AVTransport:1 Service Template Version 1.01

For UPnP™ Version 1.0
Status: Standardized DCP
Date: June 25, 2002

Authors	Company
Larry Buerk	Microsoft Corporation
Jean Moonen	Philips Electronics
Dale Sather	Microsoft Corporation
John Kai Fu	Pioneer Research Center

Contents

List of Tables

1. Overview and Scope

This service definition is compliant with the UPnP Device Architecture version 1.0.

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 meta data embedded in the resource itself.

This service type does not offer *scheduled* recording.

2. Service Modeling Definitions

2.1. ServiceType

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

urn:schemas-upnp-org:service:*AVTransport:1*

2.2. State Variables

Table 1: State Variables

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value	Default Value	Eng. Units
TransportState	R	*string*	*"STOPPED"* *"PLAYING"* see Table 1.1		
TransportStatus	R	*string*	*"OK",* *"ERROR_OCCURRED"* *vendor-extensible*		
PlaybackStorageMedium	R	*string*	see Table 1.2		
RecordStorageMedium	R	*string*	see Table 1.2		
PossiblePlaybackStorageMedia	R	*string*	CSV[1] (string)		
PossibleRecordStorageMedia	R	*string*	CSV (string)		
CurrentPlayMode	R	*string*	*"NORMAL"* see Table 1.3	*"NORMAL"*	
TransportPlaySpeed	R	*string*	*"1"*	*"1"*	
RecordMediumWriteStatus	R	*string*	see Table 1.4		
CurrentRecordQualityMode	R	*string*	see Table 1.5		
PossibleRecordQualityModes	R	*string*	CSV (string)		
NumberOfTracks	R	*ui4*	*Min = 0* see Table 1.6		
CurrentTrack	R	*ui4*	*Min = 0* *Step = 1* see Table 1.7		
CurrentTrackDuration	R	*string*			
CurrentMediaDuration	R	*string*			
CurrentTrackMetaData	R	*string*			

[1] 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.

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value	Default Value	Eng. Units
CurrentTrackURI	R	*string*			
AVTransportURI	R	*string*			
AVTransportURIMetaData	R	*string*			
NextAVTransportURI	R	*string*			
NextAVTransportURIMetaData	R	*string*			
RelativeTimePosition	R	*string*			
AbsoluteTimePosition	R	*string*			
RelativeCounterPosition	R	*i4*			
AbsoluteCounterPosition	R	*i4*			
CurrentTransportActions	O	*string*	CSV (string)		
LastChange	R	*string*			
A_ARG_TYPE_SeekMode	R	*string*	*"TRACK_NR"* see Table 1.8	*n/a*	
A_ARG_TYPE_SeekTarget	R	*string*		*n/a*	
A_ARG_TYPE_InstanceID	R	*ui4*		*n/a*	

[1] R = Required, O = Optional, X = Non-standard.

Table 1.1: allowedValueList for *TransportState*

Value	Req. or Opt.
"STOPPED"	R
"PLAYING"	R
"TRANSITIONING"	O
"PAUSED_PLAYBACK"	O
"PAUSED_RECORDING"	O
"RECORDING"	O
"NO_MEDIA_PRESENT"	O

Table 1.2: allowedValueList for *PlaybackStorageMedium*

Value	Req. or Opt.
"UNKNOWN"	O
"DV"	O
"MINI-DV"	O

"VHS"	O
"W-VHS"	O
"S-VHS"	O
"D-VHS"	O
"VHSC"	O
"VIDEO8"	O
"HI8"	O
"CD-ROM"	O
"CD-DA"	O
"CD-R"	O
"CD-RW"	O
"VIDEO-CD"	O
"SACD"	O
"MD-AUDIO"	O
"MD-PICTURE"	O
"DVD-ROM"	O
"DVD-VIDEO"	O
"DVD-R"	O
"DVD+RW"	O
"DVD-RW"	O
"DVD-RAM"	O
"DVD-AUDIO"	O
"DAT"	O
"LD"	O
"HDD"	O
"MICRO-MV"	O
"NETWORK"	O
"NONE"	O
"NOT_IMPLEMENTED"	O
Vendor-defined	O

Table 1.3: allowedValueList for *CurrentPlayMode*

Value	Req. or Opt.
“NORMAL”	R
“SHUFFLE”	O
“REPEAT_ONE”	O
“REPEAT_ALL”	O
“RANDOM”	O
“DIRECT_1”	O
“INTRO”	O
Vendor-defined	O

Table 1.4: allowedValueList for *RecordMediumWriteStatus*

Value	Req. or Opt.
“WRITABLE”	O
”PROTECTED”	O
“NOT_WRITABLE”	O
“UNKNOWN”	O
“NOT_IMPLEMENTED”	O

Table 1.5: allowedValueList for *CurrentRecordQualityMode*

Value	Req. or Opt.
“0:EP”	O
“1:LP”.	O
“2:SP”	O
“0:BASIC”	O
“1:MEDIUM”	O
“2:HIGH”	O
“NOT_IMPLEMENTED”	O
Vendor-defined	O

Table 1.6: allowedValueRange for *NumberOfTracks*

	Value	Req. or Opt.
minimum	*0*	R
maximum	*vendor-defined*	R

Table 1.7: allowedValueRange for *CurrentTrack*

	Value	Req. or Opt.
minimum	*0*	R
maximum	*vendor-defined*	R
step	*1*	R

Table 1.8: allowedValueList for *A_ARG_TYPE_SeekMode*

Value	Req. or Opt.
"TRACK_NR"	R
"ABS_TIME"	O
"REL_TIME"	O
"ABS_COUNT"	O
"REL_COUNT"	O
"CHANNEL_FREQ"	O
"TAPE-INDEX"	O
"FRAME"	O

2.2.1. *TransportState*

This state variable forms the 'core' of the AVTransport service. It defines the conceptually 'top-level' state of the transport, e.g., whether it is playing, recording, etc. Device vendors do not need to implement all allowed values of this variable, e.g., 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 may respond faster or more accurate. 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.

