ERICSSON

ABR Playout API

Video Storage and Processing Platform 3.8

INTERWORK DESCR

Contents

Before you Begin

Document Revision History

Document Revision	Software Revision	Description
A	3.8	Release

Important Notice

Document Purpose

The purpose of this document is to explain in detail the various ABR capabilities and URL structures for playout and monitoring. This document will cover:

- VOD playout
- RS-DVR “private-copy” playout
- nPVR “shared-copy” playout
- Rolling buffer playout & monitoring
- MP4 file download

Intended Audience

The intended audiences of this document are:

- Cable/Mobile TV network operators
- Ericsson associated business partners
- Technical support engineers

Comments on this Document

Your comments on this document are of high value in our constant efforts to improve our documentation. Please forward any comments to the local Ericsson Support Office. With your comments please provide the following:
Doc

- Document title
- Document number and revision
- Page number

References

1 Video Storage and Processing Platform 3.7 - Product Description Guide

2 Video Storage and Processing Platform 3.7- Administrators Guide

3 Video Storage and Processing Platform 3.7 - Content Management API document

Acronyms and Terminology

Term	Abbreviation
ABR	Adaptive Bitrate
CBR	Constant bitrate
CMS	Content Management System
CSV	Comma Separated Values
DNS	Domain Name Server
DRM	Digital Rights Management
HDS	HTTP Dynamic Streaming
HLS	HTTP Live Streaming
HSS	HTTP Smooth Streaming
HTTP	Hypertext transfer protocol
JITP	Just-In-Time Packaging
JITX	Just-In Time-Transcoding
JSON	JavaScript Object Notation
JSONP	JavaScript Object Notation with Padding
LSCP	Lightweight Stream Control Protocol
QAM	Quadrature amplitude modulation
RTP	Real time Transport Protocol
RTSP	Real Time Streaming Protocol
STB	Set-Top-Box
VBR	Variable bitrate
VOD	Video On Demand
VSPP	Video Storage and Processing Platform
XML	Extensible Markup Language

1 General Guidelines

Following guidelines, as follows:

- All API calls in this document are consumed via the HTTP protocol.
- Unless stated otherwise all services should be used in conjunction with the HTTP **`GET`** verb only.
- All parameters should be considered mandatory, unless stated otherwise.
- Unless stated otherwise the parameter named manager refers to the manager controller. You can use this value with an InetAddress, or a DNS name pointing to that server. The port to use is 80, unless configured otherwise in the manager configuration file by the system administrator.
- All xs:dateTime are in `UTC`, and should be presented according to this pattern: `YYYY-MM-DD`**`T`**`HH:MM:SS`**`Z`** (E.g 2014-02-06T05:02:00Z).

All API examples in this document are presented with the XSD describing the structure of the XML reply. As with all the VSPP API`s, the JSON standard is available for usage. It is up to the developer to decide how to use our API.

The default RPC format is XML. To use the JSON format you need to add the `Accept:application/json` header when performing the HTTP `GET` request.

1.1 Custom HTTP headers

VSPP exposes a flexible option to pass custom session parameters by the middleware as part of playout session allocation HTTP request.

These parameters may be passed as optional custom HTTP headers, so propagated to the session info and included in the SDR raw data files in order to provide further monitoring/statistic options to the external BI systems.

Please notice for the following:

- This option is exposed only for the session-based, dynamic URL playout mode.
- The max number of these custom HTTP headers is limited to 3 and the names mapping is configured via the Manager.ini file.
- It's expected that these predefined HTTP headers will be provided in all the dynamic playout requests.

1.1.1 Example

The following HTTP GET example provides *broadcastID* and *subsystemID* parameters as custom HTTP headers:

```
GET
/rolling_buffer/CNN/2014:02:15T00:28:00Z/2014:02:15T10:28:00Z/i
phone HTTP/1.1
Host:192.168.5.191
Accept-Encoding:gzip,deflate,sdch
Accept-Language:en-US,en;q=0.8,fr;q=0.6,he;q=0.4,tr;q=0.2
Cache-Control:no-cache
Connection:keep-alive
Pragma:no-cache
User-Agent:Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Ubuntu Chromium/31.0.1650.63
Chrome/31.0.1650.63 Safari/537.36
broadcastID:xg76dgb875
subsystemID:tre-987
```

2 API Section

2.1 Rolling Buffer - Dynamic Mode

Use this service to view Live or Time-Shift content (open and close buffer) while using a Manager Rolling buffer sliding window recording. This service refers to session-based playout, while a dedicated session is allocated by Manager to the specific streamer node to serve the playout request.

Please note that this service provides a 302-Redirect option by concatenating an optional last parameter in the GET request (?redirect=1). When set, Manager will reply the allocated playout URL within the HTTP Location header instead of sending a XML body in order to save time, and parsing on the client side.

2.1.1 Request

URL Format

Method: GET

`http://{manager_address:port}/rolling_buffer/{channel}/{start}/{end}/{device_profile}[/{user_id}][/{nonce}][/{uac_id}][/{signature}]`

Rolling Buffer Dynamic Mode - field description

Field	Type	Description	Mandatory
channel	xs:string	Channel name as defined in the GUI (Channel map).	Yes
start	xs:dateTime	Request start time (1)	Yes
end	xs:dateTime	Request end time (2)	Yes
device_profile	xs:string	The name of the device profile attribute to use. The type of the ABR protocol for playout will be derived from the device profile definitions.	Yes

Field	Type	Description	Mandatory
user-id	xs:string	User identifier (ID) requesting the playout service, used for monitoring and reporting.	Optional. In case ‘user-id’ is not provided, but one of the following security attributes (nonce, uac-id) is required, the literal “none” should be used.
nonce	xs:integer	A non-repeating decimal number. Once the NONCE was used any attempt to re-use it within a week will fail (403 Forbidden will be returned).	Optional. In case ‘nonce’ is not provided, but ‘uac-id’ is required, the ‘0’ should be used
uac_id	xs:string	An identifier for user access control. Repeating this UAC during another playout of the same UAC will cause an automatic tear down of the first playout. The new playout will proceed.	Optional.
signature	xs:string	Encryption of URI (4)	Optional. When enabled in the system configuration, signature is always the last parameter in the URL.

