SONY

Technical Guide | G6

# RTSP Streaming

Network Camera
October 26, 2012
G6TG005
Revision 1.0.0

IPELA

# CONTENTS

# 1. Overview

## 1.1. About This Document

This document describes technical details about RTSP streaming function supported by the 6th generation (the G6) of Sony Network Cameras (hereafter referred to as “the cameras”).

## 1.2. About CGI Version

If you would like to identify whether your camera belongs to G6 or not, please kindly use the CGI command below.

```
http://camera_address/command/inquiry.cgi?inq=system
```

In case of G6 models, the following description is in the response.

```
CGIVersion=6.*. *.*.*                    (Where each * is a number.)
```

Otherwise, this parameter does not exist.

# 2. Supported Codecs

The following codecs are supported with RTSP streaming function of the cameras.

| Video Codec |
| --- |
| H.264 |
| Motion JPEG |

| Audio Codec |
| --- |
| G.711 |
| G.726 |
| AAC |

* Some cameras may not support some of the codecs with RTSP streaming function depending on capabilities of each model/series.

For details about the codecs listed above, please refer to relevant documents such as “User’s Guide” or “CGI Commands Technical Guide”.

# 3. RTSP Request URL

RTSP request URLs of the cameras for getting live stream are as follows.

| Request URL | Description |
|---|---|
| `rtsp://<camera_address>/video1` | Requests video* bit stream corresponding to the "`ImageCodec1`"** and its related CGI parameters.<br>* Audio bit stream can be transmitted together with this video stream depending on situations. See chapter 5 "Getting Stream".<br>** `ImageCodec1` corresponds to the "Image1" settings in administrator menu of the cameras. |
| `rtsp://<camera_address>/video2` | Requests video* bit stream corresponding to the "`ImageCodec2`"** and its related CGI parameters.<br>* Audio bit stream can be transmitted together with this video stream depending on situations. See chapter 5 "Getting Stream".<br>** `ImageCodec2` corresponds to the "Image2" settings in administrator menu of the cameras. |
| `rtsp://<camera_address>/video3` | Requests video* bit stream corresponding to the "`ImageCodec3`"** and its related CGI parameters.<br>* Audio bit stream can be transmitted together with this video stream depending on situations. See chapter 5 "Getting Stream".<br>** `ImageCodec3` corresponds to the "Image3" settings in administrator menu of the cameras. |
| `rtsp://<camera_address>/audio` | Requests audio bit stream corresponding to the "`AudInCodec`"* and its related CGI parameters.<br>* `AudInCodec` corresponds to the "Audio sending" settings in administrator menu of the cameras. |

* Total number of codec instances in the camera may vary depending on capabilities of each model/series.

RTSP port of the camera (RTSP server) is 554 by factory default. If needed, the port can be changed by using "`camera.cgi`" CGI command with "`RTSPPort`" CGI parameter.

For details about the each live stream listed above (such as "what the CGI parameter `ImageCodec1` is", "how to configure the Image1 settings" and so on), please refer to the relevant documents — "CGI Commands Technical Guide" and "User's Guide".

Regarding request URLs for SSM (source-specific multicast), please refer to chapter 7 "Source-Specific Multicast".

