ONVIF™
Streaming Specification

Version 25.12

December, 2025

CONTENTS

1 Scope

This document defines the ONVIF specific streaming extensions for live and replay streaming. The corresponding web service APIs to retrieve the streaming URIs are defined in separate documents and are not covered in this document.

2 Normative references

ISO/IEC 14496-2:2004, *Information technology -- Coding of audio-visual objects -- Part 2: Visual*

ISO/IEC 14496-3:2005, *Information technology -- Coding of audio-visual objects -- Part 3: Audio*

ISO/IEC 14496-10:2008, *Information technology -- Coding of audio-visual objects -- Part 10: Advanced Video Coding*

ISO/IEC 23008-2:2015, *Information technology -- High efficiency coding and media delivery in heterogeneous environments -- Part 2: High efficiency video coding*
ITU-T G.711, *Pulse code modulation (PCM) of voice frequencies* <http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.711-198811-I!!PDF-E&type=items>

ITU-T G.726, *40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)*
<http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.726-199012-I!!PDF-E&type=items>

RSA Laboratories, PKCS #10 v1.7: *Certification Request Syntax Standard, RSA Laboratories*
<ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf>

IETF RFC 2246, The TLS Protocol Version 1.0
<http://www.ietf.org/rfc/rfc2246.txt>

IETF RFC 2326, Real Time Streaming Protocol (RTSP)
<http://www.ietf.org/rfc/rfc2326.txt>

IETF RFC 2396, Uniform Resource Identifiers (URI): General Syntax
<http://www.ietf.org/rfc/rfc2396.txt>

IETF RFC 2435, RFC2435 - RTP Payload Format for JPEG-compressed Video
<http://www.ietf.org/rfc/rfc2435.txt>

IETF RFC 3016, RTP Payload Format for MPEG-4 Audio/Visual Streams

`http://www.ietf.org/rfc/rfc3016.txt`

IETF RFC 3550, RTP: A Transport Protocol for Real-Time Applications
<http://www.ietf.org/rfc/rfc3550.txt>

IETF RFC 3551, RTP Profile for Audio and Video Conferences with Minimal Control
<http://www.ietf.org/rfc/rfc3551.txt>

IETF RFC 3640, RTP Payload Format for Transport of MPEG-4 Elementary Streams
<http://www.ietf.org/rfc/rfc3640.txt>

IETF RFC 3711, The Secure Real-time Transport Protocol (SRTP)
<https://www.ietf.org/rfc/rfc3711.txt>

IETF RFC 3984, RTP Payload Format for H.264 Video
<http://www.ietf.org/rfc/rfc3984>

IETF RFC 3016, RTP Payload Format for MPEG-4 Audio/Visual Streams
<http://www.ietf.org/rfc/rfc3016>

IETF RFC 4566, SDP: Session Description Protocol

<http://www.ietf.org/rfc/rfc4566.txt>

IETF RFC 4567, Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)
<http://www.ietf.org/rfc/rfc4567.txt>

IETF RFC 4571, Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport
<http://www.ietf.org/rfc/rfc4571.txt>

IETF RFC 4585, Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)
<http://www.ietf.org/rfc/rfc4585.txt>

IETF 5104, Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)
<http://www.ietf.org/rfc/rfc5104.txt>

IETF RFC 5888 The Session Description Protocol (SDP) Grouping Framework
<https:/tools.ietf.org/html/rfc5888>

IETF RFC 6455, The WebSocket Protocol
<http://www.ietf.org/rfc/rfc6455.txt>

IETF RFC 7714, AES-GCM Authenticated Encryption in the Secure Real-time Transport Protocol (SRTP)
<http://www.ietf.org/rfc/rfc7714.txt>

IETF RFC 7798, RTP Payload Format for High Efficiency Video Coding (HEVC)
<http://www.ietf.org/rfc/rfc7798.txt>

IETF RFC 7826, Real-Time Streaming Protocol Version 2.0
<http://www.ietf.org/rfc/rfc7826.txt>

IETF RFC 8285, A General Mechanism for RTP Header Extensions
<https://www.ietf.org/rfc/rfc8285.txt>

GZIP file format specification version 4.3
<http://tools.ietf.org/html/rfc1952>

Apple Computer Inc. RTSP over HTTP, Tunneling QuickTime RTSP and RTP over HTTP
<https://opensource.apple.com/source/QuickTimeStreamingServer/QuickTimeStreamingServer-412.42/Documentation/RTSP_Over_HTTP.pdf>

ONVIF Core Specification
<http://www.onvif.org/specs/core/ONVIF-Core-Specification.pdf>

ONVIF Media Service Specification
<http://www.onvif.org/specs/srv/media/ONVIF-Media-Service-Spec.pdf>

ONVIF Media2 Service Specification
<http://www.onvif.org/specs/srv/media/ONVIF-Media2-Service-Spec.pdf>

ONVIF Replay Control Service Specification
<https://www.onvif.org/specs/srv/replay/ONVIF-ReplayControl-Service-Spec.pdf>

3 Terms and Definitions

3.1 Definitions

Access Unit	One or more frames or samples of audio, video, or metadata which are contained in a group of RTP packets having the same presentation time.

Metadata	All streaming data except video and audio, including video analytics results, PTZ position data and other metadata (such as textual data from POS applications).
Recording	Represents the currently stored media (if any) and metadata on the NVS from a single data source. A recording comprises one or more tracks. A recording can have more than one track of the same type e.g. two different video tracks recorded in parallel with different settings
Track	An individual data channel consisting of video, audio, or metadata. This definition is consistent with the definition of track in [RFC 2326]

3.2 Abbreviations

AAC	Advanced Audio Coding
EOI	End Of Image
HEVC	High Efficiency Video Coding also coined H.265
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Expert Group
MPEG-4	Moving Picture Experts Group - 4
PTZ	Pan/Tilt/Zoom
RTCP	RTP Control Protocol
RTP	Realtime Transport Protocol
RTSP	Real Time Streaming Protocol
RTSPS	Secure Real Time Streaming Protocol
SDP	Session Description Protocol
SOI	Start Of Image
SOF	Start Of Frame
SOS	Start Of Scan
SRTP	Secure Realtime Transport Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
UTF	Unicode Transformation Format
EXI	Efficient XML Interchange Format

4 Overview

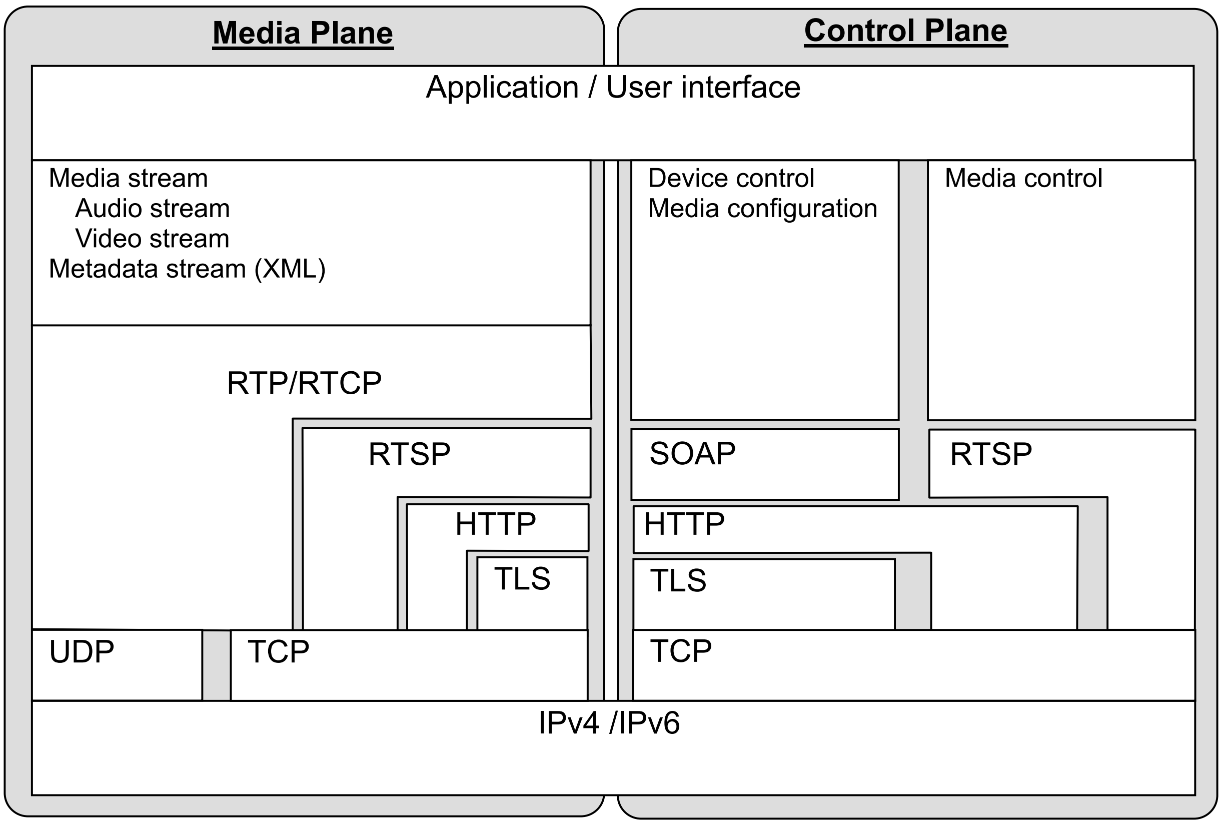

Figure 1: Layer structure

This standard defines media streaming options and formats. A distinction is made between *media plane* and *control plane,* as illustrated in Figure 1. A set of media streaming (audio, video and meta data) options, all based on RTP [RFC 3550], are described in order to provide interoperable media streaming services.

The metadata streaming container format allows well-defined, real-time streaming of analytics, PTZ status and notification data.

Media configuration is done over SOAP/HTTP and is covered by the media configuration service as discussed in Section 4.6.

Media control is accomplished over RTSP as defined in RFC 2326. This standard utilizes RTP, RTCP and RTSP profiling, as well as JPEG over RTP extensions and multicast control mechanisms.

The standard introduces extensions to the RTSP standard to allow bi-directional streaming connections.

Streaming configurations for the following video codecs are provided:

- JPEG (over RTP), see 5.1.4.
- MPEG-4, Simple Profile (SP) [ISO 14496-2]
- MPEG-4, Advanced Simple Profile (ASP) [ISO 14496-2]
- H.264, baseline [ISO 14496-10]
- H.264, main [ISO 14496-10]
- H.264, extended [ISO 14496-10]

- H.264, high [ISO 14496-10]
- HEVC [ISO23008-2]

and for the following audio codecs:

- G.711 [ITU-T G.711 uLaw]
- G.726 [ITU-T G.726]
- AAC [ISO 14496-3]

5 Live Streaming

This section describes real-time streaming of video, audio and metadata. There is *no specific* service associated with the real-time streaming. The real-time configurations via Web Service commands are defined in the Media Service and the ReceiverService.

