNA.ORG_PN=DIDL_V" />
  </playlistAllowedFormats>
</staticPlaylistInfo>
</playlistInfo>")

```

B.8.3 Submitting a Streaming Playlist

A control point uses the [SetStreamingPlaylist\(\)](#) action to deliver metadata directly to a device. This action enables the control point to continue to submit playlist data to the rendering device after playback has commenced.

The initial invocation of the [SetStreamingPlaylist\(\)](#) action specifies a [PlaylistStep](#) argument of [Initial](#). This will cause the device to discard any incomplete playlist delivery in progress. Subsequent playlist chunks are delivered with the [PlaylistStep](#) argument set to [Continue](#). If no further playlist chunks are expected to be delivered then a final invocation of [SetStreamingPlaylist\(\)](#) with a [PlaylistStep](#) value of [Stop](#) is issued.

The following is an example of the delivery of the initial part of a streaming playlist in m3u format:

```

Request:
SetStreamingPlaylist(0,
"#EXTM3U
#EXTINF:123,Mother Love Bone - Chloe Dancer
http://10.0.0.1/getcontent.asp?id=6
#EXTINF:321,Smashing Pumpkins - Drown
http://10.0.0.1/getcontent.asp?id=8
#EXTINF:220,Pearl Jam - State Of Love And Trust
http://10.0.0.1/getcontent.asp?id=7
",
253,
"audio/m3u",
"*",
"Initial")

```

Note: [PlaylistData](#) includes CRLF line-endings.

```

Response:
SetStreamingPlaylist()

```

The control point can issue a [Play\(\)](#) action to start playback on the rendering device.

```

Request:
Play()

```

```

Response:
Play()

```

The control point can submit additional playlist fragment(s) to the renderer during playback as shown in the following example. In this example, the fragment submitted is anticipated to be the last chunk so a [PlaylistStep](#) value of [Stop](#) is specified:

```

Request:
SetStreamingPlaylist(0,
"#EXTINF:120,Smashing Pumpkins - The Beginning is the End

```

```

http://10.0.0.1/getcontent.asp?id=9
",
95,
"audio/m3u",
"**,
"Stop")

```

Note: [PlaylistData](#) includes CRLF line-endings.

Response:
SetStreamingPlaylist()

B.8.4 Modifying a Current Playlist

A control point can use the [GetPlaylistInfo\(\)](#) action with the [PlaylistType](#) argument set to "[StreamingPlContents](#)" to retrieve the current streaming playlist. The control point can then update the retrieved playlist and submit it using the [SetStreamingPlaylist\(\)](#) action with the [PlaylistStep](#) argument set to "[Replace](#)". In the following example, the MediaRenderer is playing the second streaming playlist entry. The control point retrieves the current playlist, inserts a new entry and then replaces the MediaRenderer playlist.

Request:
GetPlaylistInfo(0, "StreamingPlContents")

Response:
GetPlaylistInfo(
"<?xml version="1.0" encoding="UTF-8" ?>
<playlistInfo
 xmlns="urn:schemas-upnp-org:av:rpl"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
urn:schemas-upnp-org:av:rpl
http://www.upnp.org/schemas/av/rpl.xsd">
<streamingPlaylistInfo>
 <playlistState>Idle</playlistState>
 <playlistChunkLengthMax>10240</playlistChunkLengthMax>
 <playlistDataLengthUsed>0</playlistDataLengthUsed>
 <playlistTotalLengthAvail>32768</playlistTotalLengthAvail>
 <playlistTrackMin>1</playlistTrackMin>
 <playlistTrackMax>4</playlistTrackMax>
 <playlistCurrentFormat>
 <contentType MIMETYPE="audio/m3u" />
 </playlistCurrentFormat>
 <playlistAllowedFormats>
 <contentType MIMETYPE="audio/m3u" />
 </playlistAllowedFormats>
 <playlistContents currentTrack="2">
 #EXTM3U
 #EXTINF:123,Mother Love Bone - Chloe Dancer
 http://10.0.0.1/getcontent.asp?id=6
 #EXTINF:321,Smashing Pumpkins - Drown
 http://10.0.0.1/getcontent.asp?id=8
 #EXTINF:220,Pearl Jam - State Of Love And Trust
 http://10.0.0.1/getcontent.asp?id=7
 #EXTINF:120,Smashing Pumpkins - The Beginning is the End
 http://10.0.0.1/getcontent.asp?id=9
 </playlistContents>
</streamingPlaylistInfo>
</playlistInfo">
)

Request:
SetStreamingPlaylist(0,
"#EXTINF:444,Adagio for Strings
http://10.0.0.1/getcontent.asp?id=30
#EXTINF:220,Pearl Jam - State Of Love And Trust
http://10.0.0.1/getcontent.asp?id=7
#EXTINF:120,Smashing Pumpkins - The Beginning is the End

