INTERNATIONAL STANDARD

ISO/IEC 29341-20-10

First edition
2017-09

Information technology — UPnP Device Architecture —

Part 20-10: Audio video device control protocol — Level 4 — Audio video transport service

Technologies de l'information — Architecture de dispositif UPnP —

Partie 20-10: Protocole de contrôle de dispositif audio-vidéo — Niveau 4 — Service de transport audio-vidéo

Reference number
ISO/IEC 29341-20-10:2017(E)

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

CONTENTS

List of Tables

List of Figures

Foreword

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

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

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

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

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

ISO/IEC 29341-20-10 was prepared by UPnP Forum and adopted, under the PAS procedure, by joint technical committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

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

Introduction

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

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

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

Information may be obtained from:

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

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

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

Information may be obtained from:

Microsoft Corporation
One Microsoft Way
USA – Redmond WA 98052

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

Original UPnP Document

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

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

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

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

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

1 Scope

This 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.

[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
AV Working Committee defined namespaces			
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
Externally defined namespaces			
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
AV Working Committee Defined Namespaces			
atrs	AllowedTransformSettings-v*n-yyyymmdd*.xsd AllowedTransformSettings-v*n*.xsd AllowedTransformSettings.xsd	<TransformList>	[1]
av	av-v*n-yyyymmdd*.xsd av-v*n*.xsd av.xsd	*n/a*	[3]
avdt	avdt-v*n-yyyymmdd*.xsd avdt-v*n*.xsd avdt.xsd	<AVDT>	[2]
avs	avs-v*n-yyyymmdd*.xsd avs-v*n*.xsd avs.xsd	<Capabilities> <Features> <stateVariableValuePairs>	[4]
avt-event	avt-event-v*n-yyyymmdd*.xsd avt-event-v*n*.xsd avt-event.xsd	<Event>	[6]
cds-event	cds-event-v*n-yyyymmdd*.xsd cds-event-v*n*.xsd cds-event.xsd	<StateEvent>	[8]
cm-dciu	cm-deviceClockInfoUpdates-v*n-yyyymmdd*.xsd cm-deviceClockInfoUpdates-v*n*.xsd cm-deviceClockInfoUpdates.xsd	<DeviceClockInfoUpdates>	[10]
cm-ftrlst	cm-featureList-v*n-yyyymmdd*.xsd cm-featureList-v*n*.xsd cm-featureList.xsd	<Features>	[11]
didl-lite	didl-lite-v*n-yyyymmdd*.xsd didl-lite-v*n*.xsd didl-lite.xsd	<DIDL-Lite>	[15]
dmo	dmo-v*n-yyyymmdd*.xsd dmo-v*n*.xsd dmo.xsd	<DeviceMode>	[16]
dmor	dmor-v*n-yyyymmdd*.xsd dmor-v*n*.xsd dmor.xsd	<DeviceModeRequest>	[17]
dmos	dmos-v*n-yyyymmdd*.xsd dmos-v*n*.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-v*n*-*yyyymmdd*.xsd pi-v*n*.xsd pi.xsd	<PermissionsInfo>	[20]
rcs-event	rcs-event-v*n*-*yyyymmdd*.xsd rcs-event-v*n*.xsd rcs-event.xsd	<Event>	[22]
rii	rii-*vn-yyyymmdd*.xsd rii-v*n*.xsd rii.xsd	<rendererInfo>	[23]
rpl	rpl-*vn-yyyymmdd*.xsd rpl-v*n*.xsd rpl.xsd	<PlaylistInfo>	[24]
trs	TransformSettings-*vn-yyyymmdd*.xsd TransformSettings-v*n*.xsd TransformSettings.xsd	<TransformSettings>	[28]
srs	srs-v*n-yyyymmdd*.xsd srs-v*n*.xsd srs.xsd	<srs>	[26]
srs-event	srs-event-v*n*-*yyyymmdd*.xsd srs-event-v*n*.xsd srs-event.xsd	<StateEvent>	[27]
upnp	upnp-v*n-yyyymmdd*.xsd upnp-v*n*.xsd upnp.xsd	*n/a*	[29]
Externally Defined Namespaces			
dc	*Absolute URL*: http://dublincore.org/schemas/xmls/simpledc20021212.xsd		[12]
xsd	*n/a*	<schema>	[35]
xsi	*n/a*		*n/a*
xml	*n/a*		[31]
[a] Absolute URIs are generated by prefixing the relative URIs with "http://www.upnp.org/schemas/av/"			