5.1 Media stream protocol

5.1.1 Transport format

Real-time Transport Protocol (RTP) is a media transfer protocol (see Section 5.1.2). The following four sections describe RTP data transfer.

5.1.1.1 RTP data transfer via UDP

UDP has the smallest overhead and is able to transfer real-time data in an efficient manner. A device shall support the RTP/UDP protocol and the device should support RTP/UDP multicasting.

5.1.1.2 RTP/TCP

This optional mode has been deprecated due to ambiguities in the interpretation of the respective RFCs. RTP/TCP protocol is defined in [RFC 4571] and [RFC 4572].

5.1.1.3 RTP/RTSP/TCP

The device should support media transfer using RTP/RTSP to traverse a firewall using an RTSP tunnel. This protocol shall conform to [RFC 2326] Section 10.12 (Embedded [Interleaved] Binary Data).

5.1.1.4 SRTP data transfer via UDP

This mode allows secure transmission of RTP packets via UDP unicast and multicast. See RFC 3711 for transmission and RFC 4567 for key exchange.

5.1.1.5 RTP/RTSP/HTTP/TCP

The data stream shall be sent via HTTP to traverse a firewall. A device shall support media transfer using RTP/RTSP/HTTP/TCP. If a device supports TLS, the data stream shall support transmission via HTTPS to traverse a firewall, and a device shall support media transfer using RTP/RTSP/HTTPS/TCP.

This protocol shall conform to [RFC 2326] Section 10.12.

This traversal method shall conform to [RTSP over HTTP]. Note that according to this specification all data sent from the client via POST must be base64 encoded.

5.1.1.6 RTP/RTSP/TCP/WebSocket

The device indicating support for RTSP over WebSocket, as explained in section 5.11 of the ONVIF Media2 Service Specification and section 5.5 of the ONVIF Replay Control Service Specification, shall support streaming media using WebSocket protocol according to this specification. The provided URI shall set the hierarchical part (hier_part) to absolute path (abs_path) [RFC 2396]. For example, if the WebSocket URI with network path is “ws://1.2.3.4/my-websocket-uri”, the provided URI shall be “ws:/my-websocket-uri”.

For RTSP tunneling over WebSocket a device shall support RTP/RTSP/TCP interleaved binary data as defined in [RFC 2326] Section 10.12.

WebSocket protocol implementation shall conform to [RFC 6455] - The WebSocket Protocol.

The mechanism to be used for establishing a WebSocket connection between a client and server is explained in detail in Section 7.

5.1.2 Media Transport

5.1.2.1 RTP

The Real-time Transport Protocol provides real-time transfer for media streams between two end points. The RTP protocol provides support for re-ordering, de-jittering and media synchronization.

All media streams transferred by the RTP protocol shall conform to [RFC 3550], [RFC 3551], [RFC 3984], [RFC 7798], [RFC 3016] or [RFC 3640], and JPEG over RTP (see Section 5.1.3).

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
V		P	X	CC				M	PT							sequence number															
time stamp																															
synchronization source (SSRC) identifier																															

Figure 2: RTP header

An RTP header shall be filled up with following values.

Table 1: RTP header value

Header field	Value	Description
Version (V): 2 bits	2	
Padding (P): 1 bit	0/1	If the payload includes padding octet, this should be set to “1”
Extension (X): 1 bit	0/1	Depends on the use of extension of RTP header. The specification defines two scenarios where a RTP header extension could be used to transmit additional information: 1) “JPEG over RTP” (see Section 5.1.3). 2) Replay (see Section 6) If the header extension is used the Extension bit shall be set.
CSRC count (CC):	0	

Header field	Value	Description
4 bits		
Marker (M): 1 bit	0/1	The usage shall be conform to related RFCs (e.g. [RFC 3984] for H.264 Video) or to this standard e.g "JPEG over RTP" (see Section 5.1.3) or RTP streaming of metadata (see Section 5.1.2.1.1).
Payload type (PT): 7 bits	See [RFC 3551] Section 6.	
Sequence Number: 16 bits		The initial value of the "sequence number" should be random (unpredictable) to make known-plaintext attacks on encryption more difficult. This number increments by one for each RTP data packet sent
timestamp: 32 bits		The initial value of the "timestamp" should be random (unpredictable) to make known-plaintext attacks on encryption more difficult. See Section 5.1.2.2.1 for further details of Media Synchronization. The usage of the timestamp is dependent on the codec.
SSRC 32 bits		The synchronization source for the data stream. This specification makes no restrictions on the use of this field.

5.1.2.1.1 RTP for Metadata stream

Metadata streams are also transported by RTP. The usage of payload type, marker and timestamp for RTP header for the metadata stream is defined in the following way:

- A dynamic payload type (96-127) shall be used for payload type which is assigned in the process of a RTSP session setup.
- The RTP marker bit shall be set to "1" when the XML document is closed.
- It is RECOMMENDED to use an RTP timestamp representing the creation time of the RTP packet with a RTP clock rate of 90000 Hz. Only UTC timestamps shall be used within the metadata stream. The synchronization of video and audio data streams is done using RTCP.

The Metadata payload is an XML document with root node `tt:MetaDataStream`. There is no limitation on the size of the XML document. If GZIP compression is used, the payload starts with a GZIP header according to RFC 1952 followed by the compressed data. A marker bit signals the end of the compressed data. When a synchronization point (see section "Synchronization Points" of the ONVIF Media Service Specification) is requested for the stream, the previous XML document shall be closed and a new one started. It is RECOMMENDED to start new XML documents after 1 second, at the longest. The RTP timestamp of the Metadata stream has no specific meaning. The Metadata stream multiplexes Metadata from different sources. This specification defines placeholders for the Scene Description of the Video Analytics, the PTZ Status of the PTZ controller and the Notifications of the Event Configuration. A device can select which of these parts should be multiplexed into the Metadata during the Media Configuration (see seciont “Metadata Configuration” of the ONVIF Media Service Specification). Each part can appear multiple times in arbitrary order within the document. A Metadata connection can be bi-directional using the backchannel mechanism (see Section 5.3).

Metadata stream contains the following elements:

- VideoAnalyticsStream
- PTZStream
- EventStream

The place-holders for the different metadata sources have the following XMLstructure:

```
<xs:complexType name="VideoAnalyticsStream">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Frame" type="tt:Frame"/>
    ...
  </xs:choice>
</xs:complexType>
<xs:complexType name="PTZStream">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="PTZStatus" type="tt:PTZStatus"/>
    ...
  </xs:choice>
</xs:complexType>
<xs:complexType name="EventStream">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="wsnt:NotificationMessage"/>
    ...
  </xs:choice>
</xs:complexType>
```

Note: For a PTZ supported device, the PTZStream in metadata provides the PTZ position information, whenever there is a change in the PTZ position, whereas PTZStatus defined inside VideoAnalyticsStream, provides PTZ position information at the time of generating scene description.