```
http://10.0.0.1/getcontent.asp?id=9
",
251,
"audio/m3u",
"\"",
"Replace")
```

Note: [PlaylistData](#) includes CRLF line-endings.

Response :
SetStreamingPlaylist()

B.8.5 Submitting a Static Playlist

A control point can use the [SetStaticPlaylist\(\)](#) action to deliver metadata directly to a device. This action is used for playlist formats that need to be completely delivered to the rendering device prior to commencing playback.

For large playlists, the [SetStaticPlaylist\(\)](#) action enables the delivery of a playlist using multiple invocations of the action. Each invocation delivers a chunk of the complete playlist. The size of each chunk of the playlist is indicated by the [PlaylistDataLength](#) argument. The [PlaylistOffset](#) argument provides the position of the chunk within the complete playlist. The [PlaylistTotalLength](#) argument provides the total length of the complete playlist. It is expected that the complete playlist will be submitted prior to a playback operation being requested.

For playlists using object linking metadata, the arguments [PlaylistStartObj](#) and [PlaylistStartGroup](#) will be non-null indicating the starting object and group for the object linked playlist.

The following is an example of the delivery of a static playlist containing photo items:

```
SetStaticPlaylist(0,
"<DIDL-Lite
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
http://www.upnp.org/schemas/av/didl-lite.xsd
urn:schemas-upnp-org:metadata-1-0/upnp/
http://www.upnp.org/schemas/av/upnp.xsd">
<item id="14" parentID="12" restricted="0">
<dc:title>Sunset on the beach</dc:title>
<dc:date>2001-10-20</dc:date>
<upnp:class>object.item.imageItem.photo</upnp:class>
<res protocolInfo="http-get:*:image/jpeg:*" size="20000">
http://10.0.0.1/getcontent.asp?id=14
</res>
</item>
<item id="15" parentID="12" restricted="0">
<dc:title>Playing in the pool</dc:title>
<dc:date>2001-10-25</dc:date>
<upnp:class>object.item.imageItem.photo</upnp:class>
<res protocolInfo="http-get:*:image/jpeg:*" size="25000">
http://10.0.0.1/getcontent.asp?id=15
</res>
</item>
<item id="20" parentID="13" restricted="0">
<dc:title>Playing in the pool</dc:title>
<dc:date>2001-10-25</dc:date>
<upnp:class>object.item.imageItem.photo</upnp:class>
<res protocolInfo="http-get:*:image/jpeg:*" size="25000">
http://10.0.0.1/getcontent.asp?id=15
</res>
</item>
</DIDL-Lite">
```

```

1736,
0,
1736,
"text/xml",
"*",
" ",
" " )

```

Note: In the above example, the [PlaylistDataLength](#) and [PlaylistTotalLength](#) arguments reflect the XML-escaped size of the playlist data. The [PlaylistData](#) argument value is shown in non-XML-escaped format for clarity.

The control point can issue a [Play\(\)](#) action to start playback on the rendering device.

Request:

```
Play()
```

Response:

```
Play()
```

B.8.6 Retrieving a Current Static Playlist

A control point can use the [GetPlaylistInfo\(\)](#) action with the [PlaylistType](#) argument set to "[StaticP1Contents](#)" to retrieve the current static playlist contents and current object position.

Request:

```
GetPlaylistInfo(0, "StaticP1Contents")
```

Response:

```

GetPlaylistInfo(
"<?xml version="1.0" encoding="UTF-8"?>
<playlistInfo
  xmlns="urn:schemas-upnp-org:av:rpl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
urn:schemas-upnp-org:av:rpl
http://www.upnp.org/schemas/av/rpl.xsd">
  <staticPlaylistInfo>
    <playlistState>Idle</playlistState>
    <playlistChunkLengthMax>10240</playlistChunkLengthMax>
    <playlistTotalLengthMax>1048576</playlistTotalLengthMax>
    <playlistCurrentFormat>
      <contentType mimeType="text/xml" extendedType="DLNA.ORG_PN=DIDL_V" />
    </playlistCurrentFormat>
    <playlistAllowedFormats>
      <contentType mimeType="text/xml" extendedType="*" />
      <contentType mimeType="text/xml" extendedType="DLNA.ORG_PN=DIDL_V" />
    </playlistAllowedFormats>
    <playlistContents currentObjID="20">
&lt;DIDL-Lite
  xmlns:dc="http://purl.org/dc/elements/1.1/"&gt;
  xmlns="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:upnp="http://www.upnp.org/schemas/av/didl-lite.xsd"
  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"
  &lt;item id="14" parentID="12"
    restricted="0"&gt;
    &lt;dc:title>Sunset on the beach&lt;/dc:title&gt;
    &lt;dc:date>2001-10-20&lt;/dc:date&gt;
      &lt;upnp:class>
        object.item.imageItem.photo
      &lt;/upnp:class>
    &lt;res protocolInfo="http-get:*:image/jpeg:*"&gt;

```