4.3.1 Namespace Prefix Requirements

There are many occurrences in this specification of string data types that contain XML names (property names). These XML names in strings will not be processed under namespace-aware conditions. Therefore, all occurrences in instance documents of XML names in strings shall use the standard namespace prefixes as declared in Table 3. In order to properly process the XML documents described herein, control points and devices shall use namespace-aware XML processors [32] for both reading and writing. As allowed by [32], the namespace prefixes used in an instance document are at the sole discretion of the document creator. Therefore, the declared prefix for a namespace in a document may be different from the standard prefix. All devices shall be able to correctly process any valid XML instance document, even when it uses a non-standard prefix for ordinary XML names. However, it is strongly recommended that all devices use these standard prefixes for all instance documents to avoid confusion on the part of both human and machine readers. These standard prefixes are used in all descriptive text and all XML examples in this and related UPnP specifications. However, each individual specification may assume a default namespace for its descriptive text. In that case, names from that namespace may appear with no prefix.

The assumed default namespace, if any, for each UPnP AV specification is given in Table 5.

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

Table 5 — Default Namespaces for the AV Specifications

AV Specification Name	Default Namespace Prefix
AVTransport	avt-event
ConnectionManager	*n/a*
ContentDirectory	didl-lite
MediaRenderer	*n/a*
MediaServer	*n/a*
RenderingControl	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 namepace also has a set of undated schema URIs. These undated schema URIs have two distinct formats with slightly different meanings:

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

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

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

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

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

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

- All instance documents, whether generated by a service or a control point, 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
TransportState	*R*	string	See 5.2.2		
TransportStatus	*R*	string	See 5.2.3		
CurrentMediaCategory	*R*	string	See 5.2.4		
PlaybackStorageMedium	*R*	string	See 5.2.5		
RecordStorageMedium	*R*	string	See 5.2.6		
PossiblePlaybackStorageMedia	*R*	string	CSV [b] (string) See 5.2.7		
PossibleRecordStorageMedia	*R*	string	CSV (string) See 5.2.8		
CurrentPlayMode	*R*	string	See 5.2.9	"*NORMAL*"	
TransportPlaySpeed	*R*	string	See 5.2.10	"*1*"	
RecordMediumWriteStatus	*R*	string	See 5.2.11		
CurrentRecordQualityMode	*R*	string	See 5.2.12		
PossibleRecordQualityModes	*R*	string	CSV (string) See 5.2.13		
NumberOfTracks	*R*	ui4	See 5.2.14		
CurrentTrack	*R*	ui4	See 5.2.15		
CurrentTrackDuration	*R*	string	See 5.2.16		
CurrentMediaDuration	*R*	string	See 5.2.17		
CurrentTrackMetaData	*R*	string	See 5.2.18		
CurrentTrackURI	*R*	string	See 5.2.19		
AVTransportURI	*R*	string	See 5.2.20		
AVTransportURIMetaData	*R*	string	See 5.2.21		
NextAVTransportURI	*R*	string	See 5.2.22		
NextAVTransportURIMetaData	*R*	string	See 5.2.23		
RelativeTimePosition	*R*	string	See 5.2.24		
AbsoluteTimePosition	*R*	string	See 5.2.25		
RelativeCounterPosition	*R*	i4	See 5.2.26		
AbsoluteCounterPosition	*R*	ui4	See 5.2.27		
CurrentTransportActions	*CR* [c]	string	CSV (string) See 5.2.28		
LastChange	*R*	string	See 5.2.29		
DRMState	*CR* [c]	string	See 5.2.30	"*UNKNOWN*"	
SyncOffset	*CR* [c]	string	See 5.2.31		
A_ARG_TYPE_SeekMode	*R*	string	See 5.2.32		
A_ARG_TYPE_SeekTarget	*R*	string	See 5.2.33		
A_ARG_TYPE_InstanceID	*R*	ui4	See 5.2.34		
A_ARG_TYPE_DeviceUDN	*CR* [c]	string	See 5.2.35		
A_ARG_TYPE_ServiceType	*CR* [c]	string	See 5.2.36		
A_ARG_TYPE_ServiceID	*CR* [c]	string	See 5.2.37		
A_ARG_TYPE_StateVariableValuePairs	*CR* [c]	string	See 5.2.38		
A_ARG_TYPE_StateVariableList	*CR* [c]	string	CSV (string) See 5.2.39		

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

[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*"	*R*
"*PLAYING*"	*R*
"*TRANSITIONING*"	*R*
"*PAUSED_PLAYBACK*"	*CR*, required if *Pause()* action is implemented
"*PAUSED_RECORDING*"	*CR*, required if both *Pause()* and *Record()* actions are implemented
"*RECORDING*"	*CR*, required if *Record()* action is implemented
"*NO_MEDIA_PRESENT*"	*A*
Vendor-defined	*X*