2.2.2. TransportStatus

During operation of the AVTransport service, asynchronous errors may occur 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 may be network congestion or server problems causing hickups in the rendered media. These types of situations can be signalled to control points by setting this state variable to value “ERROR_OCCURRED”. More specific error descriptions may also be used, as vendor extensions. The 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.

2.2.3. PlaybackStorageMedium

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 may be added in a vendor-specific way.

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

2.2.4. RecordStorageMedium

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 may be added in a vendor-specific way.

Note that this variable is not intended for signal- or content-formats such as, e.g., MPEG2. Such type of information is exposed by the ConnectionManager service associated with this service. If the service implementation doesn’t supported recording then this state variable must be set to value “NOT_IMPLEMENTED”.
.

2.2.5. PossiblePlaybackStorageMedia

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

2.2.6. PossibleRecordStorageMedia

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 variable RecordStorageMedium. If the service implementation doesn’t supported recording then this state variable must be set to value “NOT_IMPLEMENTED”.

2.2.7. CurrentPlayMode

Indicates the current play mode (e.g., 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.

2.2.8. TransportPlaySpeed

String representation of a rational fraction, indicates the speed relative to normal speed. Example values are ‘1’, ‘1/2’, ‘2’, ‘-1’, ‘1/10’, etc. Actually supported speeds can be retrieved from the AllowedValueList of this state variable in the AVTransport service description. Value ‘1’ is required, value ‘0’ is not allowed.

2.2.9. RecordMediumWriteStatus

Write protection status of currently loaded media. NOT_WRITABLE indicates an inherent ‘read-only’ media (e.g., 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 (e.g., a protected VHS tape). If no media is loaded, the write status will be “UNKNOWN”. If the service implementation doesn’t support recording then this state variable must be set to value “NOT_IMPLEMENTED”.

2.2.10.CurrentRecordQualityMode

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 doesn’t support recording then this state variable must be set to value “NOT_IMPLEMENTED”.

2.2.11.PossibleRecordQualityModes

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 doesn’t support recording then this state variable must be set to value “NOT_IMPLEMENTED”.

2.2.12.NumberOfTracks

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. Otherwise, it shall be 1 or higher.

For tape-based media that do not support the notion of tracks, this state variable will always be ‘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 1. However, if AVTransportURI points to a playlist file, for example, then NumberOfTracks shall be equal to the number of entries in the playlist.

2.2.13.CurrentTrack

If NumberOfTracks is 0, then CurrentTrack will be 0. Otherwise, this state variable will contain the sequence number of the currently selected track, starting at value ‘1’, up to and including NumberOfTracks. For tape-based media that do not support the notion of tracks, this state variable will always be ‘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.

2.2.14.CurrentTrackDuration

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

The string may be preceded by an optional + or – sign, and the decimal point itself may be omitted if there are no fractional second digits. This variable does not apply to Tuners. If the service implementation doesn’t support track duration information then this state variable must be set to value “NOT_IMPLEMENTED”.

2.2.15.CurrentMediaDuration

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. 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 “00:00:00”. If the service implementation doesn’t support media duration information then this state variable must be set to value “NOT_IMPLEMENTED”.

2.2.16.CurrentTrackMetaData

Metadata, in the form of a valid DIDL-Lite XML fragment (defined in the ContentDirectory service template), associated with the resource pointed to by state variable CurrentTrackURI. The meta data may have been extracted from state variable AVTransportURIMetaData, or extracted from the resource binary itself (e.g., embedded ID3 tags for MP3 audio). This is implementation dependent. If the service implementation doesn’t support this feature then this state variable must be set to value “NOT_IMPLEMENTED”.

2.2.17.CurrentTrackURI

Reference, in the form of a URI, to the current track. The URI enables a control point to retrieve any meta data associated with current track, such as, for example, title and author information, via the ContentDirectory service.

2.2.18.AVTransportURI

Reference, in the form of a URI of the resource controlled by the AVTransport instance. This URI may refer to a 'single' item (e.g., a song) or to a collection of items (e.g., 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, for example, title and author information, via the ContentDirectory service.

2.2.19.AVTransportURIMetaData

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 AVTransportURI. See the ContentDirectory service specification for details. If the service implementation doesn't support this feature then this state variable must be set to value "NOT_IMPLEMENTED".

2.2.20.NextAVTransportURI

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 streaming protocols that need buffering (e.g. HTTP GET). If the service implementation doesn't support this feature then this state variable must be set to value "NOT_IMPLEMENTED".

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

2.2.21.NextAVTransportURIMetaData

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 NextAVTransportURI. See the ContentDirectory service specification for details. If the service implementation doesn't support this feature then this state variable must be set to value "NOT_IMPLEMENTED".

2.2.22.RelativeTimePosition

This state variable contains the current position, in terms of time, from the beginning of the current track. For tape-based media that do not support multiple tracks on a single tape, this state variable contains the position, in terms of time, from a *zero reference point* on the tape. The time format is the same as for state variable CurrentTrackDuration. If the service implementation doesn't support relative time-based position information then this state variable must be set to value "NOT_IMPLEMENTED".

2.2.23.AbsoluteTimePosition

This state variable contains the current position, in terms of a time, from the beginning of the media. The time format is the same as for state variable CurrentTrackDuration. If the service implementation doesn't support any kind of position information then this state variable must be set to value "NOT_IMPLEMENTED". Devices that don't have time position information, but which are able to detect whether they are at the end of the media or not should use special value "END_OF_MEDIA" when actually at the end, and "NOT_IMPLEMENTED" otherwise.