# 4. RTSP Methods

## 4.1. Supported Methods

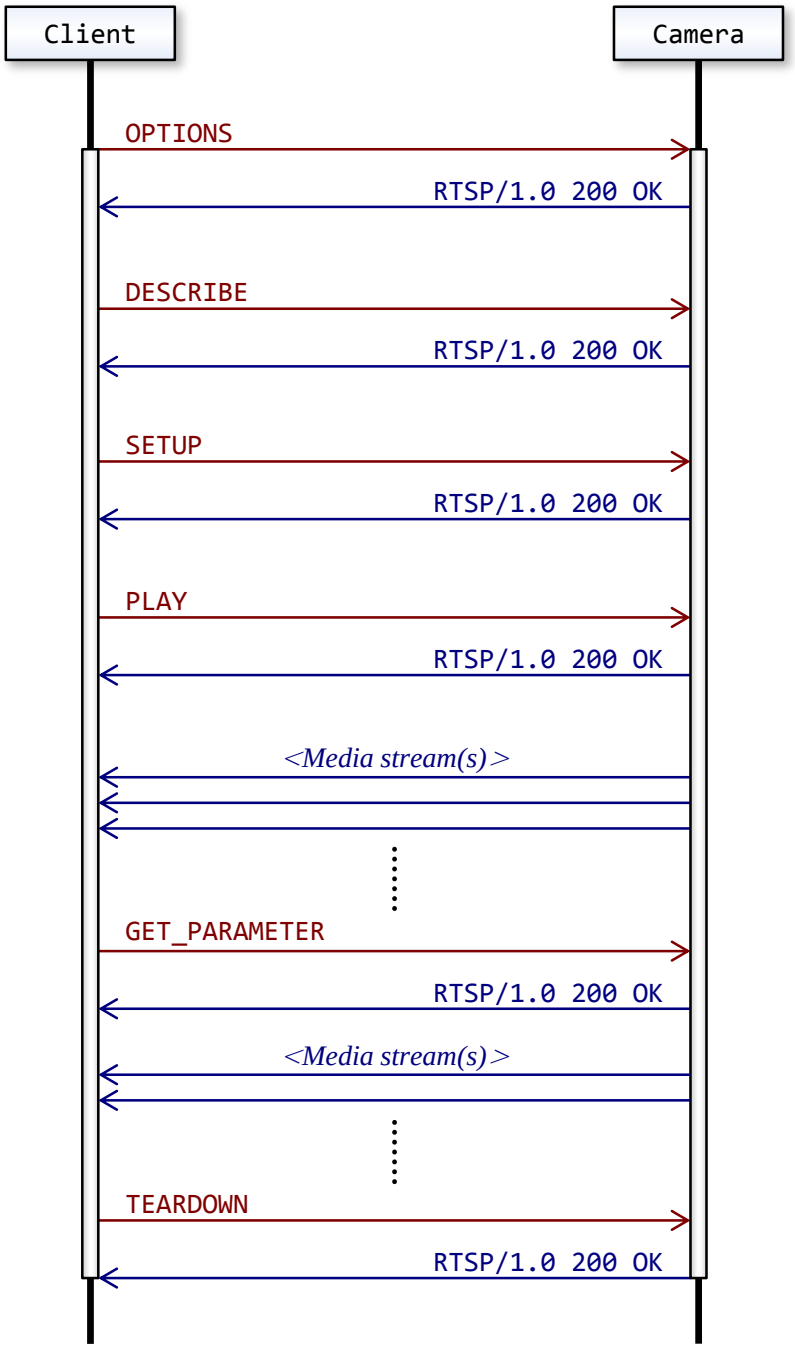
The cameras support the following RTSP methods.

| Supported Method |
| --- |
| OPTIONS |
| DESCRIBE |
| SETUP |
| PLAY |
| PAUSE *(To be supported)* |
| TEARDOWN |
| GET_PARAMETER |
| SET_PARAMETER |

For details about the RTSP methods listed above, please refer to RFC 2326.

## 4.2. Typical Sequence of RTSP Communication

Overview of a typical RTSP communication sequence between the camera (RTSP server) and a client is as shown below.




RTSP "GET_PARAMETER" method in the sequence above is used to keep the RTSP streaming alive. Please refer to chapter 5 "Getting Stream" for further descriptions.

# 5. Getting Stream

### < Transfer Protocols >

RTSP function of the cameras supports the following transfer protocols to stream video and/or audio from the camera to client(s).

**a)** TCP bit stream
**b)** UDP unicast bit stream
**c)** UDP multicast bit stream

Details of each case above are described in the following sections.

### < Number of Media Streams >

The cameras support multiple codec instances simultaneously as mentioned in the previous chapter 3 "RTSP Request URL". Number of media streams in an RTSP session of the camera varies depending on situations as described below.

**Case 1)** In a situation where audio codec is disabled and a video stream is requested:
Each RTSP session delivers just one video stream instances at a time.
(Examples of this case are deeply described in section 5.1 "Getting Video Stream" below.)

**Case 2)** In a situation where audio codec is enabled and an audio stream is requested:
Each RTSP session delivers just one audio stream at a time.
(This case is described in section 5.2 "Getting Audio Stream" below.)

**Case 3)** In a situation where audio codec is enabled and just a video stream is requested:
Each RTSP session delivers two media streams at a time — not only a video stream but also an audio stream.
(Example of this case is deeply described in section 5.3 "Getting Both Video and Audio Bit Streams" below.)