5.2.3 *TransportStatus*

This required state variable is used to indicate if asynchronous errors haved 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*"	*R*
"*ERROR_OCCURRED*"	*R*
Vendor-defined	*X*

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*"	*R*
"*TRACK_AWARE*"	*R*
"*TRACK_UNAWARE*"	*R*

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

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
"*NORMAL*"	*R*
"*SHUFFLE*"	*A*
"*REPEAT_ONE*"	*A*
"*REPEAT_ALL*"	*A*
"*RANDOM*"	*A*
"*DIRECT_1*"	*A*
"*INTRO*"	*A*
Vendor-defined	*X*

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
"*WRITABLE*"	*A*
"*PROTECTED*"	*A*
"*NOT_WRITABLE*"	*A*
"*UNKNOWN*"	*A*
"*NOT_IMPLEMENTED*"	*A*
Vendor-defined	*X*

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
"*0:EP*"	*A*
"*1:LP*"	*A*
"*2:SP*"	*A*
"*0:BASIC*"	*A*
"*1:MEDIUM*"	*A*
"*2:HIGH*"	*A*
"*NOT_IMPLEMENTED*"	*A*
Vendor-defined	*X*

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 than 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 — allowedValueRangefor *SyncOffset*

	Value	R/A
minimum	>= P00:00:00	*R*
maximum	*Vendor defined*	*R*
step	*Vendor defined*	*R*

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
"*TRACK_NR*"	*R*
"*ABS_TIME*"	*A*
"*REL_TIME*"	*A*
"*ABS_COUNT*"	*A*
"*REL_COUNT*"	*A*
"*CHANNEL_FREQ*"	*A*
"*TAPE-INDEX*"	*A*
"*REL_TAPE-INDEX*"	*A*
"*FRAME*"	*A*
"*REL_FRAME*"	*A*
Vendor-defined	*X*

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 *A_ARG_TYPE_SeekMode*	Format of *A_ARG_TYPE_SeekTarget*
"*TRACK_NR*"	**ui4**
"*ABS_TIME*"	Formatted as specified in subclause 5.2.16
"*REL_TIME*"	Formatted as specified in subclause 5.2.16
"*ABS_COUNT*"	**ui4**
"*REL_COUNT*"	**i4**
"*CHANNEL_FREQ*"	**float**, expressed in Hz.
"*TAPE-INDEX*"	**ui4**
"*REL_TAPE-INDEX*"	**i4**
"*FRAME*"	**ui4**
"*REL_FRAME*"	**i4**

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.
"StaticPlContents"	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.
"StreamingPlContents"	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 "*StreamingPlContents*" (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			
CurrentRecordQualityMode	NO	NO			
NumberOfTracks	NO	NO			
CurrentTrack	NO	NO			
CurrentTrackDuration	NO	NO			

Variable Name	Evented	Moderated Event	Min Event Interval [a] (seconds)	Logical Combination	Min Delta per Event [b]
CurrentMediaDuration	*NO*	*NO*			
CurrentTrackURI	*NO*	*NO*			
CurrentTrackMetaData	*NO*	*NO*			
AVTransportURI	*NO*	*NO*			
AVTransportURIMetaData	*NO*	*NO*			
NextAVTransportURI	*NO*	*NO*			
NextAVTransportURIMetaData	*NO*	*NO*			
RelativeTimePosition	*NO*	*NO*			
AbsoluteTimePosition	*NO*	*NO*			
RelativeCounterPosition	*NO*	*NO*			
AbsoluteCounterPosition	*NO*	*NO*			
CurrentTransportActions	*NO*	*NO*			
LastChange	*YES*	*YES*	0.2		
DRMState	*NO*	*NO*			
SyncOffset	*NO*	*NO*			
A_ARG_TYPE_SeekMode	*NO*	*NO*			
A_ARG_TYPE_SeekTarget	*NO*	*NO*			
A_ARG_TYPE_InstanceID	*NO*	*NO*			
A_ARG_TYPE_DeviceUDN	*NO*	*NO*			
A_ARG_TYPE_ServiceType	*NO*	*NO*			
A_ARG_TYPE_ServiceID	*NO*	*NO*			
A_ARG_TYPE_StateVariableValuePairs	*NO*	*NO*			
A_ARG_TYPE_PlaylistData	*NO*	*NO*			
A_ARG_TYPE_PlaylistDataLength	*NO*	*NO*			
A_ARG_TYPE_PlaylistOffset	*NO*	*NO*			
A_ARG_TYPE_PlaylistTotalLength	*NO*	*NO*			
A_ARG_TYPE_PlaylistMIMEType	*NO*	*NO*			
A_ARG_TYPE_PlaylistExtendedType	*NO*	*NO*			
A_ARG_TYPE_PlaylistStep	*NO*	*NO*			
A_ARG_TYPE_PlaylistType	*NO*	*NO*			
A_ARG_TYPE_PlaylistInfo	*NO*	*NO*			
A_ARG_TYPE_PlaylistStartObjID	*NO*	*NO*			
A_ARG_TYPE_PlaylistStartGroupID	*NO*	*NO*			
A_ARG_TYPE_StateVariableList	*NO*	*NO*			
A_ARG_TYPE_PresentationTime	*NO*	*NO*			
A_ARG_TYPE_ClockId	*NO*	*NO*			
A_ARG_TYPE_SyncOffsetAdj	*NO*	*NO*			
Non-standard state variables implemented by a UPnP vendor go here	*TBD*	*TBD*	*TBD*	*TBD*	*TBD*