2.2.24.RelativeCounterPosition

This state variable contains the current position, in terms of a dimensionless counter, from the beginning of the current track. For tape-based media that do not support multiple tracks on a single tape, this state variable contains the position, in terms of a dimensionless counter, from a *zero reference point* on the tape. If the service implementation doesn't support relative count-based position information then this state variable must be set to the maximum value of the i4 data type.

2.2.25.AbsoluteCounterPosition

This state variable contains the current position, in terms of a dimensionless counter, from the beginning of the loaded media. If the service implementation doesn't support absolute count-based position information then this state variable must be set to the maximum value of the i4 data type.

2.2.26.CurrentTransportActions

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 will contain a subset of the following actions: Play, Stop, Pause, Seek, Next, Previous and Record. For example, when a live stream from the Internet is being controlled, the variable may be only "Play, Stop". When a local audio CD is being controlled, the variable may be "Play, Stop, Pause, Seek, Next, Previous". This information can be used, for example, to dynamically enable or disable play, stop, pause buttons, etc., on a user interface.

2.2.27.LastChange

This variable is used for eventing purposes to allow 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 Section 5. For additional information, refer to the 'LastChange' state variable defined in the RenderingControl service template.

2.2.28.A_ARG_TYPE_SeekMode

This state variable is introduced to provide type information for the "seek mode" parameter 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 (e.g., for tapes with a indexing facility) or even a video frame. A device vendor is allowed to implement a subset of the allowed value list of this state variable. Only value 'TRACK_NR' is required.

2.2.29.A_ARG_TYPE_SeekTarget

This state variable is introduced to provide type information for the "target" parameter 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 must contains string representations of values of UPnP types 'ui4' (ABS_COUNT, REL_COUNT, TRACK_NR, TAPE-INDEX, FRAME), 'time' (ABS_TIME, REL_TIME) or 'float' (CHANNEL_FREQ, in Hz). Supported ranges of these integer, time or float values are device-dependent.

2.2.30.A_ARG_TYPE_InstanceID

This state variable is introduced to provide type information for the "InstanceID" input parameter 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 PrepareForConnection action.

If the device's ConnectionManager does not implement the optional PrepareForConnection action, special value '0' may be used for the "InstanceID" input parameter. In such a case, the device implements a single static AVTransport instance, and only 1 stream can be controlled and sent (or received) at any time.

2.3. Eventing and Moderation

Table 2: Event Moderation

Variable Name	Evented	Moderated Event	Max Event Rate[1]	Logical Combination	Min Delta per Event[2]
TransportState	*NO*	*NO*			
TransportStatus	*NO*	*NO*			
PlaybackStorageMedium	*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*			
CurrentMediaDuration	*NO*	*NO*			
CurrentTrackURI	*NO*	*NO*			
CurrentTrackMetaData	*NO*	*NO*			
AVTransportURI	*NO*	*NO*			
AVTransportURIMetaData	*NO*	*NO*			
NextAVTransportURI	*NO*	*NO*			
NextAVTransportURIMetaData	*NO*	*NO*			
CurrentTransportActions	*NO*	*NO*			
LastChange	*YES*	*YES*	*0.2*		

[1] Determined by N, where Rate = (Event)/(N secs).
[2] (N) * (allowedValueRange Step).

2.3.1. Event Model

Since the AVTransport Service supports multiple logical instances (via the InstanceID parameter 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" state variables listed 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 Event Model section of the RenderingControl service.

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

- RelativeTimePosition
- AbsoluteTimePosition
- RelativeCounterPosition
- 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.

2.4. Actions

Table 3: Actions

Name	Req. or Opt. [1]
SetAVTransportURI	R
SetNextAVTransportURI	O
GetMediaInfo	R
GetTransportInfo	R
GetPositionInfo	R
GetDeviceCapabilities	R
GetTransportSettings	R
Stop	R
Play	R
Pause	O
Record	O
Seek	R
Next	R
Previous	R
SetPlayMode	O
SetRecordQualityMode	O
GetCurrentTransportActions	O

[1] R = Required, O = Optional, X = Non-standard.

2.4.1. *SetAVTransportURI*

This 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. A control point can supply meta data associated with the specified resource, using a DIDL-Lite XML fragment (defined in the ContentDirectory service template), in parameter CurrentURIMetaData. If supported by the AVTransport, this meta data is stored in a state variable, and returned as output parameter as part of action GetMediaInfo. If a control point does not want to use this feature it can supply NULL (empty string) for the CurrentURIMetaData parameter.

2.4.1.1. .Arguments

Table 4: Arguments for *SetAVTransportURI*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
CurrentURI	*IN*	*AVTransportURI*
CurrentURIMetaData	*IN*	*AVTransportURIMetaData*

2.4.1.2. *Dependency on State (if any)*

2.4.1.3. Effect on State (if any).

Depending on the URI, the number of tracks available on this instance may 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 should change to “STOPPED”. If the current transport state is “PLAYING”, and it would take a noticable amount of time before a human user would actually see or hear the media at the new URI playing, the AVTransport is allowed to temporarily go to the “TRANSITIONING” state before going back to “PLAYING”. This might be appropriate, for example, 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.

2.4.1.4. Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
714	Illegal MIME-type	The specified resource has a MIME-type which is not supported by the AVTransport service
715	Content ‘BUSY’	This indicates the resource is already being played by other means. The actual implementation might detect through HTTP Busy, and returns this error code.
716	Resource not found	The specified resource cannot be found in the network
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.2. SetNextAVTransportURI

This action specifies the URI of the resource to be controlled when the playback of the current resource (set earlier via SetAVTransportURI) finishes. This action allows 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 protocol such as HTTP GET, 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.