Notes:

- Start time can be set to the past time for time-shift service or accept the string literal `‘LIVE’` to view the Live streaming of the channel.
- End time can be set to the past or future time, or to the string literal `‘END’` (this will result in live streaming indefinitely).
- Debug device profiles are available (all available ABR layers will be published in the manifest):
  - HLS
  - HLS_ENC
  - SMOOTH
  - SMOOTH_ENC
  - DASH

Request Examples:

Rolling buffer playout request - LIVE

`http://192.168.5.228/rolling_buffer/CNN/LIVE/END/iphone`

Rolling buffer playout request - Time range

`http://192.168.6.191/rolling_buffer/CNN/2014-02-15T00:28:00Z/2014-02-15T10:28:00Z/iphone`

Non reusable (user id & nonce) rolling buffer playout request - Time range

`http://192.168.6.191/rolling_buffer/CNN/2014-02-15T00:28:00Z/2014-02-15T10:28:00Z/luma/663425/24/6a0c1c9e6ed3df49a25084afa030f818`

Rolling buffer playout request - LIVE (with 302-redirect)

`http://192.168.5.228/rolling_buffer/CNN/LIVE/END/iphone?redirect=1`

2.1.2 Reply

When the Manager fails to create a playout URL, the result will be in the form of an HTTP return code (4xx or 500) with no body.

HTTP Code	Type	Description
200	OK	Session is allocated successfully
302	REDIRECT	Redirect
400	BAD_REQUEST	Missing mandatory parameters; Device profile not found; Channel not found
403	FORBIDDEN	Signature is invalid; Reused dispensable URL (user id & nonce)
500	INTERNAL_SERVER_ERROR	Unexpected internal error

Rolling_buffer_allocation_results - field description

Field	Type	Description
url	anyURL	The playout URL - send a GET request to this URL for the manifest file. **Dynamic:** `http://{streamer:port}/{abr-type}/{session-id}.{manifest-suffix}`
channel_status	Node	Segments available for playout for a single channel

URL element - field description

Field	Type	Description
address	xs:string	The InetAddress or DNS name of the streamer node
abr-type	xs:string	The type of HTTP streaming protocol selected 1. SMOOTH: `http_smooth_streaming` 2. HLS: `http_adaptive_streaming` 3. DASH: `dash`
session-id	xs:string	Session identifier allocated for the dynamic session
manifest-suffix	xs:string	1. SMOOTH: *.ism/Manifest 2. HLS: *.m3u8 3. DASH: *.mpd/Manifest

Channel_status element - field description

Field	Type	Description
@name	xs:string	Channel name
inner_buffer	Node	Collection of inner_buffer node with status

Inner_buffer Element - Field Description

Field	Type	Description
@start_time	xs:dateTime	The start time of the first rolling buffer segment in the requested range.
@end_time	xs:dateTime	The end time of the last rolling buffer segment in the requested range.
@playable	xs:string	The indexes of the ABR layers that are playable in CSV format.
@no_data	xs:string	The indexes of the ABR layers that no data is available for in CSV format.
@out_of_sync	xs:string	The index of the ABR layers that are out of sync in CSV format.
@erroneous	xs:string	The indexes of the ABR layers that are corrupted in CSV format.

Notes:

1 Sometimes a single ABR layer is not available (due to encoder issues, no actual feed, etc), while the rest of the layers are.

2 For the rolling buffer allocation result XSD, please refer to Appendix C - XSD schemas.

Response Example

```
<rolling_buffer_allocation_result>
<url>http://strm.Pod1:5555/http_smooth_streaming/IGAAAAAALFGHOBPE.ism/Manifest
</url>
<channel_status name="CNN">
<inner_buffer start_time="2012-01-24T08:55:00Z" end_time="2012-01-24T09:25:24Z"  playable="2,3,4" no_data="0,1"/>
</channel_status>
</rolling_buffer_allocation_result>
```

2.2 Rolling Buffer - Static URL

Use this service to view Live or Time-Shift content (open and close buffer) while using a Manager Rolling buffer sliding window recording. This method is different from the once described above and uses the static playout URL.

This mode is used when a CDN is part of the operator ecosystem and the CDN friendly and cacheable static playout URL structure (w/o session ID) is used.

URL Format

Method: GET

```
http://{strm.POD_name:port}/{static-abr-type}/{live-channel-identifier}/{fragment-length}.{manifest-suffix}?start={start}&end={end}&device={device}&client_version={client-version}
```

2.2.1 Request

<rolling_buffer_static> Field Description

Field	Type	Description	Mandatory
static-abr-type	xs:string	The type of HTTP streaming protocol selected 1. SMOOTH: `shss` 2. HLS: `shls` 3. DASH: `sdash`	Yes
live-channel-identifier	xs:string	The literal `'LIVE$'` should be prepend to the channel name. This channel name should be configured in the Channel Map per live channel.	Yes
fragment-length	xs:integer	The fragments length in seconds. This value, **must** match the value set in the device profile that is requested for the playout.	Yes
manifest-suffix	xs:string	Manifest specific suffix: 1. SMOOTH: *.ism/Manifest 2. HLS: *. m3u8 3. DASH: *.mpd/Manifest	Yes
start	xs:dateTime	Requested start time (1)	Yes
end	xs:dateTime	Requested end time (2)	Yes

Field	Type	Description	Mandatory
device	xs:string	The name of the device profile attribute to use. The type of the ABR protocol for playout will be derived from the device profile definitions.	Yes
client-version	xs:string	The ‘client-version’ can be explicitly provided to help the streamer node, respond with the best possible streaming properties (4).	Optional

Notes:

1 Start time can be set to the past time for time-shift service or accept the string literal ‘LIVE’ to view the Live streaming of the channel.

2 End time can be set to the past or future time, or to the string literal `‘END’` (this will result in live streaming indefinitely).

3 Debug device profiles are available (all available ABR layers will be published in the manifest):
- HLS
- HLS_ENC
- SMOOTH
- SMOOTH_ENC
- DASH

4 Different devices support different versions of the same protocol. Adding this information will help the streaming node responds with the best suited protocol version for that particular device.
- HLS
  - Accepted values are (HLS protocol versions): 3, 4, 5
  - Client user-agent
  - Values that are not recognized (see bullet 4.a.i), will be ignored and a configured default value will be selected.
- SMOOTH
  - This value has no meaning, and will be ignored if sent

Static URL Examples:

HLS

```
http://strm.POD1:5555/shls/LIVE$CNN/5.m3u8?start=2011-10-
05T08:00:00Z&end=2011-10-
05T09:00:00Z&device=iphone&client_version=5
```

SMOOTH

```
http://strm.POD1:5555/shss/LIVE$CNN/5.ism/Manifest?start=2011-
10-05T08:00:00Z&end=2011-10-05T09:00:00Z&device=webtv
```

DASH

```
http://strm.POD1:5555/sdash/LIVE$CNN/5.mpd/Manifest?start=2011-
10-05T08:00:00Z&end=2011-10-05T09:00:00Z&device=chrome
```

2.3 RS-DVR Playout Request

Use this command to request a playout for the RS-DVR private-copy recording.