The following is an example of a metadata XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
  <tt:MetadataStream xmlns:tt="http://www.onvif.org/ver10/schema">
    <tt:VideoAnalytics>
      <tt:Frame UtcTime="2008-10-10T12:24:57.321">
        ...
      </tt:Frame>
      <tt:Frame UtcTime="2008-10-10T12:24:57.621">
        ...
      </tt:Frame>
    </tt:VideoAnalytics>
  </tt:MetadataStream>
  <?xml version="1.0" encoding="UTF-8"?>
    <tt:MetadataStream xmlns:tt="http://www.onvif.org/ver10/schema">
      <tt:Event>
        <wsnt:NotificationMessage>
          <wsnt:Message>
            <tt:Message UtcTime= "2008-10-10T12:24:57.628">
              ...
            </tt:Message>
          </wsnt:Message>
        </wsnt:NotificationMessage>
      </tt:Event>
    </tt:MetadataStream>
```

5.1.2.2 RTCP

The RTP Control Protocol provides feedback on quality of service being provided by RTP and synchronization of different media streams. The RTCP protocol shall conform to [RFC 3550].

For a feedback request, [RFC 4585] and [RFC 5104] should be supported.

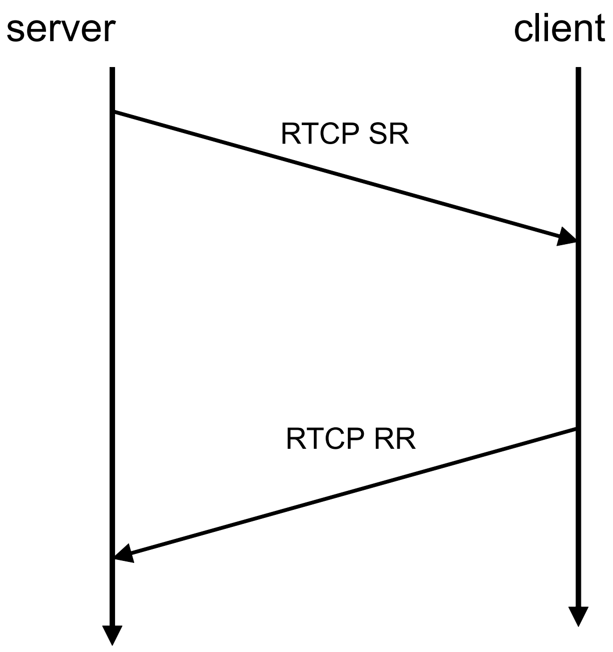

Figure 3: RTCP sequence

5.1.2.2.1 Media synchronization

A client MAY receive audio and video streams simultaneously from more than one device. In this case, each stream uses a different clock (from data acquisition to packet receiving). RTCP Sender Reports (SR) are used to synchronize different media streams. RTCP SRs shall conform to [RFC 3550].

The RTCP Sender Report (SR) packet has fields for the RTP timestamp and for a wall clock timestamp (absolute date and time, 64bit NTP [Network Time Protocol]). See Figure 4.

A device shall support RTCP Sender Report for media synchronization. The client should use RTCP for the media synchronization.

0			1	2 3
0 1	2	3 4 5 6 7	8 9 0 1 2 3 4 5	6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
V	**P**	**RC**	**PT=SR=200**	**length**
SSRC of sender				
NTP timestamp, most significant word				
NTP timestamp, least significant word				
RTP timestamp				
sender's packet count				
:				
:				

Figure 4: RTCP Sender Report

The wall clock should be common in the device and each timestamp value should be determined properly. The client can synchronize different media streams at the appropriate timing based on the RTP clock and wall clock timestamps (see Figure 5).

In case of multiple devices, the NTP timestamp should be common to all devices, and the NTP server should be required in the system [1].

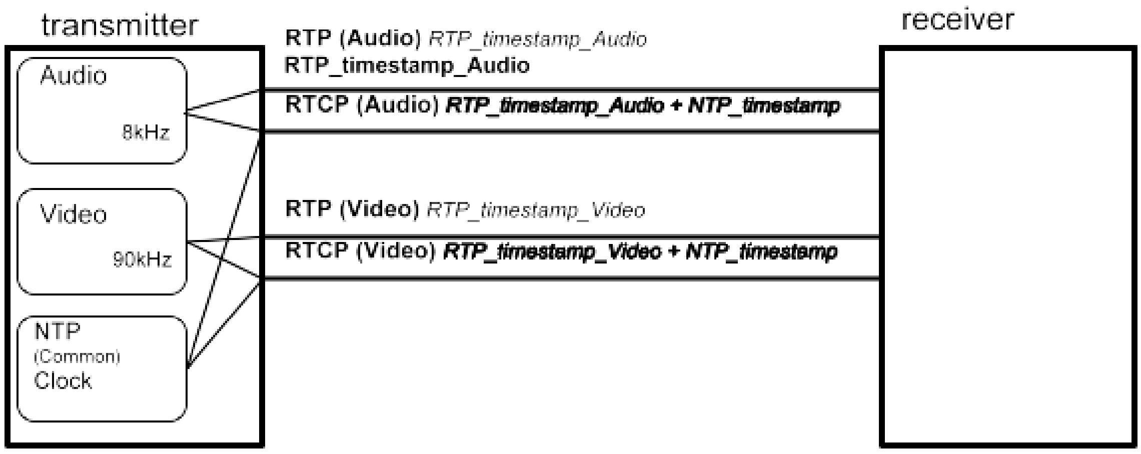
transmitter
receiver
Audio
8kHz
Video
90kHz
NTP
(Common)
Clock
RTP (Audio) RTP_timestamp_Audio
RTP_timestamp_Audio
RTCP (Audio) RTP_timestamp_Audio + NTP_timestamp
RTP (Video) RTP_timestamp_Video
RTCP (Video) RTP_timestamp_Video + NTP_timestamp

Figure 5: Media Synchronization

5.1.3 Synchronization Point

Synchronization points allow clients to decode and correctly use data after the synchronization point. A synchronization point MAY be requested by a client in case of decoder error (e.g. in consequence of packet loss) to enforce the device to add an I-Frame as soon as possible or to request the current ptz or event status.

The WebService based methods require to support the Synchronization Point request as defined in the section “Synchronization Point” of the ONVIF Media Service Specification.

In addition it is recommended to support the PLI messages as described in [RFC 4585] in order to allow receivers as defined in the ONVIF Receiver Service Specification to request a Synchronization Point.

For H.264 and H.265 Video the SPS/PPS header shall be sent in-band if these have been changed during the transmission.

5.1.4 JPEG over RTP

5.1.4.1 Overall packet structure

The syntax for transmitting JPEG streams follows [RFC 2435]. The syntax does allow embedding additional data, beyond the limits of [RFC 2435], by using an optional RTP header extension, as specified below, with some of the RTP packets. This option, however, changes the exact semantics for frames which include such packets.

The overall format of the JPEG RTP packet is shown in Figure 6.

[1]The client can get information about “NTP server availability” from the devices by using the GetNTP command. Refer to Section 8.2.5

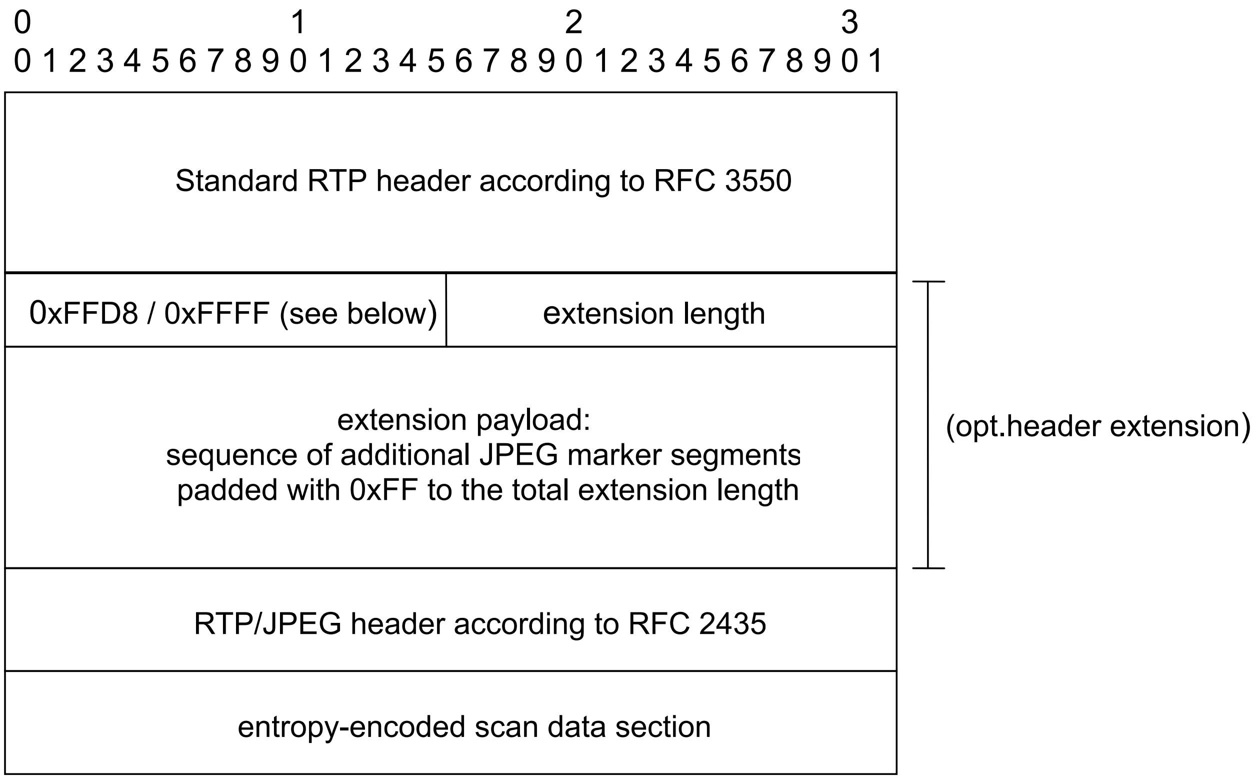
0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Standard RTP header according to RFC 3550
0xFFD8 / 0xFFFF (see below)
extension length
extension payload: sequence of additional JPEG marker segments padded with 0xFF to the total extension length
(opt.header extension)
RTP/JPEG header according to RFC 2435
entropy-encoded scan data section

Figure 6: RTP/JPEG packet structure (only the typical content is listed for the extension payload)

In order to distinguish an optional RTP header extension from possible other header extensions, the first 16 bits (the first two octets of the four-octet extension header) of an RTP shall have the value `0xFFD8` (JPEG SOI marker) for the initial packet and `0xFFFF` for other RTP packets within a frame.

As required by [RFC 3550], the presence of the optional header extension shall be signalled via the X-bit of the RTP header. The extension length field within the header extension counts the number of 32-bit items following as extension payloads. For example, a zero-length field following the 32-bit extension header represents an empty header extension).

The entropy-encoded scan data section MAY not be present in all RTP packets. A complete RTP/JPEG header however shall be present in the initial packet of every frame and all packets containing an entropy-encoded scan data section, otherwise it MAY be missing.

The fragment offset field within the RTP/JPEG header, according to [RFC 2435], should be used as if no header extension would be present. Additionally, if a packet does not contain an entropy-encoded scan data segment, but contains a header extension the fragment offset field shall not be zero if any packets containing an entropy-encoded scan data section for the same frame have been transmitted. If the initial packet of a frame contains no header extension, according to this standard, its fragment offset field shall be zero, otherwise it should be zero. All packets including an RTP/JPEG header with a fragment offset of zero and a Q value between 128-255 shall include a quantization table header according to Section 3.1.8 of [RFC 2435], other packets shall NOT include this header.

5.1.4.2 Logical decoding specification

For the decoding specification, it is assumed that the original packet order within the RTP stream has been restored according to the RTP sequence numbering.

If the initial packet of a frame contains no RTP header extension as specified above, decoders shall generate the complete scan header and perform the decoding as specified by [RFC 2435]. The scan data sections and payloads of any header extension conforming to this specification, up to and including the next RTP packet with its marker bit set, shall be concatenated as they occur within the stream ignoring their fragment offset values.

Otherwise (at least an empty header extension as specified above is present in the initial packet of a frame), the following rules apply for each such frame:

- If the initial packet of a frame does not contain an entropy-encoded scan data segment, but contains a header extension as specified above, then decoders shall concatenate its header extension payload with (possibly empty or not existing) header extension payload(s) conforming to this specification of the subsequent packets up to and including the first packet with the RTP marker bit set or containing an entropy-encoded scan data segment.
- The concatenated initial RTP header extension payload (sequence) shall be logically prepended with a JPEG SOI marker (0xFFD8).
- If the Q-value of the RTP/JPEG scan header within the initial packet of a frame is not zero, the quantization tables shall be pre-initialized according to the rules of [RFC 2435]. If Q is equal to zero the quantization tables shall be copied from the previous frame, allowing for DQT markers within this initial header extension payload (sequence) to override them.