A control point can supply meta data, using a DIDL-Lite XML fragment (defined in the ContentDirectory service template), via parameter NextURIMetaData. If supported by the AVTransport, this meta data is stored in a state variable, and returned as output parameter as part of action GetMediaInfo. If a control point does not want to use this feature it can supply NULL (empty string) for the NextURIMetaData parameter.

2.4.2.1. Arguments

Table 5: Arguments for SetNextAVTransportURI

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
NextURI	*IN*	*NextAVTransportURI*
NextURIMetaData	*IN*	*NextAVTransportURIMetaData*

2.4.2.2. *Dependency on State (if any)*

2.4.2.3. Effect on State (if any)

This action does **not** change the transport state of the specified instance. In case that the next URI buffer exists (e.g. 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 NextAVTransportURI MetaData. The process repeats itself until there is no more URI to be rendered. In such case, the state variable NextAVTransportURI will be set to NULL (empty string).

If an illegal URI is used for the SetNextAVTransportURI, which is detected immediately, and most likely while the current URI is still being rendered, the TransportState should be kept. After the current URI finishes playing, the transition to that illegal URI cannot be made. and the TransportState should be set to "STOPPED".

2.4.2.4. Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
714	Illegal MIME-type	The specified resource has a MIME-type which is not supported by the AVTransport service
715	Content 'BUSY'	This indicates the resource is already being played by other means. The actual implementation might detect through HTTP Busy, and returns this error code.
716	Resource not found	The specified resource cannot be found in the network
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.3. *GetMediaInfo*

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

2.4.3.1. Arguments

Table 6: 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*

2.4.3.2. Dependency on State (if any)

2.4.3.3. Effect on State (if any)

2.4.3.4. Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.4. GetTransportInfo

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

2.4.4.1. Arguments

Table 7: Arguments for *GetTransportInfo*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
CurrentTransportState	*OUT*	*TransportState*
CurrentTransportStatus	*OUT*	*TransportStatus*
CurrentSpeed	*OUT*	*TransportPlaySpeed*

2.4.4.2. Dependency on State (if any)

2.4.4.3. Effect on State (if any).

2.4.4.4. Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.5. GetPositionInfo

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

2.4.5.1. Arguments

Table 8: 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*

Dependency on State (if any)

2.4.5.2. Effect on State (if any)

2.4.5.3. Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.6. GetDeviceCapabilities

This 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.

2.4.6.1. Arguments

Table 9: Arguments for *GetDeviceCapabilities*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
PlayMedia	*OUT*	*PossiblePlaybackStorageMedia*
RecMedia	*OUT*	*PossibleRecordStorageMedia*
RecQualityModes	*OUT*	*PossibleRecordQualityModes*

2.4.6.2. Dependency on State (if any)

2.4.6.3. Effect on State (if any)

2.4.6.4. Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.7. GetTransportSettings

This 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.

2.4.7.1. Arguments

Table 10: Arguments for *GetTransportSettings*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
PlayMode	*OUT*	*CurrentPlayMode*
RecQualityMode	*OUT*	*CurrentRecordQualityMode*

2.4.7.2. Dependency on State (if any)

2.4.7.3. Effect on State (if any)

2.4.7.4. Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.8. Stop

This 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 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 (e.g. MediaServer devices) "output" media content to the network while other devices (e.g. a MediaRenderer) "output" a visual and/or audio representation of media content that was received from the network.

2.4.8.1. Arguments

Table 11: Arguments for *Stop*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*

2.4.8.2. Dependency on State (if any)

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

2.4.8.3. Effect on State (if any)

Changes TransportState to "STOPPED".

2.4.8.4. Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
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, i.e., 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.

2.4.9. Play

Start playing the resource of the specified instance, at the specified speed, starting at the current position, according to the current play mode. Keep playing until the resource ends or the transport state is changed via actions Stop or Pause. The device should do a 'best effort' to match the specified play speed. Actually supported speeds can be retrieved from the AllowedValueList of the TransportPlaySpeed state variable in the AVTransport service description.

If no AVTransportURI is set, the resource being played is device-dependent.

2.4.9.1. Arguments

Table 12: Arguments for *Play*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
Speed	*IN*	*TransportPlaySpeed*

2.4.9.2. Dependency on State (if any)

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

2.4.9.3. Effect on State (if any)

Changes TransportState to "PLAYING" and TransportPlaySpeed to normal speed in forward direction ("1"). If it would take a noticable amount of time before a human user would actually see or hear the media playing, the AVTransport is allowed to 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.

2.4.9.4. Errors

errorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
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 (e.g., 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, i.e., 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 the resource is already being played by other means. The actual implementation might detect through HTTP Busy, and returns this error code.
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.

2.4.10.Pause

While the device is in a playing state, e.g. TransportState is “PLAYING”, this action halts the progression of the resource that is associated with the specified instance Id. Any visual representation of the resource should remain displayed in a static manner (e.g. the last frame of video should remain displayed). Any audio representation of the resource should be muted. The difference between Pause() and Stop() is that Pause() MUST remain at the current position within the resource and the current resource must persist as describe above (e.g. the current video resource continues to be transmitted/displayed).

When the device is recording, e.g., the TransportState is “RECORDING”, the device MUST maintain its current recording position, but does not accept any more data to record. Any data received after the Pause() action and before the next Record() action will be lost.

2.4.10.1.Arguments

Table 13: Arguments for Pause

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*

2.4.10.2.Dependency on State (if any)

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

2.4.10.3.Effect on State (if any)

When recording, changes TransportState to "PAUSED_RECORDING". When playing, changes TransportState to "PAUSED_PLAYBACK". The Pause action does not operate as a toggle.

2.4.10.4. Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
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, i.e., 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.

2.4.11.Record

Start recording on the specified transport instance, at the current position on the media, according to the currently specified recording quality, and return 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 recording 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. Arguments

Table 14: Arguments for *Record*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*

2.4.11.1.Dependency on State (if any)

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

2.4.11.2.Effect on State (if any)

Changes TransportState to “RECORDING”.