RS-DVR playout always uses a session-based dynamic playout URL mode, so each playout request will hit the Manager for session allocation on the specific streamer node.

Please note that this service provides a 302-Redirect option by concatenating an optional last parameter in the GET request (?redirect=1). When set, Manager will reply the allocated playout URL within the HTTP Location header instead of sending a XML body in order to save time, and parsing on the client side.

2.3.1 Request

The request is described below.

URL Format

Method: GET

`http://{manager-address:port}/v2/abr_dvr/{`**`home-id`**`}/{`**`showing-id`**`}/{`**`device-profile`**`}[/{user-id}][/{nonce}[/{signature}]`

RS-DVR Playout Request - Field Description

Field	Type	Description	Mandatory
home-id	xs:string	The ID of the provisioned subscriber DVR box for which this recording belongs.	Yes
showing-id	xs:string	The external content identifier as provided by the external back-office as part of the RS-DVR private-copy recording request	Yes
device-profile	xs:string	The name of the device profile to use for this playout. The type of the ABR protocol for playout will be derived from the device profile definitions.	Yes

Field	Type	Description	Mandatory
user-id	xs:string	User identifier (ID) requesting the playout service, used for monitoring and reporting.	Optional. In case ‘user-id’ is not provided, but nonce is required, the literal “none” should be used.
nonce	xs:integer	A non-repeating decimal number. Once the NONCE was used any attempt to re-use it within a week will fail (403 Forbidden will be returned).	Optional
signature	xs:string	Encryption of URI.	Optional. When enabled in the system configuration, signature is always the last parameter in the URL.

Notes:

Debug device profiles are available (all available ABR layers will be published in the manifest):

- HLS
- HLS_ENC
- SMOOTH
- SMOOTH_ENC
- DASH

Request Example:

```
http://192.168.5.228/v2/abr_dvr/1111/eb4-203f0/ipad/
```

2.3.2 Reply

When the manager controller fails to create a playout URL, the result will be in the form of an HTTP return code (4xx or 500) with no body.

HTTP Code	Type	Description
200	OK	Session allocated successfully.
302	REDIRECT	Redirect

HTTP Code	Type	Description
400	BAD_REQUEST	Missing mandatory parameters; Device profile not found; asset ID not found.
403	FORBIDDEN	Signature is invalid; Reused dispensable URL (user id & nonce).
404	NOT_FOUND	Device profile/Home ID not found
500	INTERNAL_SERVER_ERROR	Unexpected internal error.

RS-DVR Playout Allocation Results - Field Description

Field	Type	Description
url	anyURI	The playout URL to propagate back to the video client.
session_id	xs:string	The session identifier.
bitrate	xs:long	The average bitrate (in bps) of the highest ABR layer published in the manifest

Note: For the abr_dvr XSD schema, please refer to Appendix C - XSD schemas.

Response Examples:

HLS Streaming:

```
<allocation_result>
 <url>

http://strm.Pod1:55555/http_adaptive_streaming/DAAAAAAAFJAGJIOE
.m3u8
 </url>
 <session_id>DAAAAAAAFJAGJIOE</session_id>
 <bitrate>1572864</bitrate>
</allocation_result>
```

SMOOTH Streaming:

```
<allocation_result>
 <url>

http://strm.Pod1:5555/http_smooth_streaming/IGAAAAAALFGHOBPE.is
m/Manifest
 </url>
 <session_id>IGAAAAAALFGHOBPE</session_id>
 <bitrate>1572864</bitrate>
</allocation_result›
```

DASH streaming:

```
<allocation_result>
 <url>
http://strm.Pod1:5555/dash/FFAAAAAAHLICCPCF.mpd/Manifest
 </url>
 <session_id>FFAAAAAAHLICCPCF</session_id>
 <bitrate>1572864</bitrate>
</allocation_result>
```

2.4 VOD Playout - Static URL

Use this service to playout a VOD asset by using a static URL mode.
This mode is used when a CDN is part of the operator ecosystem and the CDN friendly and cacheable static playout URL structure (w/o session ID) is used.

2.4.1 Request

The request is described below.

URL Format

Method GET

HLS

```
http://{strm.Pod_name:port}/shls/{asset-id}/{fragment-length}.m3u8?device={device-profile}&client_version={client-version}
```

SMOOTH

```
http://{strm.Pod_name:port}/shss/{asset-id}/{fragment-length}.ism/Manifest?device={device-profile}
```

DASH

```
http://{strm.Pod_name:port}/sdash/{asset-id}/{fragment-length}.mpd/Manifest?device={device-profile}
```

VOD playout static URL - field description

Field	Type	Description	Mandatory
asset-id	xs:string	The external VOD asset ID given as part of the ingest request.	Yes
device-profile	xs:string	The name of the device profile that requests the playout.	Yes
fragment-length	xs:integer	The fragments length in seconds. This value, **must** match the value set in the device profile that is requested for the playout.	Yes

Field	Type	Description	Mandatory
client-version	xs:string	The ‘client-version’ can be explicitly provided to help the streamer node, respond with the best possible HLS streaming properties	Optional

Note:

Debug device profiles are available (all available ABR layers will be published in the manifest):

- HLS
- HLS_ENC
- SMOOTH
- SMOOTH_ENC
- DASH

2.4.2 Reply

HTTP Code	Type	Description
200	OK	Success
400	BAD_REQUEST	Not enough parameters in the request.
404	NOT_FOUND	Asset not found.
500	INTERNAL_SERVER_ERROR	Unexpected internal server error

2.5 VOD Playout - Dynamic URL

Use this service to view a VOD asset by using a dynamic (session-based) playout URL mode.

This service may be used to request progressive download of MP4 pre-packaged VOD files as well, by using a special device profile from ‘download’ type.

2.5.1 Request

The request is described below.