```

        size="20000">
        http://10.0.0.1/getcontent.asp?id=14
    </res>
</item>
<item id="15" parentID="12"
    restricted="0"
    <dc:title>Playing in the pool</dc:title>
    <dc:date>2001-10-25</dc:date>
    <upnp:class>
        object.item.imageItem.photo
    </upnp:class>
    <res protocolInfo="http-get:*:image/jpeg:*"
        size="25000">
        http://10.0.0.1/getcontent.asp?id=15
    </res>
</item>
<item id="20" parentID="13"
    restricted="0"
    <dc:title>Playing in the pool</dc:title>
    <dc:date>2001-10-25</dc:date>
    <upnp:class>
        object.item.imageItem.photo
    </upnp:class>
    <res protocolInfo="http-get:*:image/jpeg:*"
        size="25000">
        http://10.0.0.1/getcontent.asp?id=15
    </res>
</item>
</DIDL-Lite>
</playlistContents>
</staticPlaylistInfo>
</playlistInfo>")

```

B.9 CLOCKS SYNC feature: Synchronized Playback

The examples in Annex B.9 demonstrate how to configure and manage synchronized playback of two independent pieces of matched content. In these examples, a piece of video content (without any audio) is sent to a display device and a matching piece of audio content is sent to a separate network speaker. Both content streams are sent using a push protocol such that each connection's AVTransport service is provided by the server device. Once the [SetAVTransportURI\(\)](#) actions have been invoked with the URI of the video content and the URI of the audio content, the example begins.

The *CLOCKS SYNC feature* enables synchronized playback. The main functionality of this service is control over the [TransportState](#) variable. A state machine depicting the relations between AVTransport actions and [TransportState](#) values is shown below. In case of any contradictions with the text in the descriptions of the individual actions, the text shall be considered normative.

Here is a glossary of terms used in the context of *CLOCKS SYNC feature*:

- **Clock Synchronization Protocol:** A method that enables UPnP Devices and Control Points to agree on a common clock. That clock is referred to as the Master Clock. (IEEE 802.1AS is an example of a Clock Synchronization Protocol.)
- **Master Clock:** The common clock established using a certain Clock Synchronization Protocol.
- **Media Clock:** The clock associated with a media elementary stream. Typically, each audio and video frame has a presentation time derived from the Media Clock. The presentation time can be signaled explicitly, e.g., using the RTP Timestamp field, or it can be inferred by properties of the stream, such as the video frame rate, or number of audio samples contained in an audio frame.

- **Time Stamp Mechanism:** A method that establishes the relationship between the frequencies of the Media Clock and the Master Clock. A Time Stamp Mechanism can use a Time Stamp Protocol, or it can use a static pre-defined mapping.
- **Time Stamp Protocol:** A Time Stamp Mechanism that does not use a pre-defined mapping between the Media Clock and the Master Clock. (IEEE-P1733 is an example of a Time Stamp Protocol.)

The clock synchronization features enable a control point to initiate synchronized playback using a Clock Synchronization Protocol and a Time Stamp Mechanism supported by the MediaRenderer and MediaServer devices.

This specification defines two Time Stamp Mechanisms: RTP+IEEE-1733 and Identity. Additional Time Stamp Mechanisms can be defined by vendors.

B.9.1.1 Time Stamp Mechanisms

RTP+IEEE-1733: This is a Time Stamp Protocol defined only for use with RTP. When RTP+IEEE-1733 is used, the RTP sampling clock used for each individual RTP stream is a “Media Clock”. RTP+IEEE-1733 provides a mapping between the Media Clock of each RTP stream and the Master Clock.

Identity: This is a pre-defined mapping that states that the frequencies of the Media Clock and the Master Clock are identical. This is the default Time Stamp Mechanism and it can be used with both HTTP and RTP. When Identity is used with RTP, “Media Clock” refers to the RTP wallclock [41].

The Identity Time Stamp Mechanism is not compatible with RTP+IEEE-1733. The Identity mechanism assumes that the Media Clock and Master Clock run at identical frequencies. The RTP+IEEE-1733 mechanism makes no such assumption, and enables MediaRenderer devices to estimate the frequency difference between Media Clock and Master Clock.

B.9.2 Example of *SyncOffset* State Variable

For example, assume a given sound is scheduled to be played on two different, independent loud speakers (renderers) at precisely 5:00pm on July 6, 2009. When 5:00pm occurs, both speakers play the same sound at precisely the same time, but if one of the speakers is sitting next to the listener and the other speaker is 50 yards away, then the listener will hear an echo effect since it takes a noticeable amount of time (say 150 milliseconds) for the sound from the distant speaker to reach the listener. In order for the sound from both speakers to reach the listener at the same time, the two renderers could be given two different presentation times such as 5:00:00.000pm and 4:59:59.850pm, respectively. Alternatively, rather than using two different presentation times, the *SyncOffset* state variable on the distant speaker can be set to a value of -P00:00:00.150. This value causes that renderer to play the sound (and all subsequent sounds) 150 milliseconds sooner than the nearby renderer. Consequently, the listener does not hear any echo effect (but, of course, only when standing at the listener’s current location).

B.9.3 Example Scenarios for *SyncPlay()* Action

Playlists submitted via the *SetStreamingPlaylist()* or *SetStaticPlaylist()* actions that are based on ContentDirectory service items can reflect the current metadata available. In particular, absolute host addresses contained in ContentDirectory service *res* property values can reflect current IP addresses on the home network. In addition, submitted playlists can be checked for ContentDirectory service items that have been removed or modified. For playlist formats that are not XML based, submitted playlists can insure that any absolute host addresses are updated in submitted playlists.

In Annex B.9.3, we will present three examples of *SyncPlay()* actions with various *PresentationTime* and *ReferencePosition* values to illustrate expected Renderer behavior in each case. In all three examples the *ReferencePosition* is expressed in terms of absolute time as indicated by *ReferencePositionUnits* input argument having the value of “*ABS_TIME*”.

B.9.3.1 SyncPlay() with past presentation time

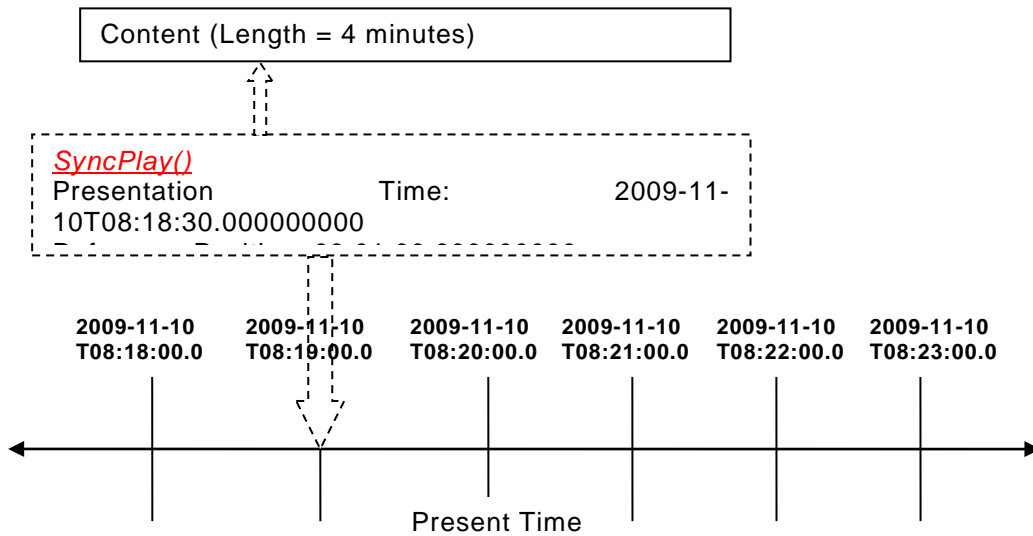


Figure B.2 — SyncPlay() with past presentation time

In this first example diagram, the SyncPlay() action is received with presentation time of one and a half a minute ago and reference position of 1 minute into the length of the content. In this case, the renderer received a command that would be started in the past. The information received, e.g. the starting time and offset in the file translates to a current offset of 2.5 minutes into the length of the content. In this case, the renderer starts rendering the content immediately starting at 2.5 minutes into the content.

Request:

```
SyncPlay(
  "0",
  "1",
  "ABS_TIME",
  "00:01:00.000000000",
  "2009-11-10T08:18:30.000000000",
  "Local Device"
)
```

Response:

```
SyncPlay("")
```

B.9.3.2 **SyncPlay()** with future presentation time

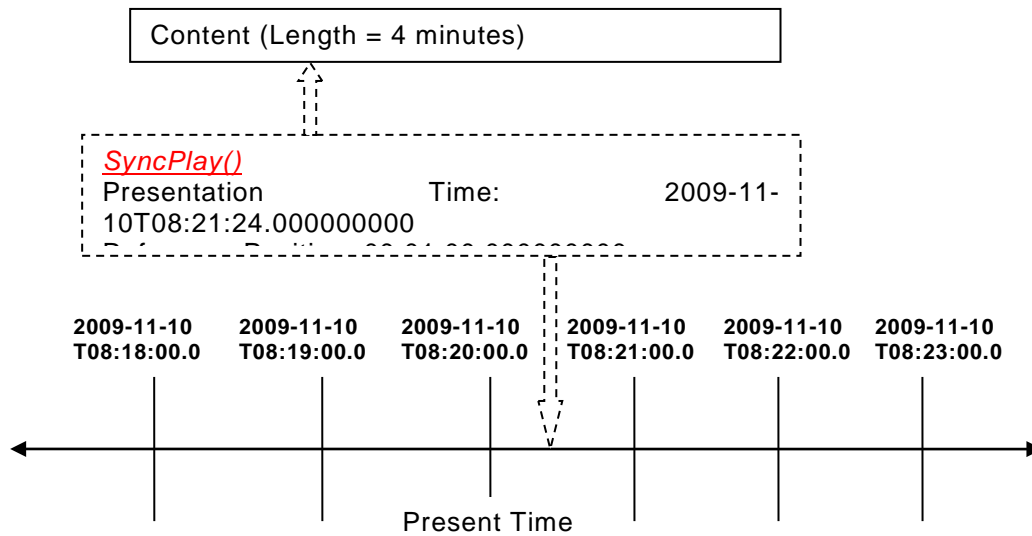


Figure B.3 — **SyncPlay()** with future presentation time

In case of example 2, the SyncPlay() action is received with presentation time that points to an instance in the future (24 seconds from now) and the reference position is 1 minute into the content. In this case, the renderer starts rendering the content at the requested presentation time (24 seconds from receiving the request) starting at 1 minute into the content. During the 24 second duration, the renderer goes into "TRANSITIONING" state.

Request :

```
SyncPlay(
  "0",
  "1",
  "ABS_TIME",
  "00:01:00.000000000",
  "2009-11-10T08:21:24.000000000",
  "Local Device"
)
```

Response :