[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]
SetAVTransportURI()	*R*	*R*
SetNextAVTransportURI()	*A*	*A*
GetMediaInfo()	*R*	*A*
GetMediaInfo_Ext()	*R*	*A*
GetTransportInfo()	*R*	*A*
GetPositionInfo()	*R*	*A*
GetDeviceCapabilities()	*R*	*A*
GetTransportSettings()	*R*	*A*
Stop()	*R*	*R* [c]
Play()	*R*	*R* [c]
Pause()	*A*	*A*
Record()	*A*	*A*
Seek()	*R*	*A*
Next()	*R*	*A*
Previous()	*R*	*A*
SetPlayMode()	*A*	*A*
SetRecordQualityMode()	*CA* [d]	*A*
GetCurrentTransportActions()	*A*	*A*
GetDRMState()	*CR* [e]	*A*
GetStateVariables()	*CR* [e]	*A*
SetStateVariables()	*CR* [e]	*A*
GetSyncOffset()	*CR* [e]	*A*
AdjustSyncOffset()	*CR* [e]	*A*
SetSyncOffset()	*CR* [e]	*A*
SyncPlay()	*CR* [e]	*A*
SyncStop()	*CR* [e]	*A*
SyncPause()	*CA* [d]	*A*
SetStaticPlaylist()	*A*	*A*
SetStreamingPlaylist()	*A*	*A*
GetPlaylistInfo()	*CR* [e]	*A*
Non-standard actions implemented by a UPnP vendor go here	*X*	*X*

Name	R/A [a]	Control Point R/A [b]

a For a device this column indicates whether the action 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 For a control point this column indicates whether a control point shall be capable of invoking this action, where *R* = required, A = allowed, *CR* = conditionally required, *CA* = conditionally allowed, *X* = non-standard, add *–D* when deprecated (e.g., *R-D*, *A-D*).

c Required only if the control point implements interaction with the AVTransport service.

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.

e See action description for conditions under which implementation of this action is required.

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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
CurrentURI	*IN*	*AVTransportURI*
CurrentURIMetaData	*IN*	*AVTransportURIMetaData*

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 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).

5.4.3 *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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
NrTracks	*OUT*	*NumberOfTracks*
MediaDuration	*OUT*	*CurrentMediaDuration*
CurrentURI	*OUT*	*AVTransportURI*
CurrentURIMetaData	*OUT*	*AVTransportURIMetaData*
NextURI	*OUT*	*NextAVTransportURI*
NextURIMetaData	*OUT*	*NextAVTransportURIMetaData*
PlayMedium	*OUT*	*PlaybackStorageMedium*
RecordMedium	*OUT*	*RecordStorageMedium*
WriteStatus	*OUT*	*RecordMediumWriteStatus*

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 InstanceID	The specified *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
CurrentType	*OUT*	*CurrentMediaCategory*
NrTracks	*OUT*	*NumberOfTracks*
MediaDuration	*OUT*	*CurrentMediaDuration*
CurrentURI	*OUT*	*AVTransportURI*
CurrentURIMetaData	*OUT*	*AVTransportURIMetaData*
NextURI	*OUT*	*NextAVTransportURI*
NextURIMetaData	*OUT*	*NextAVTransportURIMetaData*
PlayMedium	*OUT*	*PlaybackStorageMedium*
RecordMedium	*OUT*	*RecordStorageMedium*
WriteStatus	*OUT*	*RecordMediumWriteStatus*