URL Format:

Method: GET

```
http://{manager-address:port}/abr_vod/{asset-id}/{device-profile}[{user-id}][/{nonce}][/{uac-id}][/{signature}]
```

VOD playout dynamic URL – Field Description

Field	Type	Description	Mandatory
asset-id	xs:string	The external VOD asset ID given as part of the ingest request	Yes
device-profile	xs:string	The name of the device profile. The type of the ABR protocol for playout will be derived from the device profile definitions. For the progressive download of pre-packaged VOD file use a special device profile from ‘download’ type.	Yes
user-id	xs:string	User identifier (ID) requesting the playout service, used for monitoring and reporting.	Optional. In case ‘user-id’ is not provided, but one of the following security attributes (nonce, uac-id) is required, the literal “none” should be used.

Field	Type	Description	Mandatory
nonce	xs:integer	A non-repeating decimal number. Once the NONCE was used any attempt to re-use it within a week will fail (403 Forbidden will be returned).	Optional. In case ‘nonce’ is not provided, but ‘uac-id’ is required, the ‘0’ should be used
uac_id	xs:string	An identifier for user access control. Repeating this UAC during another playout of the same UAC will cause an automatic tear down of the first playout. The new playout will proceed.	Optional.
signature	xs:string	Encryption of URI	Optional. When enabled in the system configuration, signature is always the last parameter in the URL.

VOD playout request - Trusted Source

```
http://192.168.6.191/abr_vod/d34d44c8-96c41ddb0ca5/iphone
```

VOD playout request - Untrusted Source (user id & nonce & uac id)

```
http://192.168.6.191/abr_vod/d34d44c8-
96c41ddb0ca5/iphone/UXXX12345/663425/24/6a0c1c9e6ed3df49a25084a
fa030f818
```

VOD MP4 progressive download request - Trusted Source

```
http://192.168.6.191/abr_vod/d34d44c8-96c41ddb0ca5/download
```

2.5.2 Reply

When the manager controller fails to create a playout URL, the result will be in the form of an HTTP return code (4xx or 500) with no body.

HTTP Code	Type	Description
200	OK	Session allocated successfully.
302	REDIRECT	Redirect
400	BAD_REQUEST	Missing mandatory parameters; Device profile not found; VOD asset ID not found.
403	FORBIDDEN	Signature is invalid; Reused dispensable URL (user id & nonce).
500	INTERNAL_SERVER_ERROR	Unexpected internal error.

VOD Playout URL Reply – Field Description

Field	Type	Description
url	anyURI	The playout URL to propagate back to the video client.
session_id	xs:string	The session identifier.

Notes:

- Debug_device profiles are available (all available ABR layers will be published in the manifest):
  - HLS
  - HLS_ENC
  - SMOOTH
  - SMOOTH_ENC
  - DASH

- For playout of prepackaged VOD content please use the predefined device profiles (ISM; FDASH)

Note: For the `abr_vod` reply XSD schema, please refer to the Appendix C- XSD schemas

Reply Examples

HLS streaming:

```
<allocation_result>
 <url>
http://strm.Pod1:5555/http_adaptive_streaming/DAAAAAAAFJAGJIOE.
m3u8
 </url>
 <session_id>DAAAAAAAFJAGJIOE</session_id>
</allocation_result>
```

SMOOTH streaming:

```
<allocation_result>
 <url>
http://strm.Pod1:5555/http_smooth_streaming/IGAAAAAALFGHOBPE.is
m/Manifest
 </url>
 <session_id>IGAAAAAALFGHOBPE</session_id>
</allocation_result>
```

DASH streaming:

```
<allocation_result>
 <url>
    http://strm.Pod1:5555/dash/FFAAAAAAHLICCPCF.mpd/Manifest
 </url>
 <session_id>FFAAAAAAHLICCPCF</session_id>
</allocation_result>
```

VOD MP4 progressive download:

```
<allocation_result>
<url>
    http://strm.Pod1:5555/download?session_id=DINBAAAAPGILLLEF
</url>
<session_id>DINBAAAAPGILLLEF</session_id>
</allocation_result>
```

2.6 VOD Pre-packaged Playout (static URL)

Use this service to playout OTT pre-packaged VOD assets by using a static playout URL mode.

The VOD pre-packaged/pre-encrypted assets are ingested on the storage with no content manipulation and played out as-is, without using on the fly packaging - JITP.

2.6.1 Request

The request is described below:

URL Format

Method GET

HLS
`http://{strm.Pod_name:port}/sfhss/{assetID}.m3u8`

SMOOTH
`http://{strm.Pod_name:port}/sfhss/{assetID}.ism/Manifest`

DASH
`http://{strm.Pod_name:port}/sfdash/{assetID}.mpd/Manifest`

VOD playout static URL - Field Description

Field	Type	Description	Mandatory
asset-id	xs:string	The external VOD asset ID given as part of the ingest request.	Yes

2.6.2 Reply

HTTP Code	Type	Description
200	OK	Success
400	BAD_REQUEST	Not enough parameters in the request.
404	NOT_FOUND	Asset not found.
500	INTERNAL_SERVER_ERROR	Unexpected internal server error

2.7 nPVR playout - static URL

Use this service to request playout of shared-copy nPVR asset by using a static URL.

This mode is used when a CDN is part of the operator ecosystem and the CDN friendly and cacheable static playout URL structure (w/o session ID) is used.

This service is supporting the option to playout a global shared-copy recorded asset using static URL with respect to start/end time offsets based on the particular subscriber pre/post padding times.

2.7.1 Request

The request is described below:

URL Format

Method GET

HLS:
```
http://{strm.Pod_name:port}/shls/{asset-id}/{fragment-length}.m3u8?start={start-offset}&end={end-offset}&device={device-profile}&client_version={client-version}
```

SMOOTH
```
http://{strm.Pod_name:port}/shss/{asset-id}/{fragment-length}.ism/Manifest?start={start-offset}&end={end-offset}&device={device-profile}
```

DASH
```
http://{strm.Pod_name:port}/sdash/{asset-id}/{fragment-length}.mpd/Manifest?start={start-offset}&end={end-offset}&device={device-profile}
```

nPVR playout static URL - field description

Field	Type	Description	Mandatory
asset-id	xs:string	The external asset ID assigned by Manager as part of the recording request. For the playout request the 'asset-id' attribute should be constructed as : ***{showing-id}_abr*** string Where 'showing-id' is the external content identifier provided as part of recording request.	Yes

Field	Type	Description	Mandatory
device-profile	xs:string	The name of the device profile that requests the playout.	Yes
fragment-length	xs:integer	The fragments length in seconds. This value, **must** match the value set in the device profile that is requested for the playout.	Yes
start-offset	xs:integer	The start npt offset (in seconds) relative to the zero offset of the global shared-copy asset.	No
end-offset	xs:integer	The end npt offset (in seconds) relative to the zero offset of the global shared-copy asset.	No
client-version	xs:string	The ‘client-version’ can be explicitly provided to help the streamer node, respond with the best possible HLS streaming properties.	No