### < RTSP Session Timeout >

RTSP session timeout value of the cameras – "`RTSPTimeout`" CGI parameter in seconds – is set to zero by factory default. The cameras treat the value zero as infinite duration and RTSP session never timeout. To set `RTSPTimeout` to an arbitrary duration, "`camera.cgi`" CGI command with `RTSPTimeout` CGI parameter (in seconds) can be used. Please refer to relevant document "CGI

Commands Technical Guide" for details about `camera.cgi` and `RTSPTimeout`.

RTSP session timeout value of the camera is indicated to a client as "`timeout`" parameter (in seconds) in RTSP response to "`SETUP`" request. However, when the timeout value is set to zero, `timeout` parameter does not exist in the response.

To keep an RTSP streaming alive, examples in this document use RTSP "`GET_PARAMETER`" method before the camera automatically closes RTSP session in accordance with `timeout` pamameter.

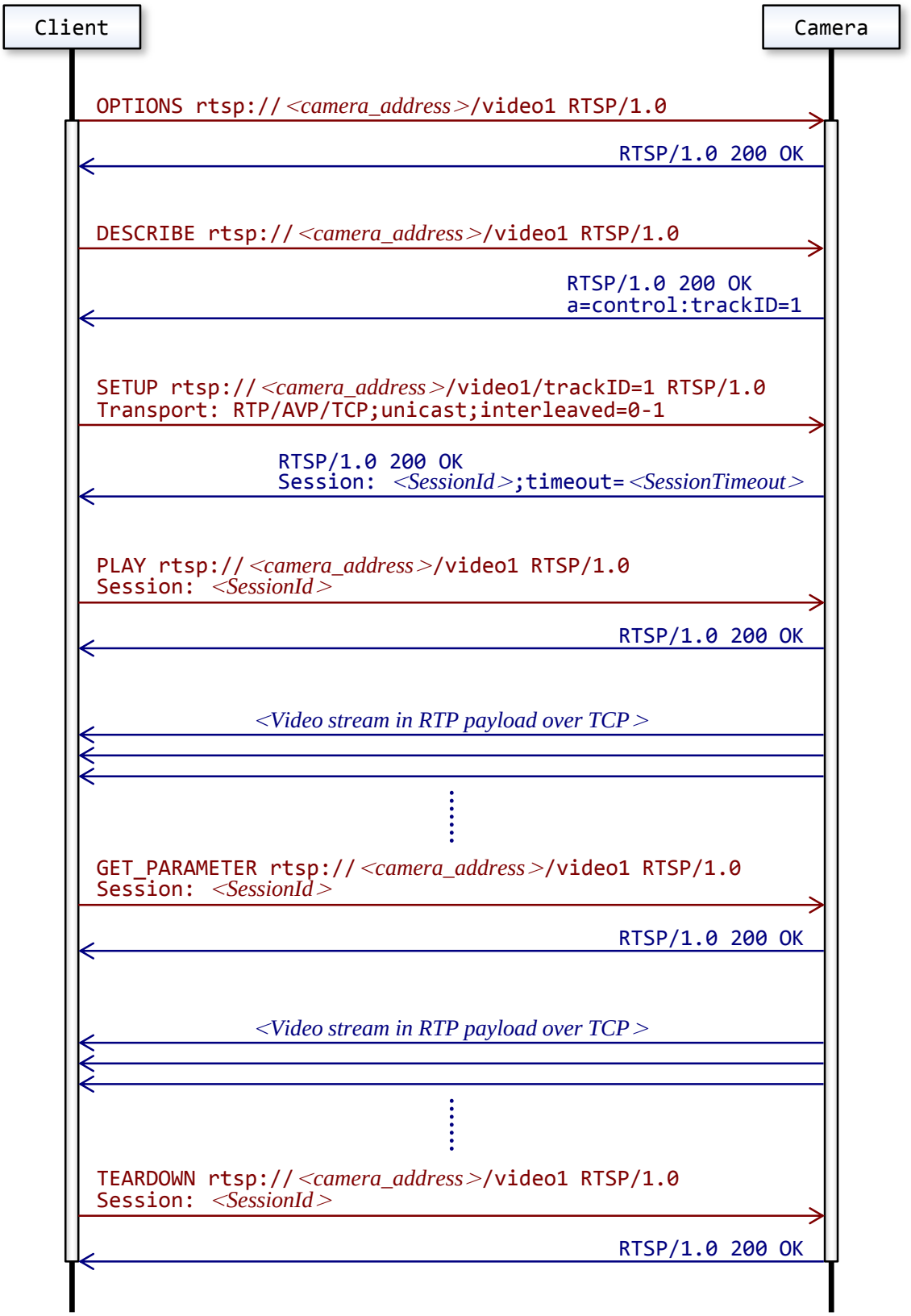
### < Closing RTSP Session >

To safely close an RTSP session and its related UDP ports invoked by it (RTP, RTCP), the cameras require clients to use RTSP "`TEARDOWN`" method.
Technically sending RTCP "`BYE`" packet from a client to the camera might trigger closing all these session/ports in the camera eventually, however, the cameras do not recommend using RTCP "`BYE`" packet in this case.

## 5.1. Getting Video Stream

### 5.1.1. TCP Bit Stream (Video)

The following diagram and captured packets in this section show an example of getting a video bit stream from the camera over TCP in a situation where audio codec is disabled and a client requests a video stream.


Client
Camera
OPTIONS rtsp://<camera_address>/video1 RTSP/1.0
RTSP/1.0 200 OK
DESCRIBE rtsp://<camera_address>/video1 RTSP/1.0
RTSP/1.0 200 OK
a=control:trackID=1
SETUP rtsp://<camera_address>/video1/trackID=1 RTSP/1.0
Transport: RTP/AVP/TCP;unicast;interleaved=0-1
RTSP/1.0 200 OK
Session: <SessionId>;timeout=<SessionTimeout>
PLAY rtsp://<camera_address>/video1 RTSP/1.0
Session: <SessionId>
RTSP/1.0 200 OK
<Video stream in RTP payload over TCP>
GET_PARAMETER rtsp://<camera_address>/video1 RTSP/1.0
Session: <SessionId>
RTSP/1.0 200 OK