```
SyncPlay("")
```


B.9.3.3 SyncPlay() with future presentation time and reference positioning

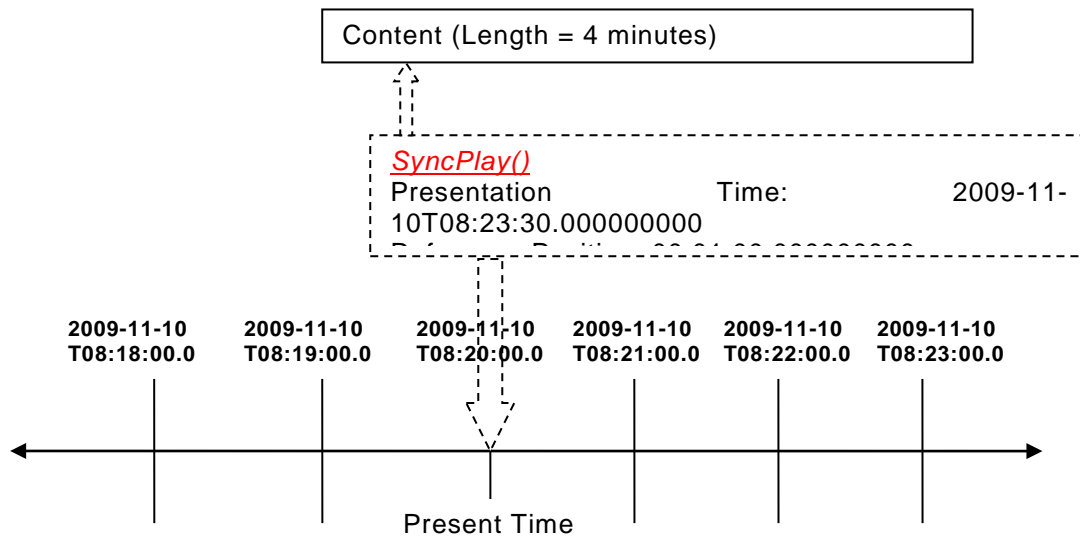


Figure B.4 — SyncPlay() with future presentation time and reference positioning

In case of example 3, the SyncPlay() action is received with presentation time that points to an instance in the future (say 3.5 minutes) and the reference position is 1 minute into the content. In this case, the renderer starts rendering the content at the requested presentation time (3.5 minutes from receiving the request) starting at 1 minute into the content. During the 3.5 minute duration, the renderer goes into "TRANSITIONING" state.

```

Request :
SyncPlay(
  "0",
  "1",
  "ABS_TIME",
  "00:01:00.000000000",
  "2009-11-10T08:23:30.000000000",
  "Local Device"
)
    
```

```

Response :
SyncPlay("")
    
```

Annex C (informative)

Bibliography

The following documents, in whole or in part, may be useful for understanding this document but they are not essential 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.

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

[43] – *AVArchitecture:2*, UPnP Forum, March 31, 2013. Available at: <http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v2-20130331.pdf>. Latest version available at: <http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v2.pdf>.

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

[45] – *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>.

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

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

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

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

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

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

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

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

- [54] – *IETF RFC 1321, The MD5 Message-Digest Algorithm*, R. Rivest, April 1992.
Available at: <http://tools.ietf.org/html/rfc1321>.
- [55] – *IETF RFC 2030, Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OS*, D Mills, October 1996.
Available at: <http://www.ietf.org/rfc/rfc2030.txt>.
- [56] – *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>.
- [57] – *IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels*, S. Bradner, 1997.
Available at: <http://www.faqs.org/rfcs/rfc2119.html>.
- [58] – *IETF RFC 3174, US Secure Hash Algorithm 1 (SHA1)*, D. Eastlake et al, September 2001.
Available at: <http://tools.ietf.org/html/rfc3174>.
- [59] – *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>.
- [60] – *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>.
- [61] – *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>.
- [62] – *Unicode Technical Standard #10, Unicode Collation Algorithm version 4.1.0*, M. Davis, K. Whistler, May 5, 2005.
Available at: <http://www.unicode.org/reports/tr10/tr10-14.html>.
- [63] – *Unicode Technical Standard #10, Unicode Collation Algorithm, version 4.1.0, revision 14*, M. Davis, K. Whistler, May 5, 2005.
Available at: <http://www.unicode.org/reports/tr10/tr10-14.html>.
- [64] – *Unicode Technical Standard #35, Locale Data Markup Language, version 1.3R1, revision 5*, M. Davis, June 2, 2005.
Available at: <http://www.unicode.org/reports/tr35/tr35-5.html>.
- [65] – *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>.
- [66] – *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>.
- [67] – *XML Path Language (XPath) 2.0*. Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernandez, Michael Kay, Jonathan Robie, Jerome Simeon. W3C Recommendation, 21 November 2006.
Available at: <http://www.w3.org/TR/xpath20>.
- [68] – *XQuery 1.0 An XML Query Language*. W3C Recommendation, 23 January 2007.
Available at: <http://www.w3.org/TR/2007/REC-xquery-20070123>.