Note:
Debug device profiles are available (all available ABR layers will be published in the manifest):HLS

- HLS_ENC
- SMOOTH
- SMOOTH_ENC
- DASH

2.7.2 Reply

HTTP Code	Type	Description
200	OK	Success
400	BAD_REQUEST	Not enough parameters in the request.
404	NOT_FOUND	Asset not found.
500	INTERNAL_SERVER_ERROR	Unexpected internal server error

Examples:

HLS request:

```
http://strm.Pod1:80/shls/{ert-567_abr}/5.m3u8?device=iPhone
```

DASH request (with offsets):

```
http://origin.server.com:80/sdash/{rt5456_abr}/2.mpd/Manifest?s
tart=600&end=3500&device=OTT_DASH_CENC
```

2.8 nPVR playout – dynamic URL

Use this service to request playout of the shared-copy nPVR asset by using a dynamic (session-based) playout URL mode.

This service provides an option to playout a global “shared-copy” recorded asset with respect to start/end time offsets based on the particular subscriber recording request time boundaries.

Please note that this service provides a 302-Redirect option by concatenating an optional last parameter in the GET request (?redirect=1). When set, the Manager will reply the allocated playout URL within the HTTP Location header instead of sending a XML body in order to save time, and parsing on the client side.

2.8.1 Request

The request is described below

URL Format

Method: GET

```
http://{manager-address:port}/v2/npvr/{showing-id}/{start-offset}/{end-offset}/{device-profile}
```

nPVR playout dynamic URL – field description

Field	Type	Description	Mandatory
showing-id	xs:string	The external content identifier as provided in the recording request	Yes
start-offset	xs:integer	The start npt offset (in seconds) relative to the zero offset of the global shared-copy asset. For playout from the beginning of the global shared-copy asset use ‘0’ literal	Yes
end-offset	xs:integer	The end npt offset (in seconds) relative to the zero offset of the asset. For playout till the end of the global shared-copy asset use ‘END’ literal	Yes

Field	Type	Description	Mandatory
device-profile	xs:string	The name of the device profile. The type of the ABR protocol for playout will be derived from the device profile definitions.	Yes

nPVR playout request (entire shared-copy asset):

`http://192.168.6.191/v2/npvr/d34d44c8/0/END/iphone`

nPVR playout request (with specific offsets):

`http://192.168.6.191/v2/npvr/d34d44c8/300/3600/iphone`

2.8.2 Reply

When the manager controller fails to create a playout URL, the result will be in the form of an HTTP return code (4xx or 500) with no body.

HTTP Code	Type	Description
200	OK	Session allocated successfully.
302	REDIRECT	Redirect
400	BAD_REQUEST	Missing mandatory parameters; Device profile not found; VOD asset ID not found.
403	FORBIDDEN	Signature is invalid; Reused dispensable URL (user id & nonce).
500	INTERNAL_SERVER_ERROR	Unexpected internal error.

nPVR playout URL reply – field description

Field	Type	Description
url	anyURI	The playout URL to propagate back to the video client.
session_id	xs:string	The session identifier.

Notes:
Debug device profiles are available (all available ABR layers will be published in the manifest):

a. HLS
b. HLS_ENC
c. SMOOTH
d. SMOOTH_ENC
e. DASH

For the `abr_vod` reply XSD schema, please refer to the Appendix C- XSD schemas

Reply Examples:

HLS streaming:

```
<allocation_result>
 <url>
http://strm.Pod1:5555/http_adaptive_streaming/DAAAAAAAFJAGJIOE.
m3u8
 </url>
 <session_id>DAAAAAAAFJAGJIOE</session_id>
</allocation_result>
```

SMOOTH streaming:

```
<allocation_result>
 <url>
http://strm.Pod1:5555/http_smooth_streaming/IGAAAAAALFGHOBPE.is
m/Manifest
 </url>
 <session_id>IGAAAAAALFGHOBPE</session_id>
</allocation_result>
```

DASH streaming:

```
<allocation_result>
 <url>
    http://strm.Pod1:5555/dash/FFAAAAAAHLICCPCF.mpd/Manifest
 </url>
 <session_id>FFAAAAAAHLICCPCF</session_id>
</allocation_result>
```

2.9 MP4 Download Request

Request to download a section of the rolling buffer in MP4 file format.

Once this request is sent, it will trigger an internal process that will package the requested rolling buffer time frame as an MP4 file. Since the packaging process may take some time (depending on system load, file size etc), the client will need to continuously send the same exact request, and monitor the `progress_percentage` value. Once the packaging is completed and this value reaches to a 100, the download URL will be indicated in the response (during the packaging process the URL is not available). This download URL needs to be propagated back to the client for progressive download.

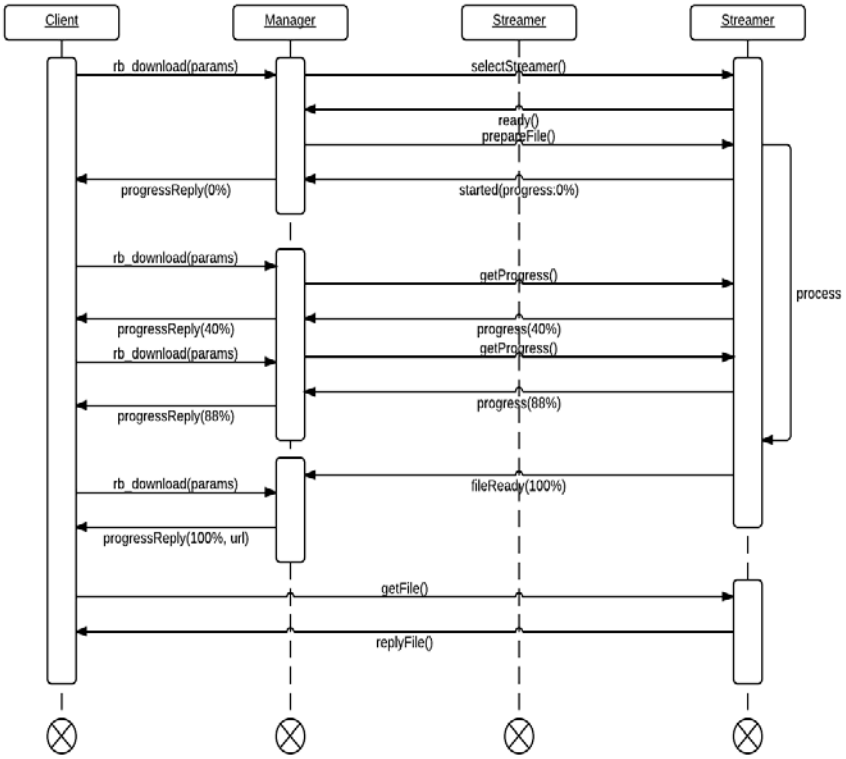

Figure 1: mp4 file download

2.9.1 Request

URL Format

Method: GET

```
http://{manager-
address:port}/rb_download/{channel}/{start}/{end}/{device-
profile}/[signature]
```