<Video stream in RTP payload over TCP>
TEARDOWN rtsp://<camera_address>/video1 RTSP/1.0
Session: <SessionId>
RTSP/1.0 200 OK

```
OPTIONS rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 1\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 1\r\n
Public: DESCRIBE, SETUP, TEARDOWN, PLAY, OPTIONS, SET_PARAMETER, GET_PARAMETER\r\n
\r\n

DESCRIBE rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 2\r\n
User-Agent <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 2\r\n
Content-Length: <Length>\r\n
Content-Type: application/sdp\r\n
Content-Base: rtsp://<camera_address>/video1/\r\n
\r\n

v=<ProtocolVersion>\r\n
o=- <SessionIdForOrigin> 1 IN IP4 <camera_address>\r\n
s=<SessionName>\r\n
t=0 0\r\n
a=range:npt=now-\r\n
c=IN IP4 <ConnectionAddress>\r\n
m=<MediaNameAndTransportAddress>\r\n
a=rtpmap:<PayloadType> <EncodingName>/<ClockRate>\r\n
a=control:trackID=1\r\n
a=framerate:<FrameRate>\r\n
a=fmtp:<Format> <FormatSpecificParameters>\r\n
\r\n

SETUP rtsp://<camera_address>/video1/trackID=1 RTSP/1.0\r\n
CSeq: 3\r\n
Transport: RTP/AVP/TCP;unicast;interleaved=0-1\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 3\r\n
Session: <SessionId>[;timeout=<SessionTimeout>]\r\n
Cache-Control: must-revalidate\r\n
Transport: RTP/AVP/TCP;interleaved=0-1;ssrc=<SSRC>\r\n
```

```
\r\n

PLAY rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 4\r\n
Session: <SessionId>\r\n
Range: npt=0.000-\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 4\r\n
Session: <SessionId>\r\n
RTP-Info: url=trackID=1;seq=<SequenceNumber>;rtptime=0\r\n
\r\n

<Video stream in RTP payload over TCP>

GET_PARAMETER rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 5\r\n
Session: <SessionId>\r\n
User-Agent: <UA>\r\n
\r\n

<Video stream in RTP payload over TCP>

TEARDOWN rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 6\r\n
Session: <SessionId>\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 6\r\n
Session: <SessionId>\r\n
```

### 5.1.2. UDP Unicast Bit Stream (Video)

The following diagram and captured packets in this section show an example of getting video bit stream from the camera over UDP unicast in a situation where audio codec is disabled and a client requests a video stream.