2.4.11.3.Errors

errorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
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, i.e., 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 (e.g., 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.

2.4.12.Seek

Start seeking through the resource controlled by the specified instance - as fast as possible - to the specified target position. Unit value “TRACK_NR” indicates seeking to a particular track number. For tape-based media that do not support the notion of track (such as VCRs), Seek(“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(“TRACK_NR”,”0”) is equivalent to the common “FastForward” VCR functionality.

2.4.12.1.Arguments

Table 15: Arguments for Seek

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
Unit	*IN*	*A_ARG_TYPE_SeekMode*
Target	*IN*	*A_ARG_TYPE_SeekTarget*

2.4.12.2.Dependency on State (if any)

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

2.4.12.3.Effect on State (if any)

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 event mechanism.

2.4.12.4.Errors

errorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
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, i.e., 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 specified in terms of the seek mode, or is not present on the media.
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.13.Next

Convenient action 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.

2.4.13.1.Arguments

Table 16: Arguments for *Next*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*

2.4.13.2.Dependency on State (if any)

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

2.4.13.3.Effect on State (if any)

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 it can be detected through the event mechanism..

2.4.13.4.Errors

errorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
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, i.e., 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.)
711	Illegal seek target	The specified seek target is not specified in terms of the seek mode, or is not present on the media.
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.14.Previous

Convenient action 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.

2.4.14.1.Arguments

Table 17: Arguments for *Previous*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*

2.4.14.2.Dependency on State (if any)

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

2.4.14.3.Effect on State (if any)

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 it can be detected through the event mechanism..

2.4.14.4.Errors

errorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
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, i.e., 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.)
711	Illegal seek target	The specified seek target is not specified in terms of the seek mode, or is not present on the media.
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.15.SetPlayMode

Sets the play mode of the specified AVTransport instance.

2.4.15.1.Arguments

Table 18: Arguments for *SetPlayMode*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
NewPlayMode	*IN*	*CurrentPlayMode*

2.4.15.2.Dependency on State (if any)

2.4.15.3.Effect on State (if any)

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.

2.4.15.4.Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.

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, i.e., 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.

2.4.16.SetRecordQualityMode

Sets the record quality mode of the specified AVTransport instance.

2.4.16.1.Arguments

Table 19: Arguments for *SetRecordQualityMode*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
NewRecordQualityMode	*IN*	*CurrentRecordQualityMode*

2.4.16.2.Dependency on State (if any)

2.4.16.3.Effect on State (if any)

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.

2.4.16.4.Errors

ErrorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
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.

2.4.17.GetCurrentTransportActions

Returns the CurrentTransportActions state variable for the specified instance.

2.4.17.1.Arguments

Table 20: Arguments for *GetCurrentTransportActions*

Argument	Direction	relatedStateVariable
InstanceID	*IN*	*A_ARG_TYPE_InstanceID*
Actions	*OUT*	*CurrentTransportActions*

2.4.17.2.Dependency on State (if any)

2.4.17.3.Effect on State (if any)

2.4.17.4.Errors

errorCode	errorDescription	Description
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
718	Invalid InstanceID	The specified instanceID is invalid for this AVTransport.

2.4.18.Common Error Codes

The following table 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 21: Common Error Codes

errorCode	errorDescription	Description
401	Invalid Action	No action by that name at this service.
402	Invalid Args	Could be any of the following: not enough in args, too many in args, no in arg by that name, one or more in args are of the wrong data type.
404	Invalid Var	No state variable by that name at this service.
501	Action Failed	May be returned in current state of service prevents invoking that action.

errorCode	errorDescription	Description
600-699		Common action errors. Defined by UPnP Forum Technical Committee.
701-799		Common action errors defined by the UPnP Forum working committees.
800-899		*(Specified by UPnP vendor.)*