MP4 download - Field Description

Field	Type	Description	Mandatory
channel	xs:string	The channel name as defined in the GUI management application.	Yes
start	xs:dateTime	Start range for the download section.	Yes
end	xs:dateTime	End range for the download section.	Yes
device-profile	xs:string	The name of the device profile. This profile needs to be set with the “MP4 download” package type.	Yes
signature	xs:string	Encryption of URI.	Optional

Note:

For the encryption related issues, please refer to the encryption appendix.

Request Example
`http://192.168.100.127/rb_download/CNN/2012-10-05T08:00:00Z/2012-10-05T09:00:00Z/mp4`

2.9.2 Reply

When the manager controller fails to initiate the preparation progress the result will be in the form of an HTTP return code (4xx or 500), and no body.

HTTP Code	Type	Description
200	OK	MP4 file created successfully.
400	BAD_REQUEST	Missing mandatory parameters; Channel not found; Device profile not found; Failed resolving the playlist because the start and end time are not within buffer limits.
403	FORBIDDEN	Signature is invalid.
500	INTERNAL_SERVER_ ERROR	Unexpected internal error.

MP4 download reply - field description

Field	Type	Description
url	anyURI	The download URL. This attribute will return the result only when the file is ready for download.
progress_percentage	xs:integer	File preparation progress.
channel_status	Node	Segments available for playout for a single channel.

Channel_status element field description

Field	Type	Description
@name	xs:string	Channel name.
inner_buffer	Node	Collection of inner_buffer node with status.

Inner_buffer element field description

Field	Type	Description
@start_time	xs:dateTime	The start time of the first rolling buffer segment in the requested range.
@end_time	xs:dateTime	The end time of the last rolling buffer segment in the requested range.
@playable	xs:string	The indexes of the ABR layers that are playable, in CSV format.
@no_data	xs:string	The indexes of the ABR layers that no data is available for, in CSV format.
@out_of_sync	xs:string	The index of the ABR layers that are out of sync in CSV format.
@erroneous	xs:string	The indexes of the ABR layers that are corrupted in CSV format.

Notes:

For the `mp4_download` XSD schema, please refer to the Appendix C - XSD schemas

Reply Examples:

Reply for the file in preparation:

```
<mp4_download_result>
<url>
<url/>
<session_id>0</session_id>
 <channel_status name="CNN" bitrate_type="ABR">
 <inner_buffer start_time="2012-10-05T08:00:00Z"
     end_time="2012-05-10T09:00:00Z" playable="3"/>
  </channel_status>
<progress_percentage>60</progress_percentage>
</mp4_download_result>
```

Reply for the ready to download file:

```
<mp4_download_result>
<url>http://192.168.9.228:80/mp4_download_file?string_id=
NIDAAAAAGIFOOMPE&external_asset_id=</url>
<session_id>NIDAAAAAGIFOOMPE</session_id>
<channel_status name="CNN" bitrate_type="ABR">
  <inner_buffer start_time="2012-06-06T05:02:00Z"
   end_time="2012-06-06T05:04:00Z" playable="3"/>
 </channel_status>
</mp4_download_result>
```

2.10 Session Destroy

Use this service to explicitly tear down an active session. If a session is paused/idle, meaning that the video client did not request additional fragments during the configured allotted time (def: 5 min), the automatic tear down will be initiated internally.

This service is relevant only if the session was created in the dynamic mode.

This is the only service described in this document where the URI port is the port assigned to the management application. By default this value is 5929.

2.10.1 Request

The request is described below**.**

URL format

Method: GET

`http://{manager-address:5929}/mng_session_teardown?id={session-id}`

Session destroy – field description

Field	Type	Description	Mandatory
id	xs:string	Session ID	Yes

Request Example:

`http://192.168.5.228:5929/mng_session_teardown?id=FGBAAAAACFEGCPCF`

2.10.2 Reply

HTTP Code	Type	Description
200	OK	Success
400	BAD REQUEST	Bad Request
500	INTERNAL_SERVER_ERROR	Unexpected internal error

Reply Examples:

Existing session:

```
<?xml version="1.0" encoding="UTF-8"?>
<X>
<code>0</code>
<reply>
<success>1</success>
<params>
<scale_numerator>1</scale_numerator>
<client_ip>192.168.100.204:57731</client_ip>
<client_user_agent>GStreamer souphttpsrc
libsoup/2.44.2</client_user_agent>
</params>
</reply>
<torn_down>1</torn_down>
</X>
```

Unknown session:

```
<?xml version="1.0" encoding="UTF-8"?>
<X>
<code>-17014</code>
<description>Teardown request for unknown session
(PBAAAAAAJJOJCKEF)</description>
<reply>
<success>1</success>
<params></params>
</reply>
<torn_down>0</torn_down>
</X>
```

2.11 Rolling Buffer Status

This service specifies how to check that the rolling buffer content is playable (or partially playable) before allocating a payout session.

2.11.1 Request

URL format

Method: GET

```
http://{manager-address:port}/status_rolling_buffer/{channel}/{start}/{end}
```

Rolling buffer status – Field Description

Field	Type	Description	Mandatory
channel	xs:string	Channel name as defined in the Video Storage & Processing platform GUI (Channel map).	Yes
start	xs:dateTime	Requested start time.	Yes
end	xs:dateTime	Requested end time.	Yes

Request Example:

```
http://192.168.6.191/status_rolling_buffer/CNN/2014-01-06T17:30:00/2014-01-06T18:10:00
```

2.11.2 Reply

HTTP Code	Type	Description
200	OK	Success.
400	BAD_REQUEST	Missing mandatory parameters; Channel not found; Failed resolving the playlist because the start and end time are not within buffer limits.
500	INTERNAL_SERVER_ERROR	Unexpected internal error.