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 InstanceID	The specified *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
CurrentTransportState	*OUT*	*TransportState*
CurrentTransportStatus	*OUT*	*TransportStatus*
CurrentSpeed	*OUT*	*TransportPlaySpeed*

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 InstanceID	The specified *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
Track	*OUT*	*CurrentTrack*
TrackDuration	*OUT*	*CurrentTrackDuration*
TrackMetaData	*OUT*	*CurrentTrackMetaData*
TrackURI	*OUT*	*CurrentTrackURI*
RelTime	*OUT*	*RelativeTimePosition*
AbsTime	*OUT*	*AbsoluteTimePosition*
RelCount	*OUT*	*RelativeCounterPosition*
AbsCount	*OUT*	*AbsoluteCounterPosition*

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 InstanceID	The specified *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
PlayMedia	*OUT*	*PossiblePlaybackStorageMedia*
RecMedia	*OUT*	*PossibleRecordStorageMedia*
RecQualityModes	*OUT*	*PossibleRecordQualityModes*

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 InstanceID	The specified *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
PlayMode	*OUT*	*CurrentPlayMode*
RecQualityMode	*OUT*	*CurrentRecordQualityMode*

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 InstanceID	The specified *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*

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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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	*IN*	*A_ARG_TYPE_InstanceID*
Speed	*IN*	*TransportPlaySpeed*

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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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 *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.

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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*

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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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.

5.4.13 *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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*

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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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 InstanceID	The specified *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
Unit	*IN*	*A_ARG_TYPE_SeekMode*
Target	*IN*	*A_ARG_TYPE_SeekTarget*

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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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 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.

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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*

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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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 *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.

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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*

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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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 *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.

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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
NewPlayMode	*IN*	*CurrentPlayMode*

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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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.)
718	Invalid InstanceID	The specified *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
NewRecordQualityMode	*IN*	*CurrentRecordQualityMode*

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 *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
Actions	*OUT*	*CurrentTransportActions*

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 InstanceID	The specified *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
CurrentDRMState	*OUT*	*DRMState*

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 InstanceID	The specified *InstanceID* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
AVTransportUDN	*IN*	*A_ARG_TYPE_DeviceUDN*
ServiceType	*IN*	*A_ARG_TYPE_ServiceType*
ServiceId	*IN*	*A_ARG_TYPE_ServiceID*
StateVariableValuePairs	*IN*	*A_ARG_TYPE_StateVariableValuePairs*
StateVariableList	*OUT*	*A_ARG_TYPE_StateVariableList*

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 InstanceID	The specified *InstanceID* is invalid for this AVTransport.
728	Invalid State Variable Value	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.

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 subclause 5.2.31 for details.

5.4.23.1 Arguments

Table 67 — Arguments for *GetSyncOffset()*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
CurrentSyncOffset	*OUT*	*SyncOffset*

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	*IN*	*A_ARG_TYPE_InstanceID*
NewSyncOffset	*IN*	*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	*IN*	*A_ARG_TYPE_InstanceID*
Adjustment	*IN*	*A_ARG_TYPE_SyncOffsetAdj*

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 *ReferecePresentationTime*. 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 *RendererPresentationTime* (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		
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
Speed	*IN*	*TransportPlaySpeed*
ReferencePositionUnits	*IN*	*A_ARG_TYPE_SeekMode*
ReferencePosition	*IN*	*A_ARG_TYPE_SeekTarget*
ReferencePresentationTime	*IN*	*A_ARG_TYPE_PresentationTime*
ReferenceClockId	*IN*	*A_ARG_TYPE_ClockId*

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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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 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.
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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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 occured 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	*IN*	*A_ARG_TYPE_InstanceID*
PauseTime	*IN*	*A_ARG_TYPE_PresentationTime*
ReferenceClockId	*IN*	*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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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 occured 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 *InstanceID*	The specified *InstanceID* is invalid for this AVTransport.
735	Incorrect Playlist length	The *PlaylistDataLength* 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
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
PlaylistType	*IN*	*A_ARG_TYPE_PlaylistType*
PlaylistInfo	*OUT*	*A_ARG_TYPE_PlaylistInfo*

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 *InstanceID*	The specified *InstanceID* 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 *hold locked*. (Some portable mobile devices have a small mechanical toggle switch called a *hold lock switch*. 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 *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.
726	Invalid StateVariableList	Some of the variables are invalid.
727	Ill-formed CSV List	The CSV list is not well formed.

errorCode	errorDescription	Description
728	Invalid State Variable Value	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 occured 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>
            <argument>
               <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>
         <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>
                <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>
            <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">
```