- If this frame is the initial frame of a sequence, the Huffman tables shall be pre-initialized according to [RFC 2435]. The Huffman tables for all subsequent frames shall be copied from the previous frame, allowing the frames to be overridden by DHT markers within the initial header extension payload (sequence).
- If the initial RTP header extension payload (sequence) supplies no DRI marker, but the RTP/JPEG header of the initial packet of a frame contains an RTP/JPEG restart marker, a DRI marker corresponding to the rules of [RFC 2435] shall be appended to the initial header extension payload (sequence). Otherwise, if the initial RTP header extension (sequence) supplies a DRI marker, the marker shall take precedence over any other RTP/JPEG restart marker according to [RFC 2435] for the same frame. However, for compatibility with decoders conforming to [RFC 2435] only, encoders normally should use an RTP/JPEG restart marker with consistent values, if restart intervals are to be used.
- DRI markers shall NOT be derived from previous frames.
- If the initial RTP header extension payload (sequence) supplies no SOF marker, which otherwise takes precedence, a SOF marker shall be appended to it with the following values:
  - If both the width and height field of the RTP/JPEG header are zero, the SOF marker of the previous frame shall be used.
  - Otherwise it shall be derived according to the rules of [RFC 2435].

  However, as long as the (rounded up) image size fits within the range as specified in [RFC 2435], encoders should specify the image size within the RTP/JPEG header consistent with the values of an additional SOF header.
- If the initial header extension payload (sequence) supplies no SOS marker, a corresponding marker shall be derived according to [RFC 2435] and appended to it, otherwise the SOS marker in the extension takes precedence.

  An SOS marker shall NOT be derived from previous frames.

  If the SOS marker is present and not followed by entropy-encoded scan data within the extension, the marker shall be the final marker within the initial extension payload (sequence) of a frame. Necessary padding with 0xFF-octets shall NOT follow this marker but MAY precede it.
- The remaining entropy-encoded scan data and header extensions payloads shall be logically appended in the same order as they occur within the RTP stream up to the end of the frame as indicated by the RTP marker bit. A final EOI marker shall also be added if it is not yet present within the logical sequence for this frame,.

For each frame, the resulting sequence up to and including the first (possibly added) EOI marker shall be a valid (possibly abbreviated) JPEG stream, resulting in one complete image from the decoding process for this frame. The meaning of any data after this first EOI marker for each frame is outside the scope of this specification.

Implementations should provide for each frame the complete JPEG headers. This holds especially for the width and height information as well as the quantization and Huffman tables. If such important information is not provided for each frame, both playback and multicast streaming may suffer from incomplete JPEG header information.

5.1.4.3 Supported colour spaces and sampling factors

A Transmitter should use only greyscale and YCbCr colour space. A Client shall support both greyscale and YcbCr.

The sampling factors for YcbCr shall correspond to the values supported by [RFC 2435]. For example, a sampling factor of 4:2:0 (preferred) or 4:2:2.

5.1.4.4 Pixel aspect ratio handling

The pixel aspect ratio of JPEG files can be specified within the JFIF marker. If the pixel aspect ratio is different from the standard 1:1 and 1:2 ratio according to [RFC 2435], this marker should be transmitted in the initial header extension payload (sequence) of every frame to specify the (for interlaced material field-based) pixel aspect ratio.

5.1.4.5 Interlaced handling

Interlaced video is encoded as two independent fields and signalled as specified by [RFC 2435] within the RTP/JPEG header.

Both fields shall use the same colour space, sampling factors and pixel aspect ratio.

Interlaced encoding should NOT be used if the frame was originally scanned progressively.

5.2 Media control protocol

5.2.1 Stream control

The media stream is controlled using the protocol defined in the URI. The URI is returned in response to the GetStreamUri command defined in the ONVIF Media Service Specification.

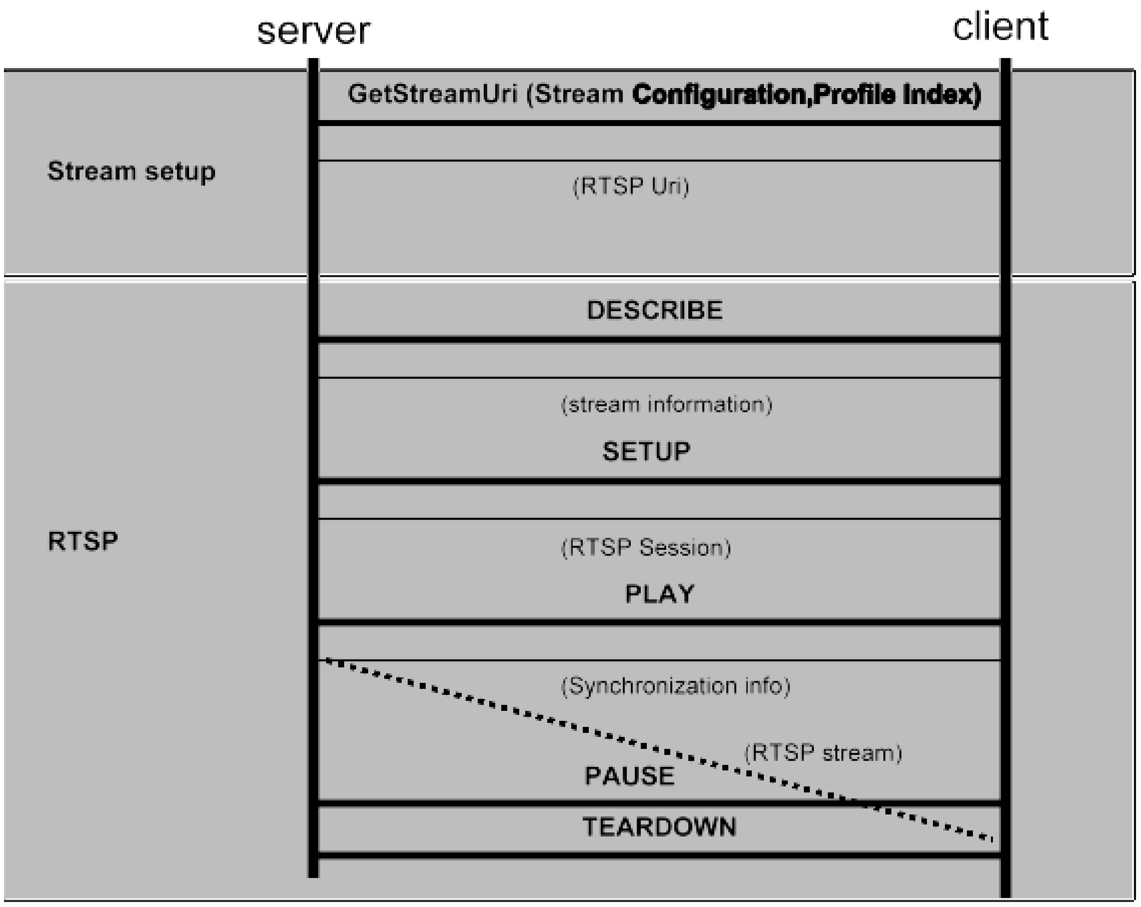

Figure 7: Stream Control

5.2.2 RTSP

5.2.2.1 General

All devices and clients shall support RTSP ([RFC 2326]) for session initiation and playback control. RTSP shall use TCP as its transport protocol, the default TCP port for RTSP traffic is 554. The Session Description Protocol (SDP) shall be used to provide media stream information and SDP shall conform to [RFC 4566].

Table 2: RTSP methods.

Method	Direction	SPEC[a]	Description
OPTIONS	R->T T->R	M X	Required to get optional method capability and to allow different versions in the future.
DESCRIBE	R->T	M	Required to retrieve media parameters within the designated profile.
ANNOUNCE	R->T T->R	X	
SETUP	R->T	M	Required to set media session parameters.
PLAY	R->T	M	Required to start media stream.
PAUSE	R->T	Live: O Replay: M	Required to temporarily stop media playback. Handling multiple streams in a narrow bandwidth network, by suspending RTP stream, the traffic can be well controlled by

Method	Direction	SPEC[a]	Description
			reducing redundant data and congested network traffic can be avoided.
TEARDOWN	R->T	M	Required to release a media session.
GET_PARAMETER	R->T T->R	O	
SET_PARAMETER	R->T T->R	O O	An optional method to keep an RTSP session alive (R->T direction only).
REDIRECT	T->R	X	
RECORD	R->T	X	

aX: Not supported, M: Mandatory, O: Optional

Devices shall support aggregate stream control, in which PLAY commands are sent to the control URI for the whole session. Devices may support non aggregate stream control, in which PLAY commands are sent separately to the stream control URIs in the media sections of the SDP file. Support for non aggregate stream control is signalled via the Media Streaming Capabilities.

5.2.2.2 Timeout and keep-alive handling

An RTSP client is required to keep the RTSP session alive and prevent it from session timeout (see [RFC 2326] Section 12.37).

This specification recommends the following behavior of keeping RTSP sessions alive for both Unicast and Multicast streaming:

- In all RTSP SETUP responses, a server should include the Timeout value according to [RFC 2326] Section 12.37.
- A server shall interpret any RTSP request other than TEARDOWN as a keep-alive.
- A server shall interpret any RTCP receiver report for a unicast stream as keep-alive for the RTSP session.
- If an RTSP session times out, a server should close related RTP streams as long as they are not shared with other active RTSP sessions.

Note: for timeout handling see also the following sections of RFC 7826:

- Section 10.5. Showing Liveness
- Annex C.1.6.2. RTSP Session Keep-Alive

Figure 8 depicts the timing requirement for a client to signal being alive. For improved server performance a client should send RTCP receiver reports which in turn do not require to send additional RTSP commands for keeping the session alive.

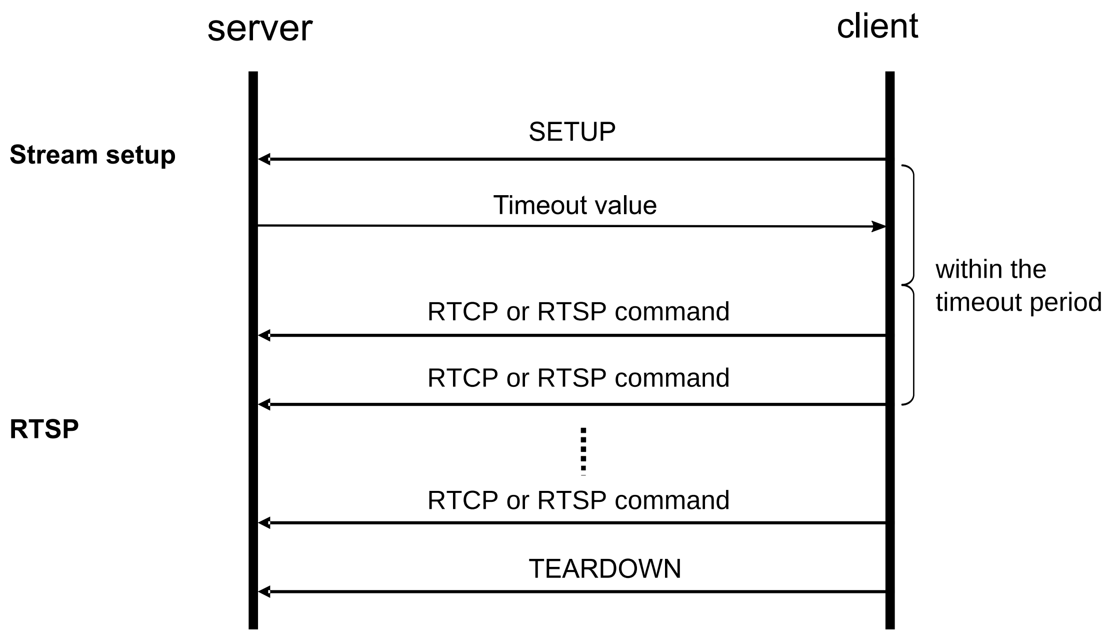

Figure 8: Keep Alive

5.2.2.3 RTSP audio and video synchronization

In order that clients may immediately begin synchronizing video and audio streams, and computing absolute UTC timestamps for incoming packets for recording purposes, a transmitter should include the following header fields in the RTSP PLAY response:

- Range ([RFC 2326] section 12.29). This SHALL include a start time in clock units ([RFC 2326] section 3.7), *not* SMPTE or NPT units.
- RTP-Info ([RFC 2326] section 12.33). This SHALL include an rtptime value which corresponds to the start time specified in the Range header.

Example:

```
client->server:   PLAY rtsp://example.com/onvif_camera/video RTSP/1.0
                  Cseq: 4
                  Range: npt=now-
                  Session: 12345678

server->client:   RTSP/1.0 200 OK
                  Cseq: 4
                  Session: 12345678
                  Range: clock=20100217T143720.257Z-
                  RTP-Info: url=rtsp://example.com/onvif_camera/video; seq=1234;rtptime=3450012
```

5.2.2.4 RTSP session for a metadata stream

In the case of a metadata stream, the SDP description “application” shall be used in the DESCRIBE response for media type and one of these encoding a names shall be used

- “vnd.onvif.metadata” for uncompressed
- “vnd.onvif.metadata+gzip” for GZIP compressed
- "vnd.onvif.metadata.exi.ext" for EXI using compression parameters that are sent in-band

Example RTSP DESCRIBE message exchange between an RTSP Server (server) and an RTSP client (client):

```
client->server:   DESCRIBE rtsp://example.com/onvif_camera RTSP/1.0
                  Cseq: 1

server->client:   RTSP/1.0 200 OK
                  Cseq: 1
                  Content-Type: application/sdp
                  Content-Length: XXX
                  v=0
                  o=- 2890844256 2890842807 IN IP4 172.16.2.93
                  s=RTSP Session
                  m=audio 0 RTP/AVP 0
                  a=control:rtsp://example.com/onvif_camera /audio
                  m=video 0 RTP/AVP 26
                  a=control:rtsp://example.com/onvif_camera /video
                  m=application 0 RTP/AVP 107
                  a=control:rtsp://example.com/onvif_camera/metadata
                  a=recvonly
                  a=rtpmap:107 vnd.onvif.metadata/90000
```

5.2.2.5 Multicast streaming

A device shall include a valid multicast address in the "c=" field of a DESCRIBE response according to RFC 4566.

Remark: the optional dynamic multicast address assignment exception described in appendix C.1.7 of RFC 2326 allowing 0.0.0.0 addresses does not apply.

Chapter 10.7 TEARDOWN of [RFC 2326] states that a device shall stop the stream delivery for the given URI on tear down. This needs to be clarified in case of multicast: for a multicast stream the device shall stop sending packets for a multicast configuration when no more RTSP sessions are using the same multicast configuration nor its AutoStart flag has been set.

5.2.2.6 RTSP message example

This example shows the message transfer between an RTSP client (client) and an RTSP server (server). The client requests one audio and one video stream from the device. The Stream Uri “rtsp://example.com/onvif_camera” can be retrieved using the GetStreamUri command. Refer to Section „Stream URI“ of the ONVIF Media Service Specification.

```
Client->server:     DESCRIBE rtsp://example.com/onvif_camera RTSP/1.0
Cseq: 1
server->client:     RTSP/1.0 200 OK
                    Cseq: 1
                    Content-Type: application/sdp
                    Content-Length: XXX
                    v=0
                    o=- 2890844256 2890842807 IN IP4 172.16.2.93
                    s=RTSP Session
                    m=audio 0 RTP/AVP 0
                    a=control:rtsp://example.com/onvif_camera/audio
                    m=video 0 RTP/AVP 26
                    a=control:rtsp://example.com/onvif_camera/video

client->server:     SETUP rtsp://example.com/onvif_camera/audio RTSP/1.0
                    Cseq: 2
                    Transport: RTP/AVP;unicast;client_port=8002-8003

server->client:     RTSP/1.0 200 OK
                    Cseq: 2
                    Transport: RTP/AVP;unicast;client_port=8002-8003;
                    server_port=9004-9005
                    Session: 12345678; timeout=60
```

```
client->server:     SETUP rtsp://example.com/onvif_camera/video RTSP/1.0
                    Cseq: 3
                    Transport: RTP/AVP;unicast;client_port=8004-8005
                    Session: 12345678

server->client:     RTSP/1.0 200 OK
                    Cseq: 3
                    Transport: RTP/AVP;unicast;client_port=8004-8005;
                    server_port=9006-9007
                    Session: 12345678; timeout=60

client->server:     PLAY rtsp://example.com/onvif_camera RTSP/1.0
                    Cseq: 4
                    Range: npt=now-
                    Session: 12345678

server->client:     RTSP/1.0 200 OK
                    Cseq: 4
                    Session: 12345678
                    RTP-Info: url=rtsp://example.com/onvif_camera/video; seq=1234;rtptime=3450012, u
                    seq=22434;rtptime=1234566

client->server:     TEARDOWN rtsp://example.com/onvif_camera RTSP/1.0
                    Cseq: 5
                    Session: 12345678

server->client:     RTSP/1.0 200 OK
                    Cseq: 5
                    Session: 12345678
```

5.2.3 RTSP over HTTP

The RTSP over HTTP/HTTPS shall be supported in order to traverse a firewall. See Section 5.1.1.5 RTP/RTSP/HTTP/TCP.

5.2.4 Secure RTSPS

Secure RTP streaming requires a secure channel for key exchange. For this reason the RTSPS communication shall use TLS encryption.

5.3 Back channel connection

This section describes how a bidirectional connection can be established between a client and a server. The backchannel connection handling is done using RTSP [RFC 2326]. Therefore a mechanism is introduced which indicates that a client wants to establish a backchannel connection. RTSP provides feature-tags to deal with such functionality additions.

A device that supports bi-directional connections (e.g audio or metadata connections) shall support the introduced RTSP extensions.

5.3.1 RTSP Require tag

The RTSP standard [RFC 2326] can be extended by using additional headers objects. For that purpose a Require tag is introduced to handle special functionality additions (see [RFC 2326], 1.5 Extending Rtsp and 12.32 Require).

The Require-tag is used to determine the support of this feature. This header shall be included in any request where the server is required to understand that feature to correctly perform the request.

A device that supports backchannel and signals Audio output support via the AudioOutputs capability shall understand the backchannel tag:

- `www.onvif.org/ver20/backchannel`

An RTSP client that wants to built up an RTSP connection with a data backchannel shall include the Require header in its requests.

5.3.2 Connection setup for a bi- directional connection

A client shall include the feature tag in it's DESCRIBE request to indicate that a bidirectional data connection shall be established.

A server that understands this Require tag shall include an additional media stream in its SDP file as configured in its Media Profile.

An RTSP server that does not understand the backchannel feature tag or does not support bidirectional data connections shall respond with an error code *551 Option not supported* according to the RTSP standard. The client can then try to establish an RTSP connection without backchannel.

A SDP file is used to describe the session. To indicated the direction of the media data the server shall include the a=sendonly in each media section representing media being sent from the client to the server and a=recvonly attributes in each media section representing media being sent from the server to the client.

The server shall list all supported decoding codecs as own media section and the client chooses which one is used. The payload type and the encoded bitstream shall be matched with one of the a=rtpmap fields provided by the server so that the server can properly determine the audio decoder.

5.3.2.1 Describe example for a server without backchannel support:

```
Client – Server:  DESCRIBE rtsp://192.168.0.1 RTSP/1.0
Cseq: 1
User-Agent: ONVIF Rtsp client
Accept: application/sdp
Require:www.onvif.org/ver20/backchannel
Server – Client:    RTSP/1.0 551 Option not supported
                    Cseq: 1
                    Unsupported:www.onvif.org/ver20/backchannel
```

5.3.2.2 Describe example for a server with Onvif backchannel support:

```
Client – Server:  DESCRIBE rtsp://192.168.0.1 RTSP/1.0
Cseq: 1
User-Agent: ONVIF Rtsp client
Accept: application/sdp
Require:www.onvif.org/ver20/backchannel
Server – Client:    RTSP/1.0 200 OK
                    Cseq: 1
                    Content-Type: application/sdp
                    Content-Length: xxx
                    v=0
                    o= 2890842807 IN IP4 192.168.0.1
                    s=RTSP Session with audiobackchannel
                    m=video 0 RTP/AVP 26
                    a=control:rtsp://192.168.0.1/video
                    a=recvonly
                    m=audio 0 RTP/AVP 0
                    a=control:rtsp://192.168.0.1/audio
                    a=recvonly
                    m=audio 0 RTP/AVP 0
                    a=control:rtsp://192.168.0.1/audioback
                    a=rtpmap:0 PCMU/8000
                    a=sendonly
```

This SDP file completely describes the RTSP session. The Server gives the client its control URLs to setup the streams.

In the next step the client can setup the sessions:

```
Client – Server:   SETUP rtsp://192.168.0.1/video RTSP/1.0
                  Cseq: 2
                  Transport: RTP/AVP;unicast;client_port=4588-4589

Server – Client:  RTSP/1.0 200 OK
                  Cseq: 2
                  Session: 123124;timeout=60
                  Transport:RTP/AVP;unicast;client_port=4588-4589; server_port=6256-6257

Client – Server:  SETUP rtsp://192.168.0.1/audio RTSP/1.0
                  Cseq: 3
                  Session: 123124
                  Transport: RTP/AVP;unicast;client_port=4578-4579

Server – Client:  RTSP/1.0 200 OK
                  Cseq: 3
                  Session: 123124;timeout=60
                  Transport:RTP/AVP;unicast;client_port=4578-4579; server_port=6276-6277

Client – Server:  SETUP rtsp://192.168.0.1/audioback RTSP/1.0
                  Cseq: 4
                  Session: 123124
                  Transport: RTP/AVP;unicast;client_port=6296-6297
                  Require:www.onvif.org/ver20/backchannel

Server – Client:  RTSP/1.0 200 OK
                  Cseq: 4
                  Session: 123124;timeout=60
                  Transport:RTP/AVP;unicast;client_port=6296-6297; server_port=2346-2347
```

The third setup request establishes the audio backchannel connection.

In the next step the client starts the session by sending a PLAY request.

```
Client – Server:   PLAY rtsp://192.168.0.1 RTSP/1.0
Cseq: 5
Session: 123124
Require:www.onvif.org/ver20/backchannel
Server – Client:    RTSP/1.0 200 OK
Cseq: 5
Session: 123124;timeout=60
```

After receiving the OK response to the PLAY request the client can start sending audio data to the server. It shall not start sending data to the server before it has received the response.

The Require-header indicates that a special interpretation of the PLAY command is necessary. The command covers both starting of the video and audio stream from NVT to the client and starting the audio connection from client to server.

To terminate the session the client sends a TEARDOWN request.

```
Client – NVT:  TEARDOWN rtsp://192.168.0.1 RTSP/1.0
Cseq: 6
Session: 123124
Require:www.onvif.org/ver20/backchannel
NVT – Client:  RTSP/1.0 200 OK
Cseq: 6
Session: 123124
```

5.3.2.3 Describe example in case of backchannel support with multiple decoding capability

If a device supports multiple audio decoders as backchannel, it can signal such capability by listing multiple a=rtpmap fields illustrated as follows.