Rolling buffer status reply – field description

Field	Type	Description
Channel	List <xs:string>	Channel parameters.
inner_buffer	Node	Collection of inner_buffer node with status, according the requested time range.

Channel element - field description

Field	Type	Description
name	xs:string	Channel name.
bitrate_type	xs:string	The type of the inner assets (values can be ABR | CBR).

Inner_buffer element - field description

Field	Type	Description
@start_time	xs:dateTime	The start time of the first rolling buffer segment in the requested range.
@end_time	xs:dateTime	The end time of the last rolling buffer segment in the requested range.
@playable	xs:string	The indexes of the ABR layers that are playable in CSV format.
@no_data	xs:string	The indexes of the ABR layers that no data is available for in CSV format.
@out_of_sync	xs:string	The index of the ABR layers that are out of sync in CSV format.
@erroneous	xs:string	The indexes of the ABR layers that are corrupted in CSV format.

Reply Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rolling_buffer_status>
<channel name="CNN" bitrate_type="ABR"/>
<inner_buffer
start_time="2014-01-06T17:18:33Z" end_time="2014-01-
06T18:21:54Z"
playable="0,3,4"
erroneous="1,2,5"/>
</rolling_buffer_status>
```

2.12 Mixed Rolling Buffer Status

This service specifies how to check that the rolling buffer content is playable (or partially playable) before allocating a playout session. The difference between this service and the previous one is that the first one will return values only for **ABR buffers**, while this one will hold values and statues for **CBR buffers** as well.

2.12.1 Request

URL format

Method: GET

```
http://{manager_address:port}/mixed_status_rolling_buffer/{channel}/{start}/{end}
```

Mixed status rolling buffer - field description

Field	Type	Description	Mandatory
channel	xs:string	Channel name as defined in the Video Storage & Processing platform GUI (Channel map).	Yes
start	xs:dateTime	Requested start time.	Yes
end	xs:dateTime	Requested end time.	Yes

Request Example:

```
http://192.168.6.191/mixed_status_rolling_buffer/CNN/2014-01-06T17:30:00/2014-01-06T18:10:00
```

2.12.2 Reply

Mixed rolling buffer status reply - field description

Field	Type	Description
Channel	List <xs:string>	Channel parameters.
inner_buffer	Node	Collection of inner_buffer node with status, according the requested time range.

Channel element - field description

Field	Type	Description
name	xs:string	Channel name.
bitrate_type	xs:string	The type of the inner assets (values can be ABR | CBR).

Inner_buffer (ABR) element - field description

Field	Type	Description
@start_time	xs: xs:dateTime	The start time of the first rolling buffer segment in the requested range
@end_time	xs: xs:dateTime	The end time of the last rolling buffer segment in the requested range.
@playable	xs:string	The indexes of the ABR layers that are playable in CSV format
@no_data	xs:string	The indexes of the ABR layers that no data is available for in CSV format.
@out_of_sync	xs:string	The index of the ABR layers that are out of sync in CSV format.
@erroneous	xs:string	The indexes of the ABR layers that are corrupted in CSV format.

Inner_buffer (CBR) element - field description

Field	Type	Description
@start_time	xs:dateTime	The start time of the first rolling buffer segment in the requested range.
@end_time	xs:dateTime	The end time of the last rolling buffer segment in the requested range.
@playable	xs:boolean	True if the rolling buffer segments in the requested range is playable
@no_data	xs:boolean	True if no data is available for the rolling buffer segments in the requested range.
@out_of_sync	xs:boolean	True if the rolling buffer segments in the requested range are out of sync.
@erroneous	xs:boolean	True if the rolling buffer segments in the requested range are erroneous.

Reply Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mixed_rolling_buffer_status>
<channel name="FOX" bitrate_type="ABR"/>
<inner_buffer
start_time="2014-01-06T17:18:33Z" end_time="2014-01-
06T18:21:54Z"
playable="0,3,4"
erroneous="1,2,5"/>
<channel name="FOX" bitrate_type="CBR"/>
<inner_buffer
start_time="2014-01-06T17:18:33Z" end_time="2014-01-
06T18:21:54Z"
playable="true"/>
</mixed_rolling_buffer_status>
```

Appendix A - Signature

The signature is a long uninterrupted string of alphanumeric characters that comes after the last slash. The signed signature is the URI, from its beginning and up until the last slash before the signature (not including the last slash).

Example:

```
/rolling_buffer/SF1/LIVE/END/ipad/6a0c1c9e6ed3df49a25084afa030f
818
```

Signature Signing

The signature signing process is as follows:

- MD5 the URI. The result is 16 Bytes string.
- Encrypt the result using a key (shared by the portal and manager) and an IV. The encryption used is AES 256-bit CBC encryption.
- Add a slash and the encrypted 16 bytes (HEX) to the URL.

Signature Validation

Signature validation is done by following the steps listed in Signature Signing process (above), then comparing the result with signature provided by the URL.

Generate Signature via Python Script

```
#!/usr/bin/python

import sys
import aes
import md5

KEY_FILE = 'enc.key'

def sign_url(url):
   key_bin = ''
   try:
           keyfile = file (KEY_FILE,'rb')
           key_bin = keyfile.read()
           keyfile.close()
   except:
           print "Unable to open key file", KEY_FILE
           print
           sys.exit(1)

   if len(key_bin) != 32:
```

```
            print "Expected 32 bytes in file", KEY_FILE, ", only
found", len(key_bin)
            print
            sys.exit(1)

    key = [ord(i) for i in key_bin]

    if url[-1] == '/':
            url = url[:-1]
    slash_slash_loc = url.find('//')
    if slash_slash_loc == -1:
            print "Unable to find // in given url"
            print
            sys.exit(1)
    slash_loc = url.find('/', slash_slash_loc + 2)
    if slash_loc == -1:
            print "Unable to find beginning of URI in given URL"
            print
            sys.exit(1)
    uri = url[slash_loc:]
    md5_hash = md5.new(uri)
    hash_res = md5_hash.digest ()

    moo = aes.AESModeOfOperation()
    iv = [154, 213, 34, 67, 49, 24, 240, 78, 245, 254, 9, 60, 57,
181, 80, 28]
    _, _, ciph = moo.encrypt(hash_res,
moo.modeOfOperation["CBC"],
     key, moo.aes.keySize["SIZE_256"], iv)
    return url + "/" + ''.join([hex(i)[2:] for i in ciph])

if __name__ == "__main__":
    if len(sys.argv) != 2:
            print "Expected parameter: URL to sign"
            print
            sys.exit (1)
    print sign_url(sys.argv[1])
```

Appendix B - Manifests

In this section the examples provided are intended for educational purposes only. The first step is the same for all the protocols, where you request a playout URL, and send this URL back to the appropriate video player. What happens next is under the responsibility of the video player, and is not within scope of this API document.