2.5. Theory of Operation

2.5.1. TransportState Control

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

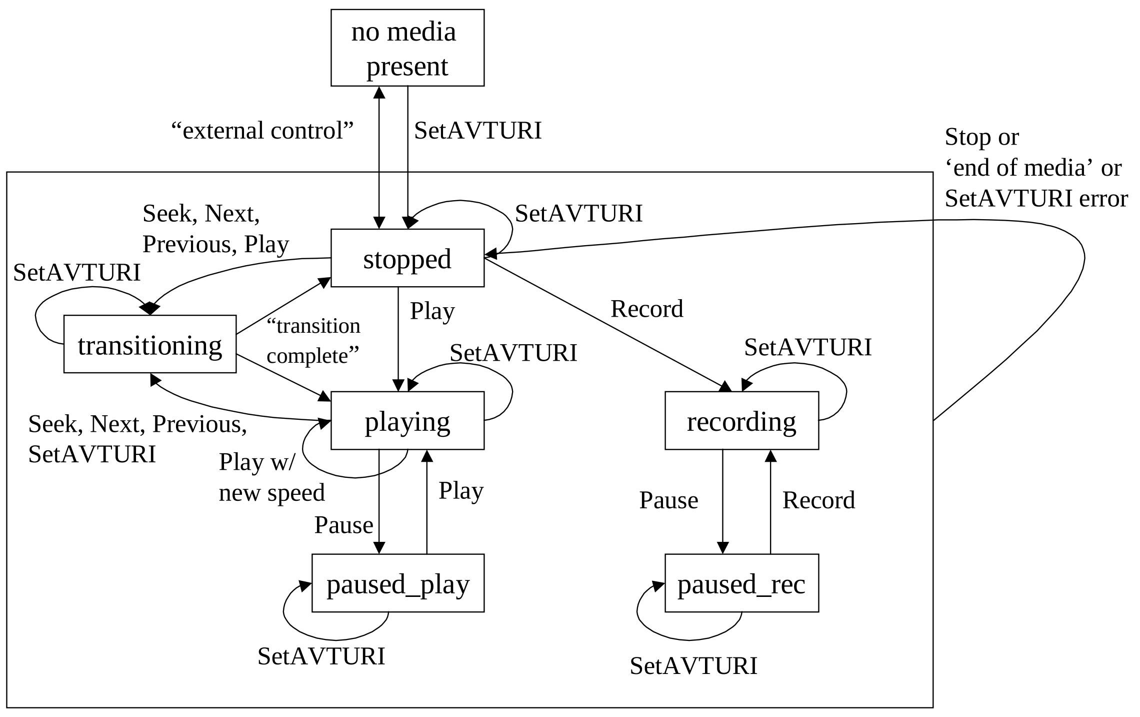

Figure 1: TransportState transitions - INFORMATIVE

Note that the Stop action is allowed in all states except "no media present", and returns the TransportState to the "stopped" value.

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

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

2.5.2. Transport Settings

Besides control over the transport state, the AVTransport also allows control over various settings related to playback and control. These settings, such as play mode, record mode and record quality mode only take effect on subsequent Play or Record actions. In other words, they do not change the behavior on any ongoing playback or record 'session'.

2.5.3. Navigation

The AVTransport allows two types of navigation through the media:

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

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

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

The Seek action is very generic and allows a control point to specify a seek operation in various 'dimensions'. For example, a control point may instruct a device to position itself 30 seconds from the current position (using the "RELATIVE_TIME" seek mode) or to go to track number 12 (using the "TRACK_NR" seek mode).

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

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

2.5.4. AVTransportURI Concept

Uniform Resource Identifiers (URIs) are the Internet standard for resource identification. URIs are simply character strings which identify abstract or physical resources. The complete URI definition is available in the text Uniform Resource Identifiers (URI): Generic Syntax, RFC 2396 (http://www.ietf.org/rfc/rfc2396.txt).

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

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

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

The table below lists how the protocolInfo definitions from the ConnectionManager specification relate to valid URLs. The appendix provides a more detailed explanation per protocol.

Table 22: Allowed AVTransportURIs

ProtocolInfo	URI Scheme	Reference
`http-get`	`http`	Section 6.1
`rtsp-rtp-udp`	`rtsp`	Section 6.2
`internal`	`file`	Section 6.3
`iec61883`	`«Vendor-specific»`	Section 6.4
`«registered ICANN domain name of vendor»`	`«Vendor-specific»`	Section 6.5

2.5.5. AVTransport Abstraction

Via the SetAVTransportURI action an AVTransport service instance is ‘bound’ to a content resource. Content resources are exposed by the ContentDirectory service. A content resource can represent a single atomic piece of content (e.g., a single song), or a collection of contents (e.g., a CD disc or playlist). The *types* of content resources that can be sent or received by a device are exposed by the GetProtocolInfo action of the device’s ConnectionManager service.

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

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

The AVTransport abstracts and minimizes the differences between various specific transport media such as tapes, discs and solid-state media. The tables below gives an overview on how generic AVTransport concepts such as “track” and “next” and ‘previous” actions apply to certain specific types of AVTransportURIs. The precise mapping is implementation-dependent.

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

PVR – Single Stored program	1 program	1	duration of the program	no effect
EPF	1 image	all files of the slide show	display time of the slide in the slide show	next of previous slide in a slide show.

Figure 2: Example mappings of resources type to track sequences.

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

AVTransport URI	Applicable (not required) Seek modes	Applicable (not required) Play modes	Pausing possible
Audio CD	TRACK_NR	NORMAL, SHUFFLE, REPEAT_ONE, REPEAT_ALL, RANDOM, DIRECT_1, INTRO	yes
Audio CD Changer	TRACK_NR	NORMAL, SHUFFLE, REPEAT_ONE, REPEAT_ALL, RANDOM, DIRECT_1, INTRO	yes
Audio Playlist **(HDD/SolidState-based player)**	TRACK_NR	NORMAL, SHUFFLE, REPEAT_ONE, REPEAT_ALL, RANDOM, DIRECT_1, INTRO	yes
Video DVD-Volume	TRACK_NR, FRAME	NORMAL	yes
Video DVD Changer	TRACK_NR, FRAME	NORMAL	yes
VCR (Tape)	TRACK_NR, ABS_TIME, REL_TIME, ABS_COUNT , REL_COUNT, TAPE_INDEX, FRAME	NORMAL	yes
List-based Tuner	TRACK_NR	NORMAL	no

Frequency-based Tuner	TRACK_NR	NORMAL	no
PVR – Tuner subsystem	TRACK_NR, ABS_TIME, REL_TIME, FRAME	NORMAL	yes
PVR – Collection of Stored programs	TRACK_NR, ABS_TIME, REL_TIME, FRAME	NORMAL	yes
PVR – Stored program	TRACK_NR, ABS_TIME, REL_TIME, FRAME	NORMAL	yes
EPF	TRACK_NR, ABS_TIME	NORMAL, SHUFFLE, REPEAT_ONE, REPEAT_ALL, RANDOM, DIRECT_1	yes

Figure 3: Example seek modes, play modes and transport actions, per resource type.

2.5.6. Supporting multiple virtual Transports

The UPnP Architecture v1.0 requires the number of service instances in a device to be static. In certain cases it is desirable for devices to offer a dynamic number of 'virtual' service instances. A control point should be able to control and receive events from each virtual instance individually.

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

A generic strategy to achieve this is as follows:

- Have a single static UPnP service instance in a device (hence, a single UPnP serviceId).
    - In the AVTransport case, a MediaServer device or MediaRenderer device can have a single UPnP AVTransport instance.
- Define the notion of a virtual 'instance identifier' (this is **not** the UPnP serviceId).
    - In the AVTransport case, a 'ui4' value.
- Add to all actions of the service definition an input parameter that holds the virtual instance identifier to which the action applies.
    - All actions in the AVTransport service, such as Play, Stop, Pause, have as first parameter an input parameter of type ui4 that identifies the instance.
- Add an evented state variable (in this case "LastChange") to the service that holds both the instance identifier and the name and value of the latest state change of this instance. All other variable are not evented.