```
Client – Server:  DESCRIBE rtsp://192.168.0.1 RTSP/1.0
Cseq: 1
User-Agent: ONVIF Rtsp client
Accept: application/sdp
Require: www.onvif.org/ver20/backchannel
Server – Client:   RTSP/1.0 200 OK
Cseq: 1
Content-Type: application/sdp
Content-Length: xxx
v=0
o= 2890842807 IN IP4 192.168.0.1
s=RTSP Session with audiobackchannel
m=video 0 RTP/AVP 26
a=control:rtsp://192.168.0.1/video
a=recvonly
m=audio 0 RTP/AVP 0
a=control:rtsp://192.168.0.1/audio
a=recvonly
m=audio 0 RTP/AVP 0 97 98 99 100
a=control:rtsp://192.168.0.1/audioback
a=rtpmap:0 PCMU/8000
a=rtpmap:97 G726-16/8000
a=rtpmap:98 G726-24/8000
a=rtpmap:99 G726-32/8000
a=rtpmap:100 G726-40/8000
a=sendonly
```

5.3.3 Multicast streaming

If the client intents to send its data in multicast it uses the transport parameter in the SETUP request to tell the server the multicast address and port.

Note that multicast streaming for Audio Backchannel is outside of the scope of this specification.

5.3.3.1 Example: Multicast Setup

```
Client – Server:  SETUP rtsp://192.168.0.1/audioback RTSP/1.0
Cseq: 4
Session: 123124
Transport:RTP/AVP;multicast;destination=224.2.1.1;port=60000-60001;ttl=128
Require:www.onvif.org/ver20/backchannel
Server – Client:   RTSP/1.0 200 OK
Cseq: 4
Session: 123124;timeout=60
Transport:RTP/AVP;multicast;destination=224.2.1.1;port=60000-60001;ttl=128;mode=”PLAY”
```

5.4 Multitrack streaming

This section describes how to establish an RTSP stream containing mutiple video tracks between a client and a server. The connection handling refers to the Session Description Protocol (SDP) Grouping Framework [RFC 5888].

5.4.1 Group Attribute

The session-level attribute "group" tdifferent media streams together. Its format in SDP is described by the following ABNF:

group-attribute = "a=group:" semantics

*(SP identification-tag)

semantics = "LS" / "FID" / semantics-extension

semantics-extension = token; token is defined in [RFC 4566]

This section defines one standard semantics named llp synchronization (LS). Note, that audio video synchronization is out of scope of this section, but theis reused for multitrack streaming.

5.4.2 Media Stream Identification Attribute

The media attribute "media stream identification" is defined in this section, which is used to identify media streams within a session description. Its format in SDP [RFC 4566] is described by the following ABNF:

mid-attribute = "a=mid:" identification-tag

identification-tag = token; token is defined in [RFC 4566]

The identification-tag should be unique within an SDP session description.

5.4.3 Extension Attribute

The SDP returned by the RTSP describes command shall include for each video stream a profile reference pointing to an exsiting media profile. It allows clients to map tracks to profiles for retrieve additional stitching information.

The tag shall use the following format:

a=x-onvif-profile:<ProfileReference>

5.4.4 Example

This section provides as example a camera with three sensors. For this example, the SDP information sent to client uses the Real Time Streaming Protocol (RTSP).

When the client sends the describle command to server, the server generates the response information and the track to be sent in one session will be chosen. According to [RFC 5888], the fields “group” and “mid” are provided in the sdp information. The attribute “group” represents that the following “mid” stream can be sent in one session. To clearly notify the client which URL uses for setup, we use the attribute “control” with absolute path.

```
Client – Server:  DESCRIBE rtsp://192.168.0.1 RTSP/1.0
Cseq: 1
User-Agent: ONVIF Rtsp client
Accept: application/sdp
Server – Client:  RTSP/1.0 200 OK
                  Cseq: 1
                  Content-Type: application/sdp
                  Content-Length: xxx
                  v=0
                  o=- 2890844256 2890842807 IN IP4 192.168.0.1
                  s=RTSP Session
                  a=group: LS 1 2 3
                  m=video 0 RTP/AVP 97
                  a=control:rtsp://192.168.0.1/video/trackID=1
                  a=ssrc:1234
                  a=mid:1
                  a=x-onvif-profile:ProfileToken1
                  m=video 0 RTP/AVP 98
```

```
a=control:rtsp://192.168.0.1/video/trackID=2
a=ssrc:2345
a=mid:2
a=x-onvif-profile:ProfileToken2
m=video 0 RTP/AVP 99
a=control:rtsp://192.168.0.1/video/trackID=3
a=mid:3
a=ssrc:5678
a=x-onvif-profile:ProfileToken3
```

5.5 Error handling

RTSP and HTTP protocol errors are classified into different categories (for example, status codes 1xx, 2xx, 3xx, 4xx and 5xx respectively). The device and the client shall support and handle these status codes. For RTSP status code definitions refer to [RFC 2326], Section 11.0. For HTTP status code definitions refer HTTP/1.1 [RFC 2616],Section 10.0.

6 Playback

6.1 RTSP usage

The replay protocol is based on RTSP [RFC 2326]. However because RTSP does not directly support many of the features required by CCTV applications, this standard defines several extensions to the protocol; these are detailed below.

This standard makes the following stipulations on the usage of RTSP:

1. RTP/RTSP/HTTP/TCP shall be supported by the server. This is the same transport protocol as a device that implements media streaming through the media service shall support, and the same requirements shall apply to replay streaming.

2. The server shall support the unicast RTP/UDP transport for streaming.

3. Clients should use a TCP-based transport for replay, in order to achieve reliable delivery of media packets.

4. The server MAY elect not to send RTCP packets during replay. In typical usage RTCP packets are not required, because usually a reliable transport will be used, and because absolute time information is sent within the stream, making the timing information in RTCP sender reports redundant.

6.2 RTSP describe

The SDP returned by the RTSP describe command shall include the TrackReference for each track of the recording to allow a client to map the tracks presented in the SDP to tracks of the recording. The tag shall use the following format:

a:x-onvif-track:<TrackReference>

For example:

```
NVS – NVT:  DESCRIBE rtsp://192.168.0.1 RTSP/1.0
            Cseq: 1
            User-Agent: ONVIF Rtsp client
            Accept: application/sdp

NVT – NVS:  RTSP/1.0 200 OK
            Cseq: 1
            Content-Type: application/sdp
            Content-Length: xxx
            v=0
            o= 2890842807 IN IP4 192.168.0.1
```

```
          m=video 0 RTP/AVP 26
           a=control:rtsp://192.168.0.1/video
 a=x-onvif-track:VIDEO001
 m=audio 0 RTP/AVP 98
 a=control:rtsp://192.168.0.1/audio
 a=x-onvif-track:AUDIO001
```

6.3 RTP header extension

In order to allow clients to report a stable and accurate timestamp for each frame played back regardless of the direction of playback, it is necessary to associate an absolute timestamp with each packet, or each group of packets with the same RTP timestamp (e.g. a video frame). This is achieved using an RTP header extension containing an NTP timestamp and some additional information also useful for replay.

The replay mechanism uses the extension ID 0xABAC for the replay extension.

Below shows the general form of an RTP packet containing this extension:

Table 3: RTP packet layout

V=2	P	X=1	CC	M	PT	sequence number
timestamp						
synchronization source (SSRC) identifier						
0xABAC						length=3
NTP timestamp…						
...NTP timestamp						
C	E	D	T	mbz	Cseq	padding
payload…						

The fields of this extension are as follows:

- NTP timestamp. An NTP [RFC 1305] timestamp indicating the absolute UTC time associated with the access unit.
- C: 1 bit. Indicates that this access unit is a synchronization point or “clean point”, e.g. the start of an intra-coded frame in the case of video streams.
- E: 1 bit. Indicates the end of a contiguous section of recording. The last access unit in each track before a recording gap, or at the end of available footage, shall have this bit set. When replaying in reverse, the E flag shall be set on the last frame at the end of the contiguous section of recording.
- D: 1 bit. Indicates that this access unit follows a discontinuity in transmission. It is primarily used during reverse replay; the first packet of each GOP has the D bit set since it does not chronologically follow the previous packet in the data stream (see section 6.6).
    - T: 1 bit. Indicates that this is the terminal frame on playback of a track. A device should signal this flag in both forward and reverse playback whenever no more data is available for a track.
- mbz: This field is reserved for future use and must be zero.
- Cseq: 1 byte. This is the low-order byte of the Cseq value used in the RTSP PLAY command that was used to initiate transmission. When a client sends multiple, consecutive PLAY commands, this value may be used to determine where the data from each new PLAY command begins.

The replay header extension shall be present in the first packet of every access unit (e.g. video frame).

6.3.1 NTP Timestamps

The NTP timestamps in the RTP extension header shall correspond to the wallclock time as measured at the original frame grabber before encoding of the stream.

For forward playback of I and P frames the NTP timestamps in the RTP extension header shall increase monotonically over successive packets within a single RTP stream.

6.3.2 Compatibility with the JPEG header extension

The replay header extension may co-exist with the header extension used by the JPEG RTP profile; this is necessary to allow replay of JPEG streams that use this extension. The JPEG extension is simply appended to the replay extension; its presence is indicated by an RTP header extension length field with a value greater than 3, and by the extension start codes of 0xFFD8 or 0xFFFF at the start of the fourth word of the extension content.

The following illustrates a JPEG packet that uses both extensions:

Table 4: RTP packet with JPEG header layout

V=2	P	X=1	CC	M	PT	sequence number
timestamp						
synchronization source (SSRC) identifier						
0xABAC						length=N+4
NTP timestamp…						
...NTP timestamp						
C	E	D	mbz	Cseq		padding
0xFFD8						jpeglength=N
extension payload: sequence of additional JPEG marker segments padded with 0xFF to the total extension length						
payload…						

6.4 RTSP Feature Tag

The Replay Service uses the "onvif-replay" feature tag to indicate that it supports the RTSP extensions described in this standard. This allows clients to query the server's support for these extensions using the Require header as described in [RFC 2326] section 5.3.1 .

Example:

```
C->S:   SETUP rtsp://server.com/foo/bar/baz.rm RTSP/1.0
           Cseq: 302
           Require: onvif-replay
S->C:   RTSP/1.0 551 Option not supported
           Cseq: 302
           Unsupported: onvif-replay
```

The Replay Server shall accept a SETUP and PLAY command that includes a Require header containing the onvif-replay feature tag.

6.5 Initiating Playback

Playback is initiated by means of the RTSP PLAY method. For example:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
```

The ReversePlayback capability defined in the ONVIF Replay Control Service Specification signals if a device supports reverse playback. Reverse playback is indicated using the Scale header field with a negative value. For example to play in reverse without no data loss a value of –1.0 would be used.