HLS

Send (request playout URL):
`http://192.168.5.228/abr_vod/1359049483450/HLS`

Receive (session-based dynamic URL):
`http://192.168.5.228:5555/http_adaptive_streaming/`**`JHGAAAAAJCEJDPCF`**`.m3u8`

Send (propagate to client device for the manifest request) :
`http://192.168.5.228:5555/http_adaptive_streaming/JHGAAAAAJCEJDPCF.m3u8`

Receive (manifest with the list of available ABR layers):

```
#EXTM3U
#EXT-X-VERSION:4
#EXT-X-STREAM-INF:PROGRAM-
ID=1,BANDWIDTH=394382,RESOLUTION=240x180
JHGAAAAAJCEJDPCF.m3u8/Level(1)
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=96002
JHGAAAAAJCEJDPCF.m3u8/Level(2)
```

Send (request on the selected layer):
`http://192.168.5.228:5555/http_adaptive_streaming/MHGAAAAAJKEJDPCF.m3u8/Level(1)`

Receive (available fragments list):

```
#EXTM3U
#EXT-X-VERSION:4
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:8.008,
Level(1)/Segment(0).ts?ts=0
#EXTINF:8.008,
Level(1)/Segment(1).ts?ts=80080000
#EXTINF:8.008,
Level(1)/Segment(2).ts?ts=160160000
#EXTINF:8.008,
Level(1)/Segment(3).ts?ts=240240000
.....
#EXT-X-ENDLIST
```

Send (fragment request):
http://192.168.5.228:5555/http_adaptive_streaming/OHGAAAAACAFJDPCF.m3u8/Level(1)/**Segment(0).ts?ts=0**

Receive:

Video binary data

SMOOTH

Send (request playout URL for VOD asset):
http://192.168.5.228/abr_vod/1359049483450/SMOOTH

Receive (session-based dynamic URL):
http://192.168.5.228:5555/http_smooth_streaming/DIGAAAAAEPFJDPCF.ism/Manifest

Send (propagate to client device for the client manifest request) :
http://192.168.5.228:5555/http_smooth_streaming/DIGAAAAAEPFJDPCF.ism/Manifest

Receive (client manifest file):

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Created with Video and Storage Processing Platform Streamer
for version 3.4.0.0-->
<SmoothStreamingMedia
  MajorVersion="2"
  MinorVersion="1"
  Duration="56396340000">
    <StreamIndex
    Type="video"
    Name="video"
    Chunks="2817"
    QualityLevels="1"
    MaxWidth="240"
    MaxHeight="180"
    DisplayWidth="240"
    DisplayHeight="180"
    Url="QualityLevels({bitrate})/Fragments
    (video={start time})">
        <QualityLevel
      Index="0"
      Bitrate="394382"
      FourCC="H264"
      MaxWidth="240"
      MaxHeight="180"
```

```
CodecPrivateData="000000016742C00DF2078CFCF80A9060606F00000303E
90000EA60E040030D400124FE70180A000000000168CE3C8000" />
        <c
       t="0"
       d="20020000" />
        <c
       d="20020000" />
       ....
    </StreamIndex>
</SmoothStreamingMedia>
```

Send (fragment request):
http://192.168.5.228:5555/http_smooth_streaming/FJGAAAAAPNJJDPCF.ism/QualityLevels(394382)/Fragments(video=0)

Receive:

Video binary data

Appendix C - XSD Schemas

XSD Schemas

Rolling Buffer Allocation Reply

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="rolling_buffer_allocation_result">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:anyURI" name="url"/>
        <xs:element name="channel_status">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="inner_buffer" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:xs:dateTime" name="start_time" use="optional"/>
                      <xs:attribute type="xs:xs:dateTime" name="end_time" use="optional"/>
                      <xs:attribute type="xs:string" name="no_data" use="optional"/>
                      <xs:attribute type="xs:string" name="playable" use="optional"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute type="xs:string" name="name"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

RS-DVR Session Allocation Reply

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="allocation_result">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:anyURI" name="url"/>
        <xs:element type="xs:string" name="session_id"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

VOD Session Allocation Reply

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="allocation_result">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:anyURI" name="url"/>
        <xs:element type="xs:string" name="session_id"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

MP4 Rolling Buffer Download Reply

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="mp4_download_result">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="url"/>
        <xs:element name="channel_status">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="inner_buffer">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:xs:dateTime" name="start_time"/>
                      <xs:attribute type="xs:xs:dateTime" name="end_time"/>
                      <xs:attribute type="xs:string" name="playable"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute type="xs:string" name="name"/>
          </xs:complexType>
        </xs:element>
        <xs:element type="xs:byte" name="progress_percentage"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Tear Down Session Reply

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="X">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:boolean" name="torn_down"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Rolling Buffer Status

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="rolling_buffer_status">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="channel">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="inner_buffer" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:xs:dateTime" name="start_time"/>
                      <xs:attribute type="xs:xs:dateTime" name="end_time" />
                      <xs:attribute type="xs:string" name="erroneous" use="optional"/>
                      <xs:attribute type="xs:string" name="no_data" use="optional"/>
                      <xs:attribute type="xs:string" name="playable" use="optional"/>
                      <xs:attribute type="xs:string" name="out_of_sync" use="optional"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute type="xs:string" name="name"/>
            <xs:attribute type="xs:string" name="bitrate_type"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>


</xs:schema>
```

Mixed Rolling Buffer Status

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="rolling_buffer_status">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="channel" maxOccurs="2" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="inner_buffer" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:xs:dateTime" name="start_time" use="optional"/>
                      <xs:attribute type="xs:xs:dateTime" name="end_time" use="optional"/>
                      <xs:attribute type="xs:string" name="playable" use="optional"/>
                      <xs:attribute type="xs:string" name="no_data" use="optional"/>
                      <xs:attribute type="xs:string" name="out_of_sync" use="optional"/>
                      <xs:attribute type="xs:string" name="erroneous" use="optional"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute type="xs:string" name="name"/>
            <xs:attribute type="xs:string" name="bitrate_type"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```