In the example below in this section, client uses "`client_port`" parameter in RTSP "`SETUP`" request to specify UDP unicast ports of the client's end for RTP session and RTCP session. One of the following ways can be used to specify the ports.

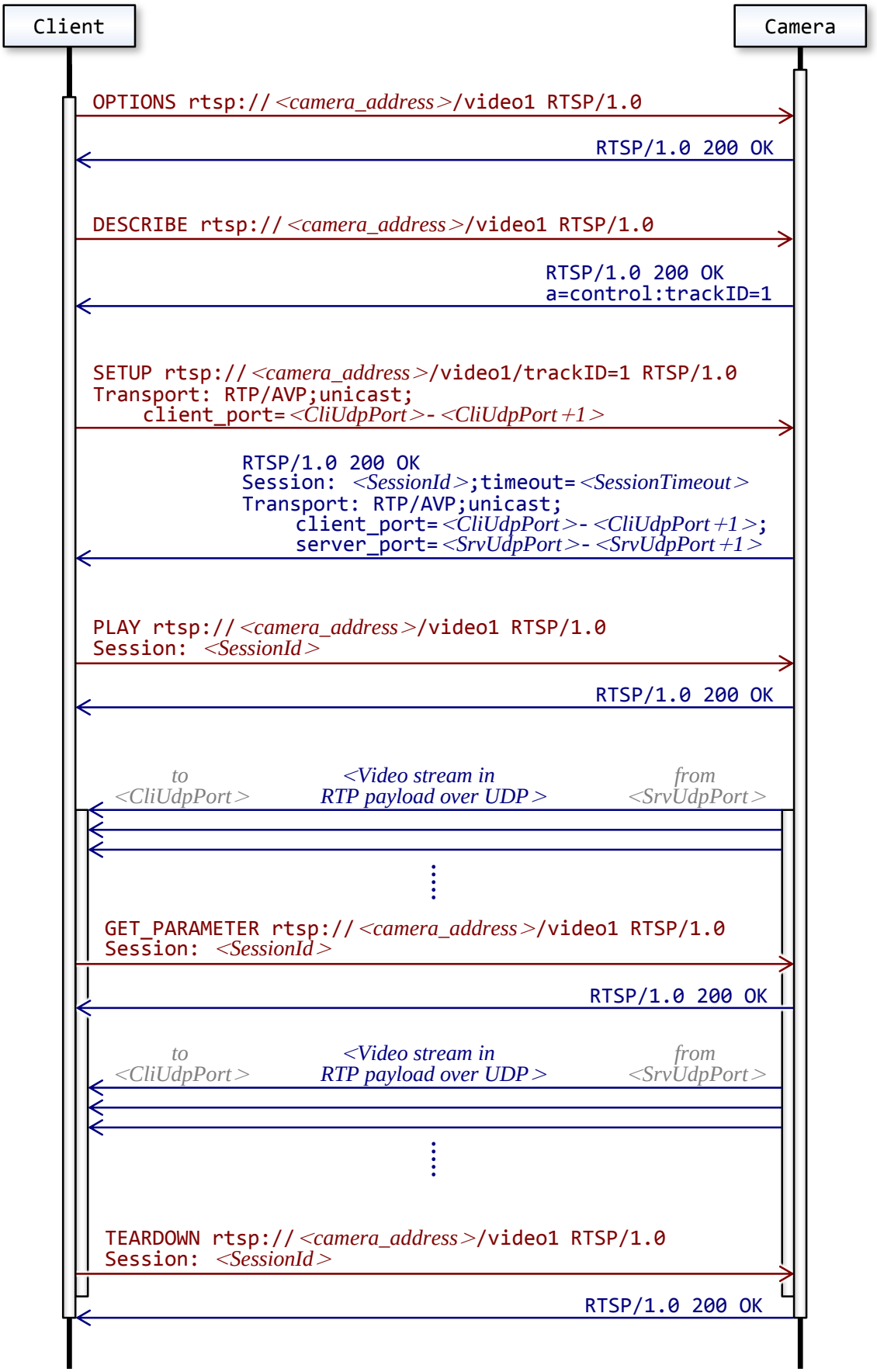
**a)** Specifying ports by using "`client_port`" parameter in RTSP "`SETUP`" request — This is used in the example mentioned above.