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
Scale: -1.0
```

If a device supports reverse playback it shall accept a Scale header with a value of –1.0. A device MAY accept other values for the Scale parameter. Unless the Rate-Control header is set to “no” (see below), the Scale parameter is used in the manner described in [RFC 2326]. If Rate-Control is set to “no”, the Scale parameter, if it is present, shall be either 1.0 or –1.0, to indicate forward or reverse playback respectively. If it is not present, forward playback is assumed.

6.5.1 Range header field

A device shall support the Range field expressed using absolute times as defined by [RFC 2326]. Absolute times are expressed using the utc-range from [RFC 2326].

Either open or closed ranges may be used. In the case of a closed range, the range is increasing (end time later than start time) for forward playback and decreasing for reverse playback. The direction of the range shall correspond to the value of the Scale header.

In all cases, the first point of the range indicates the starting point for replay.

The time itsel shall be given as

```
utc-range = "clock" ["=" utc-range-spec]
utc-range-spec = ( utc-time "-" [ utc-time ] ) / ( "-" utc-time )
utc-time = utc-date "T" utc-clock "Z"
utc-date = 8DIGIT
utc-clock = 6DIGIT [ "." 1*9DIGIT ]
```

as defined in [RFC2326].

Examples:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-20090615T115000Z
Rate-Control: no
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
```

```
Require: onvif-replay
Range: clock=20090615T115000.440Z-20090615T114900Z
Rate-Control: no
Scale: -1.0
```

6.5.2 Rate-Control header field

This specification introduces the Rate-Control header field, which may be either “yes” or “no”. If the field is not present, “yes” is assumed, and the stream is delivered in real time using standard RTP timing mechanisms. If this field is “no”, the stream is delivered as fast as possible, using only the flow control provided by the transport to limit the delivery rate.

The important difference between these two modes is that with “Rate-Control=yes”, the server is in control of the playback speed, whereas with “Rate-Control=no” the client is in control of the playback speed. Rate-controlled replay will typically only be used by non-ONVIF specific clients as they will not specify “Rate-Control=no”.

When replaying multiple tracks of a single recording, started by a single RTSP PLAY command and not using rate-control, the data from the tracks should be multiplexed in time in the same order as they were recorded.

An ONVIF compliant RTSP server shall support operation with “Rate-Control=no” for playback.

6.5.3 Frames header field

The Frames header field may be used to reduce the number of frames that are transmitted, for example to lower bandwidth or processing load. Three modes are possible:

1. Intra frames only. This is indicated using the value “intra”, optionally followed by a minimum interval between successive intra frames in the stream. The latter can be used to limit the number of frames received even in the presence of “I-frame storms” caused by many receivers requesting frequent I-frames.
2. Intra frames and predicted frames only. This is indicated using the value “predicted”. This value can be used to eliminate B-frames if the stream includes them.
3. All frames. This is the default.

Examples:

To request intra frames only:

```
Frames: intra
```

To request intra frames with a minimum interval of 4000 milliseconds:

```
Frames: intra/4000
```

To request intra frames and predicted frames only:

```
Frames: predicted
```

To request all frames (note that it is not necessary to explicitly specify this mode but the example is included for completeness):

```
Frames: all
```

The interval argument used with the “intra” option refers to the recording timeline, not playback time; thus for any given interval the same frames are played regardless of playback speed. The interval argument shall NOT be present unless the Frames option is “intra”.

The server shall support the Frames header field. This does not preclude the use of the Scale header field as an alternative means of limiting the data rate. The implementation of the Scale header field may vary between different server implementations, as stated by [RFC 2326].

An ONVIF compliant RTSP server shall support the Frames parameters “intra” and “all” for playback.

6.5.4 Synchronization points

The transmitted video stream shall begin at a synchronization point (see section “Synchronization Point” of the ONVIF Media Service Specificaton). The rules for choosing the starting frame are as follows:

- If the requested start time is within a section of recorded footage, the stream starts with the first clean point at or before the requested start time. This is the case regardless of playback direction.
- If the requested start time is within a gap in recorded footage and playback is being initiated in the forwards direction, the stream starts with the first clean point in the section following the requested start time.
- If the requested start time is within a gap in recorded footage and playback is being initiated in the reverse direction, the stream starts with the last clean point in the section preceding the requested start time.

6.6 Reverse replay

Reverse replay is initiated using the Scale header field with a negative value as described above.

6.6.1 Packet transmission order

The example in Figure 9 shows how frames are transmitted during normal forward playback for a recording with two short video clips each consisting of two GOPs. As shown all packets are transmitted in recording order. The last frame before a gap is marked with the E flag to signal a gap in the recording.

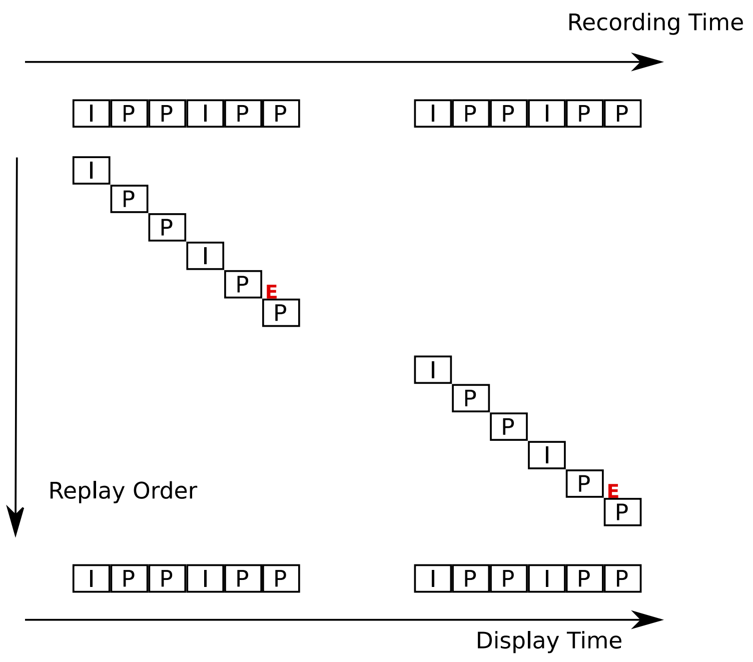

Figure 9: Packet transmission during forward playback

The order in which video packets are transmitted during reverse replay is based on GOPs, where a GOP consists of a clean point followed by a sequence of non-cleanpoint packets.

During reverse playback, GOPs shall be sent in reverse order, but packets within a GOP shall be sent in forward order. The first packet of each GOP shall have the “discontinuity” bit set in its RTP extension header. The last frame of a GOP immediately following a gap (or the beginning of available footage) shall have the E bit set in its RTP extension header.

When transmitting only key frames, or when the codec is not motion-based (e.g. JPEG), a GOP is considered to consist of a single frame, but may still be composed of multiple packets. In this case the packets within each frame shall be again sent in forward order, while the frames themselves shall be sent in reverse order.

Audio and metadata streams MAY be transmitted in an order mirroring that of the video stream. Thus packets from these streams are sent in forward playback order until the occurrence of a packet (generally a video packet) with the D bit set in the extension header, at which point they jump back to a point before the discontinuity.

Note that reverse playback of Audio packet isn't useful. Threfore Audio packets should generally not be transmitted during reverse playback.

The example of Figure 10 shows for the same recording as depicted in Figure 9 how packets are transmitted during reverse forward playback. As shown all packets are transmitted in recording order.

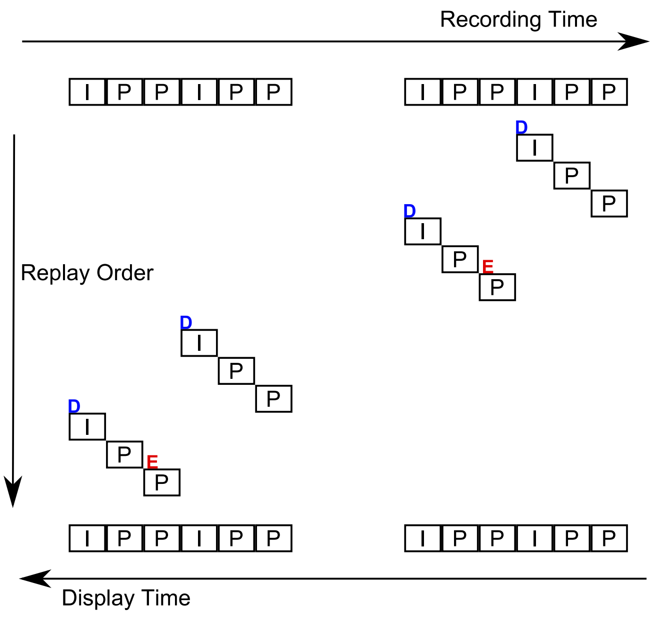

Figure 10: Packet transmission during reverse playback

6.6.2 RTP sequence numbers

The RTP sequence numbers of packets transmitted during reverse playback shall increment monotonically *in the order of delivery*, not in the intended order of playback.

6.6.3 RTP timestamps

The use of RTP timestamps depends on the value of the Rate-Control header. If the value of this header is "no" (i.e. the client controls playback speed), the RTP timestamps are derived from the original sampling times of the recorded frames. If the Rate-Control header is not present or has the value "yes" (i.e. the server controls playback speed), the RTP timestamps correspond to playback timing as described in [RFC 2326] Appendix B.

If Rate-Control is "no", the RTP timestamps of packets transmitted during reverse playback shall be the same as they would be if those same packets were being transmitted in the forwards direction. Unlike the sequence numbers, the RTP timestamps correspond to the original recording order, not the delivery order. The server MAY use the same RTP timestamps that were originally received when the stream was recorded.

This means that successive RTP packets of a single GOP will always have increasing RTP timestamps (see transmission order above), but that the timestamp on index frames of successively received GOPs will decrease during reverse replay.

If Rate-Control is "yes", the RTP timestamps of packets transmitted during reverse playback shall indicate the times at which each frame should be rendered at the client. Thus successive packets of a single GOP will have *decreasing* RTP timestamps (since the first one delivered should be played last), and the timestamps on index frames will *increase*. In this mode the interval between successive timestamps depends on the values of the Speed and Scale headers, as described in [RFC 2326] Appendix B.

Note that strict decreasing order of RTP timestamps does not apply for GOPs with B-Frames.

6.7 RTSP Keepalive

When rate control is disabled and the RTP stream is tunneled through the RTSP connection (i.e. using the RTP/RTSP/TCP or RTP/RTSP/HTTP/TCP transports), the client must be aware that it may not be able to receive the response to any request if for example replay is paused.

6.8 Currently recording footage

If the client commences playback from the current real world time or shortly before it, it can end up playing footage in real time as it is being recorded. In this event the server simply continues to send stream data to the client as it receives it.

Note that the E bit is not set on access units currently being recorded even though each access unit sent to the replay client will typically be the last one known to the server. If recording stops however, the E bit is set on the last access unit of the recording.

6.9 End of footage

If playback reaches a point after which there is no further data in one or more of the streams being sent, it stops transmitting data but does not enter the "paused" state. If the server resumes recording after this has happened, delivery will resume with the new data as it is received.