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

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

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

2.5.7. Playlist Playback

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

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

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

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

3. XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
    <specVersion>
        <major>1</major>
        <minor>0</minor>
    </specVersion>
    <actionList>
        <action>
            <name>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>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>
    </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>
            </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>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+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> 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+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> 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>
            </allowedValueList>
            <defaultValue>NORMAL</defaultValue>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>TransportPlaySpeed</name>
            <dataType>string</dataType>
            <allowedValueList>
                <allowedValue>1</allowedValue>
                <allowedValue> vendor-defined </allowedValue>
            </allowedValueList>
        </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>
            </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>i4</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>CurrentTransportActions</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="yes">
            <name>LastChange</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_SeekMode</name>
            <dataType>string</dataType>
            <allowedValueList>
                <allowedValue>ABS_TIME</allowedValue>
                <allowedValue>REL_TIME</allowedValue>
                <allowedValue>ABS_COUNT</allowedValue>
                <allowedValue>REL_COUNT</allowedValue>
                <allowedValue>TRACK_NR</allowedValue>
                <allowedValue>CHANNEL_FREQ</allowedValue>
                <allowedValue>TAPE-INDEX</allowedValue>
                <allowedValue>FRAME</allowedValue>
            </allowedValueList>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_SeekTarget</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_InstanceID</name>
            <dataType>ui4</dataType>
        </stateVariable>
    </serviceStateTable>
</scpd>
```

4. Test

No semantic tests have been specified for this service.

5. “LastChange” State Variable Schema

The following XML schema describes the format of the LastChange state variable, which is used for eventing state changes that occur with this service. Refer to Sections 2.2.27 and 2.3.1 for additional details.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:schemas-upnp-org:metadata-1-0/AVT/"
xmlns:avt="urn:schemas-upnp-org:metadata-1-0/AVT/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">Schema for UPnP A/V AVTransport
Service events, version 0.1</xsd:documentation>
    </xsd:annotation>
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
    <!--============================================================

     'Event' is the root element of AVTransport event document
fragments.
     'InstanceID' is the only valid child of 'Event'.

    ============================================================-->
    <xsd:element name="Event" type="avt:rootType"/>
    <xsd:complexType name="rootType">
        <xsd:annotation>
            <xsd:documentation>Event is the root
element</xsd:documentation>
        </xsd:annotation>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="avt:InstanceID"/>
        </xsd:choice>
    </xsd:complexType>
    <!--============================================================

     'InstanceID' elements identify an individual event instance.

    ============================================================-->
    <xsd:group name="allowed-under-InstanceID">
        <xsd:annotation>
            <xsd:documentation>This group defines the elements allowed
under the InstanceID element</xsd:documentation>
        </xsd:annotation>
        <xsd:choice>
            <xsd:element ref="avt:TransportState"/>
            <xsd:element ref="avt:TransportStatus"/>
            <xsd:element ref="avt:PlaybackStorageMedium"/>
            <xsd:element ref="avt:RecordStorageMedium"/>
            <xsd:element ref="avt:PossiblePlaybackStorageMedia"/>
            <xsd:element ref="avt:PossibleRecordStorageMedia"/>
            <xsd:element ref="avt:CurrentPlayMode"/>
            <xsd:element ref="avt:TransportPlaySpeed"/>
            <xsd:element ref="avt:RecordMediumWriteStatus"/>
            <xsd:element ref="avt:CurrentRecordQualityMode"/>
            <xsd:element ref="avt:PossibleRecordQualityMode"/>
            <xsd:element ref="avt:NumberOfTracks"/>
            <xsd:element ref="avt:CurrentTrack"/>
            <xsd:element ref="avt:CurrentTrackDuration"/>
            <xsd:element ref="avt:CurrentMediaDuration"/>
            <xsd:element ref="avt:CurrentTrackMetadata"/>
            <xsd:element ref="avt:CurrentTrackURI"/>
            <xsd:element ref="avt:AVTransportURI"/>
            <xsd:element ref="avt:NextAVTransportURI"/>
            <xsd:element ref="avt:CurrentTransportActions"/>
        </xsd:choice>
    </xsd:group>
    <xsd:element name="InstanceID" type="avt:InstanceIDtype"/>
    <xsd:complexType name="InstanceIDtype">
        <xsd:annotation>
            <xsd:documentation>InstanceID elements identify an
individual event instance.</xsd:documentation>
        </xsd:annotation>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="avt:allowed-under-InstanceID"/>
        </xsd:choice>
    </xsd:complexType>
    <!--============================================================

     TransportState

    ============================================================-->
    <xsd:element name="TransportState">
        <xsd:complexType>
            <xsd:attribute name="val" use="required">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="STOPPED"/>
                        <xsd:enumeration value="PLAYING"/>
                        <xsd:enumeration value="TRANSITIONING"/>
                        <xsd:enumeration value="PAUSED_PLAYBACK"/>
                        <xsd:enumeration value="PAUSED_RECORDING"/>
                        <xsd:enumeration value="RECORDING"/>
                        <xsd:enumeration value="NO_MEDIA_PRESENT"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
        </xsd:complexType>
    </xsd:element>
    <!--============================================================

     TransportStatus

    ============================================================-->
    <xsd:element name="TransportStatus">
        <xsd:complexType>
            <xsd:attribute name="val" use="required">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="OK"/>
                        <xsd:enumeration value="ERROR_OCCURRED"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
        </xsd:complexType>
```

```
</xsd:element>
<!--============================================================

 PlaybackStorageMedium

============================================================-->
<xsd:element name="PlaybackStorageMedium">
    <xsd:complexType>
        <xsd:attribute name="val" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<!--============================================================

 RecordStorageMedium

============================================================-->
<xsd:element name="RecordStorageMedium">
    <xsd:complexType>
        <xsd:attribute name="val" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<!--============================================================

 PossiblePlaybackStorageMedia

============================================================-->
<xsd:element name="PossiblePlaybackStorageMedia">
    <xsd:complexType>
        <xsd:attribute name="val" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<!--============================================================

 PossibleRecordStorageMedia

============================================================-->
<xsd:element name="PossibleRecordStorageMedia">
    <xsd:complexType>
        <xsd:attribute name="val" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<!--============================================================

 CurrentPlayMode

============================================================-->
<xsd:element name="CurrentPlayMode">
    <xsd:complexType>
        <xsd:attribute name="val" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<!--============================================================

 TransportPlaySpeed

============================================================-->
<xsd:element name="TransportPlaySpeed">
    <xsd:complexType>
        <xsd:attribute name="val" type="xsd:string" use="required"/>
    </xsd:complexType>
    </xsd:element>
    <!--================================================================

     RecordMediumWriteStatus

    ================================================================-->
    <xsd:element name="RecordMediumWriteStatus">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     CurrentRecordQualityMode

    ================================================================-->
    <xsd:element name="CurrentRecordQualityMode">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     PossibleRecordQualityMode

    ================================================================-->
    <xsd:element name="PossibleRecordQualityMode">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     NumberOfTracks

    ================================================================-->
    <xsd:element name="NumberOfTracks">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:unsignedInt"
use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     CurrentTrack

    ================================================================-->
    <xsd:element name="CurrentTrack">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:unsignedInt"
use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     CurrentTrackDuration

    ================================================================-->
    <xsd:element name="CurrentTrackDuration">
        <xsd:complexType>
```

```
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     CurrentMediaDuration

    ================================================================-->
    <xsd:element name="CurrentMediaDuration">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     CurrentTrackMetadata

    ================================================================-->
    <xsd:element name="CurrentTrackMetadata">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     CurrentTrackURI

    ================================================================-->
    <xsd:element name="CurrentTrackURI">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     AVTransportURI

    ================================================================-->
    <xsd:element name="AVTransportURI">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     AVTransportURIMetadata

    ================================================================-->
    <xsd:element name="AVTransportURIMetaData">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     NextAVTransportURI

    ================================================================-->
    <xsd:element name="NextAVTransportURI">
        <xsd:complexType>
```

```
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     NextAVTransportURIMetaData

    ================================================================-->
    <xsd:element name="NextAVTransportURIMetaData">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!--================================================================

     CurrentTransportActions

    ================================================================-->
    <xsd:element name="CurrentTransportActions">
        <xsd:complexType>
            <xsd:attribute name="val" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

6. Appendix A – SetAVTransportURI protocol specifics

6.1. Application to HTTP streaming

6.1.1. AVTransportURI definition

URIs are well defined for the HTTP scheme in the Internet standard Request For Comment document entitled Hypertext Connection Protocol – HTTP/1.1 (http://www.ietf.org/rfc/rfc2616.txt).

An example of a content URI for `http-get` streaming, in this case referring to an audio playlist resource, is:

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

6.1.2. Implementation of SetAVTransportURI

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

6.1.3. Cleanup

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

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

6.2. Application to RTSP/RTP/UDP streaming

6.2.1. AVTransportURI definition

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

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

An example is:

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

The scheme `rtsp` requires that commands are issued via a reliable protocol (within the Internet, TCP), while the scheme `rtspu` identifies an unreliable protocol (within the Internet, UDP).

6.2.2. Implementation of SetAVTransportURI

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

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

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