**b)** Specifying ports by using "`camera.cgi`" CGI command with CGI parameters listed below.

| CGI Parameter | Corresponding to |
|---|---|
| `RTSPUcVideoPort1` | A pair of UDP unicast ports for the "`ImageCodec1`" live stream |
| `RTSPUcVideoPort2` | A pair of UDP unicast ports for the "`ImageCodec2`" live stream |
| `RTSPUcVideoPort3` | A pair of UDP unicast ports for the "`ImageCodec3`" live stream |
| `RTSPUcAudioPort` | A pair of UDP unicast ports for the "`AudInCodec`" live stream |

For details about them, please refer to relevant document — "CGI Commands Technical Guide".

**c)** Not explicitly specifying UDP unicast ports — In this case, the ports are automatically specified by the camera (RTSP server) and are indicated in RTSP response to "`SETUP`" request.


Client
Camera
OPTIONS rtsp://<camera_address>/video1 RTSP/1.0
RTSP/1.0 200 OK
DESCRIBE rtsp://<camera_address>/video1 RTSP/1.0
RTSP/1.0 200 OK
a=control:trackID=1
SETUP rtsp://<camera_address>/video1/trackID=1 RTSP/1.0
Transport: RTP/AVP;unicast;
client_port=<CliUdpPort>-<CliUdpPort+1>
RTSP/1.0 200 OK
Session: <SessionId>;timeout=<SessionTimeout>
Transport: RTP/AVP;unicast;
client_port=<CliUdpPort>-<CliUdpPort+1>;
server_port=<SrvUdpPort>-<SrvUdpPort+1>
PLAY rtsp://<camera_address>/video1 RTSP/1.0
Session: <SessionId>
RTSP/1.0 200 OK
to <CliUdpPort>
<Video stream in RTP payload over UDP>
from <SrvUdpPort>
GET_PARAMETER rtsp://<camera_address>/video1 RTSP/1.0
Session: <SessionId>
RTSP/1.0 200 OK
to <CliUdpPort>
<Video stream in RTP payload over UDP>
from <SrvUdpPort>
TEARDOWN rtsp://<camera_address>/video1 RTSP/1.0
Session: <SessionId>
RTSP/1.0 200 OK

```
OPTIONS rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 1\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 1\r\n
Public: DESCRIBE, SETUP, TEARDOWN, PLAY, OPTIONS, SET_PARAMETER, GET_PARAMETER\r\n
\r\n

DESCRIBE rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 2\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 2\r\n
Content-Length: <Length>\r\n
Content-Type: application/sdp\r\n
Content-Base: rtsp://<Address>/video1/\r\n
\r\n

v=0\r\n
o=- <SessionIdForOrigin> 1 IN IP4 <camera_address>\r\n
s=<SessionName>\r\n
t=0 0\r\n
a=range:npt=now-\r\n
c=IN IP4 <ConnectionAddress>\r\n
m=<MediaNameAndTransportAddress>\r\n
a=rtpmap:<PayloadType> <EncodingName>/<ClockRate>\r\n
a=control:trackID=1\r\n
a=framerate:<FrameRate>\r\n
a=fmtp:<Format> <FormatSpecificParameters>\r\n
\r\n

SETUP rtsp://<camera_address>/video1/trackID=1 RTSP/1.0\r\n
CSeq: 3\r\n
Transport: RTP/AVP;unicast;client_port=<CliUdpPort>-<CliUdpPort+1>\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 3\r\n
Session: <SessionId>[;timeout=<SessionTimeout>]\r\n
Cache-Control: must-revalidate\r\n
Transport: RTP/AVP;unicast;client_port=<CliUdpPort>-<CliUdpPort+1>;source=<Sou
```