6.10 Go To Time

As stated in [RFC 2326] section 10.5, a PLAY command received when replay is already in progress will not take effect until the existing play operation has completed. This specification adds a new RTSP header, "Immediate", which overrides this behaviour for the PLAY command that it is used with:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
Immediate: yes
```

If the server receives a PLAY command with the Immediate header set to "yes", it will immediately start playing from the new location, cancelling any existing PLAY command. The first packet sent from the new location shall have the D (discontinuity) bit set in its RTP extension header.

An ONVIF compliant RTSP server shall support the immediate header field for playback with value "yes". The behavior without immediate header or value "no" is not defined by the specification.

6.11 Use of RTCP

A server is not required to send RTCP packets. If it does send them, the following rules apply:

If Rate Control is enabled (see section 6.5.2), RTCP packets shall be constructed and transmitted as specified in [RFC 3550]. In particular, the NTP timestamp in a sender report indicates the current wallclock time, and is not related to the NTP timestamps embedded in the RTP extension headers in the data streams.

If Rate Control is not enabled, both the NTP timestamp and RTP timestamp in each sender report shall be set to zero.

7 WebSocket transport for RTP/RTSP/TCP

WebSocket protocol basically provides bidirectional communication between Client and Server for web application. This section describes how a WebSocket connection can be established between a client and a server. The connection handling is done using WebSocket [RFC 6455].

The WebSocket transport mechanism explained in the Figure 11 is common for live streaming and playback. The RTP/RTSP/TCP data transmitted by the device over the WebSocket shall follow the device requirements as explained in Section 5 Live Streaming and Section 6 Playback.

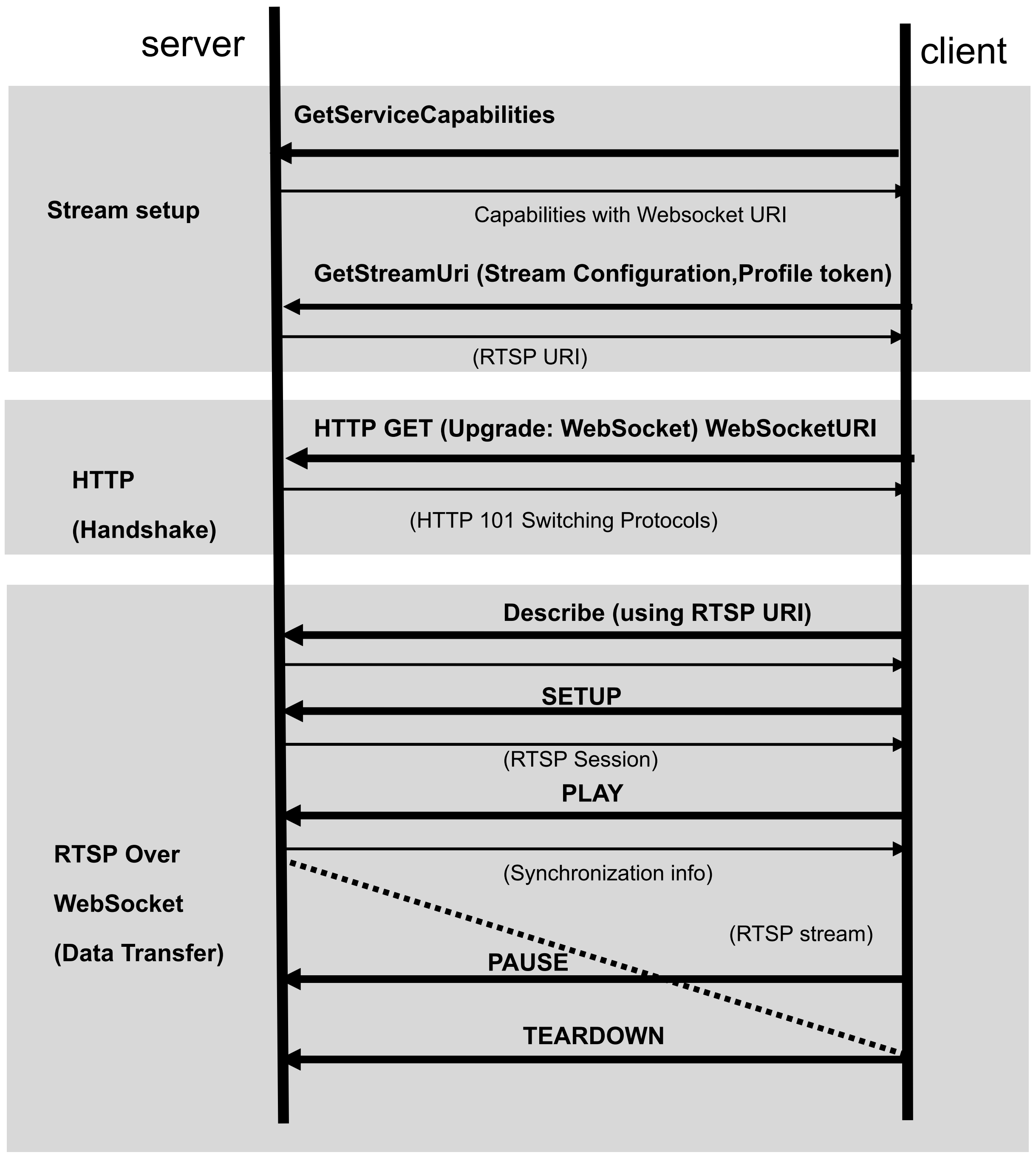
server
client
Stream setup
GetServiceCapabilities
Capabilities with Websocket URI
GetStreamUri (Stream Configuration,Profile token)
(RTSP URI)
HTTP
(Handshake)
HTTP GET (Upgrade: WebSocket) WebSocketURI
(HTTP 101 Switching Protocols)
RTSP Over
WebSocket
(Data Transfer)
Describe (using RTSP URI)
SETUP
(RTSP Session)
PLAY
(Synchronization info)
(RTSP stream)
PAUSE
TEARDOWN

Figure 11: Streaming over WebSocket

7.1 WebSocket version

The Device shall support WebSocket protocol version 13.

7.2 Authentication

The device shall follow RTSP authentication requirements as explained in Section 5.12.1 Authentication of ONVIF Core Specification. The access class of streams shall be READ_MEDIA.

7.3 WebSocket Connection

The initial connection setup is done in two steps:

1. Handshake

2. Data Transfer

7.3.1 Handshake

7.3.1.1 WebSocket Subprotocol

The device shall support the WebSocket subprotocol as defined in Section 1.9 of [RFC 6455]. The websocket sub protocol defined by this specification is identified with 'rtsp.onvif.org'. A device shall support the 'rtsp.onvif.org' sub protocol.

7.3.1.2 Example: WebSocket Handshake

This example shows the message transfer between an Web client (client) and an Web server (server). The client requests server to initiate a WebSocket connection using the WebSocket URI. The WebSocket Uri can be retrieved using the GetServiceCapabilities command of the ONVIF Media2 Service. An example WebSocket URI may look like, “ws:/webSocketServer”

```
client->server:   GET /webSocketServer HTTP/1.1
Host: 192.168.0.1
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: rtsp.onvif.org
Sec-WebSocket-Version: 13
server->client: HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: rtsp.onvif.org
```

7.3.2 Data Transfer

After a successful handshake, client and server can transfer bidirectional data over WebSocket connection.

Device supporting live streaming and playback streaming over WebSocket provides the WebSocket streaming URI in the GetServiceCapabilities of Media2 Service Section 5.11 and Replay Service Section 5.4.1 respectively.

7.3.2.1 WebSocket Message Frame Format

The basic units of data transmitted over WebSocket connection are referred as “messages” in the RFC 6455. The messages are composed of one or more frames and each frame has an associated data type. There are different type of frames like text frames and data frames.

The device shall only use data frames for transmitting the RTP/RTSP/TCP data over the WebSocket.

8 MulticastAudioDecoder SRTP Packet Format

The following diagram illustrates the structure of an SRTP packet used by the `MulticastAudioDecoder`. It includes the RTP header, a header extension carrying the Roll-Over Counter (ROC), the encrypted

RTP payload, and the SRTP authentication tag. The header extension format adheres to the one-byte extension mechanism defined in RFC 8285. Encryption and authentication are handled via SRTP, as specified in RFC 3711 for AES_CM_128_HMAC_SHA1_80, and in RFC 7714 for AEAD_AES_128_GCM and AEAD_AES_256_GCM modes.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       Sequence Number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Synchronization Source (SSRC) Identifier            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Contributing Source (CSRC) Identifiers             |  <- Optional, 0 to 15 items
|                             ...                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Header Extension ID (0xBEDE) |     Length (in 32-bit words)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  ID=1 | L=3   | ROC byte 1    | ROC byte 2    | ROC byte 3    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ROC byte 4    | Padding (if needed)                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Encrypted RTP Payload                     |
|                             ...                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     SRTP Authentication Tag                   |
|                             ...                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Diagram of SRTP packet format with ROC extension

Annex A. Revision History

Rev.	Date	Editor	Changes
2.1	Jul-2011	Hans Busch	Split from Core 2.0 Change Requests 201, 225
2.1.1	Jan-2012	Hans Busch	Change Requests 282, 283, 284, 296, 289, 290, 264, 316, 333, 549
2.2	May-2012	Hans Busch	Change Request 613
2.2.1	Dec-2012	Michio Hirai	Change Requests 768, 878
2.3	May-2013	Michio Hirai	Change Request 1051
2.4.1	Dec-2013	Michio Hirai	Change Request 1285
2.4.2	Jun-2014	Michio Hirai	Change Request 1323
2.5	Dec-2014	Hans Busch, Michio Hirai	Added support for gzip compressed metadata Change Request 1443
2.6	Jun-2015	Michio Hirai	Change Request 1571, 1605, 1624
16.06	Jun-2016	Hiroyuki Sano	Change Request 1717, 1725, 1852
16.12	Dec-2016	Sujith Raman, Hans Busch, Hiroyuki Sano	Added RTSP over WebSocket Change Request 1886, 1888, 1890
17.06	Jun-2017	Hiroyuki Sano	Change Request 1843, 1954, 1997, 2084, 2085, 2128
17.12	Dec-2017	Hiroyuki Sano	Change Request 2165, 2180, 2181, 2206, 2210
18.06	Jun-2018	Hiroyuki Sano	Change Request 2229, 2247, 2254, 2275, 2284, 2285
19.06	Jun-2019	Hiroyuki Sano	Change Request 2446, 2464
19.12	Dec-2019	Dora Han	Added Multitrack Streaming
21.06	Jun-2021	Hans Busch	Improve section on timeout and keep-alive handling.
21.12	Dec-2021	Hans Busch	Remove obsolete exi default compression.
22.12	Dec-2022	Hans Busch	Add secure RTSPS streaming.
23.06	Jun-2023	Hans Busch	Update description and figure to remove emphasis on SET_PARAMETER.
25.12	Dec-2025	Sujith Raman	Add support for multicast audio decoder.