```
R->S: SETUP rtsp://example.com/video/birthday.m2v RTSP/1.0
      CSeq: 30.3
      Transport: RTP/AVP;unicast;client_port=4588-4589

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

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

6.2.3. Cleanup

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

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

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

S->R: RTSP/1.0 200 OK
      CSeq: 892
```

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

6.2.4. Implementation of Transport controls

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

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

S->R: RTSP/1.0 200 OK
      CSeq: 3
      Session: 0456804596
```

Time-based seeking may be supported by an RTSP server. The RTSP client can optionally specify a time-range when issueing a PLAY command, see http://www.ietf.org/rfc/rfc2326.txt for details. Implementation of the AVTransport's Next and Previous commands (changes to a new 'track') will generally require a new RTSP session. URLs for such a new session need to be known by the AVTransport implementation, via, for example, a 'playlist'. This is similar to the HTTP GET case, where an AVTransport can be bound to a playlist URL.

6.3. Application to internal streaming

6.3.1. AVTransportURI definition

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

- The 'scheme' of the URI shall be set to `file`.
- The rest of the URI must follow the rules for specifying legal URIs with this scheme (see http://www.ietf.org/rfc/rfc1738.txt).

An examples of internal AVTransportURIs are:

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

6.3.2. Implementation of SetAVTransportURI

This is device-specific.

6.3.3. Cleanup

This is device-specific.

6.4. Application to IEC61883 streaming

6.4.1. AVTransportURI definition

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

6.4.2. Implementation of SetAVTransportURI

This is device-specific.

6.4.3. Cleanup

This is device-specific.

6.5. Application to vendor-specific streaming

6.5.1. AVTransportURI definition

A AVTransportURI for a vendor-specific protocol is unspecified. The only restrictions are that it must follow the rules for specifying legal URIs (see http://www.ietf.org/rfc/rfc2396.txt).

6.5.2. Implementation of SetAVTransportURI

This is vendor-specific.

6.5.3. Cleanup

This is vendor-specific.