*rceAddress*>;server_port=<*SrvUdpPort*>-<*SrvUdpPort+1*>;ssrc=<*SSRC*>\r\n
\r\n

PLAY rtsp://<*camera_address*>/video1 RTSP/1.0\r\n
CSeq: 4\r\n
Session: <*SessionId*>\r\n
Range: npt=0.000-\r\n
User-Agent: <*UA*>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <*ServerName*>\r\n
CSeq: 4\r\n
Session: <*SessionId*>\r\n
RTP-Info: url=trackID=1;seq=<*SequenceNumber*>;rtptime=0\r\n
\r\n

<*Video stream in RTP payload over UDP unicast*>

GET_PARAMETER rtsp://<*camera_address*>/video1 RTSP/1.0\r\n
CSeq: 5\r\n
Session: <*SessionId*>\r\n
User-Agent: <*UA*>\r\n
\r\n

<*Video stream in RTP payload over UDP unicast*>

TEARDOWN rtsp://<*camera_address*>/video1 RTSP/1.0\r\n
CSeq: 6\r\n
Session: <*SessionId*>\r\n
User-Agent: <*UA*>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <*ServerName*>\r\n
CSeq: 6\r\n
Session: <*SessionId*>\r\n

### 5.1.3. UDP Multicast Bit Stream (Video)

The following diagram and captured packets in this section show an example of getting video bit stream from the camera over UDP multicast in a situation where audio codec is disabled and a client requests a video stream.

In the example below in this section, the client uses "`destination`" parameter and "`client_port`" parameter in RTSP "`SETUP`" request to specify multicast address and ports for RTP session and RTCP session. One of the following ways can be used to specify the multicast address and/or the ports.

**a)** Specifying address and/or ports by using "`destination`" parameter and/or "`client_port`" parameter in RTSP "`SETUP`" request — This way is used in the example mentioned below.

**b)** Specifying address and/or ports by using "`camera.cgi`" CGI command with CGI parameters listed below.

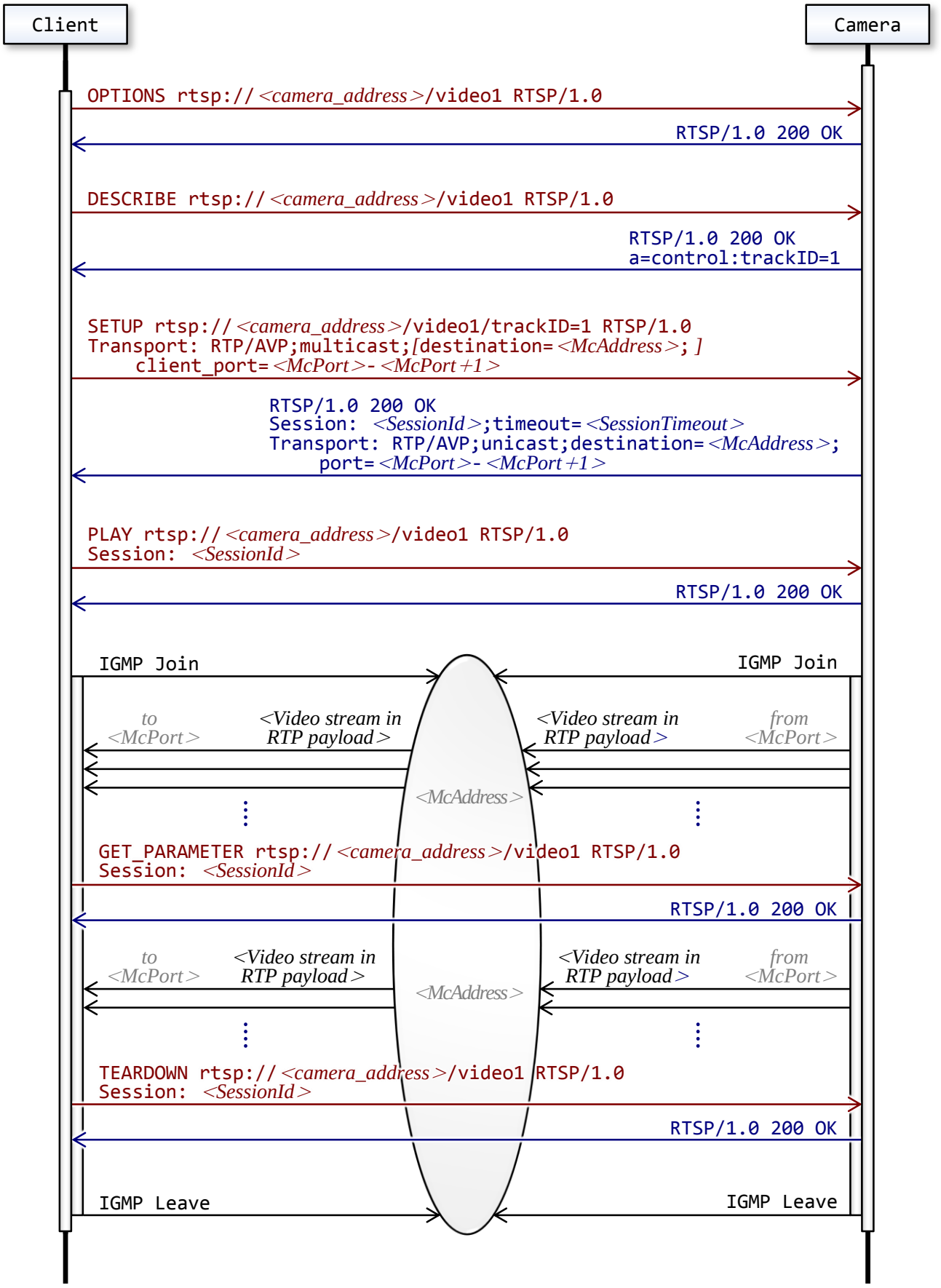
| CGI Parameter | Corresponding to |
|---|---|
| `RTSPMcAddress` | IPv4 multicast address |
| `RTSPMcVideoPort1` | A pair of multicast ports for the "`ImageCodec1`" live stream |
| `RTSPMcVideoPort2` | A pair of multicast ports for the "`ImageCodec2`" live stream |
| `RTSPMcVideoPort3` | A pair of multicast ports for the "`ImageCodec3`" live stream |
| `RTSPMcAudioPort` | A pair of multicast ports for the "`AudInCodec`" live stream |

For details about them, please refer to relevant document — "CGI Commands Technical Guide".

**c)** Not explicitly specifying multicast address and/or ports — In this case, multicast address and/or ports are automatically specified by the camera (RTSP server) and are indicated in RTSP response to "`SETUP`" request.

Regarding multicast "time to live" (TTL) value, the cameras indicate it to clients as "`ttl`" parameter in RTSP response to "`SETUP`" request.
Clients can set an arbitrary TTL value in RTSP "`SETUP`" request if needed. Or, when clients do not explicitly specify TTL value in "`SETUP`" request, the cameras automatically use "`McTtl`" CGI parameter as TTL value instead. `McTtl` can be changed by using "`camera.cgi`" CGI command if needed. Please refer to relevant document "CGI Commands Technical Guide" for details about `camera.cgi` and `McTtl`.

Regarding SSM (source-specific multicast), please refer to chapter 7 "Source-Specific Multicast".


Client
Camera
OPTIONS rtsp://<camera_address>/video1 RTSP/1.0
RTSP/1.0 200 OK
DESCRIBE rtsp://<camera_address>/video1 RTSP/1.0
RTSP/1.0 200 OK
a=control:trackID=1
SETUP rtsp://<camera_address>/video1/trackID=1 RTSP/1.0
Transport: RTP/AVP;multicast;[destination=<McAddress>; ]
client_port=<McPort>-<McPort+1>
RTSP/1.0 200 OK
Session: <SessionId>;timeout=<SessionTimeout>
Transport: RTP/AVP;unicast;destination=<McAddress>;
port=<McPort>-<McPort+1>
PLAY rtsp://<camera_address>/video1 RTSP/1.0
Session: <SessionId>
RTSP/1.0 200 OK
IGMP Join
IGMP Join
to <McPort>
<Video stream in RTP payload>
<Video stream in RTP payload>
from <McPort>
<McAddress>
GET_PARAMETER rtsp://<camera_address>/video1 RTSP/1.0
Session: <SessionId>
RTSP/1.0 200 OK
to <McPort>
<Video stream in RTP payload>
<Video stream in RTP payload>
from <McPort>
<McAddress>
TEARDOWN rtsp://<camera_address>/video1 RTSP/1.0
Session: <SessionId>
RTSP/1.0 200 OK
IGMP Leave
IGMP Leave

```
OPTIONS rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 1\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 1\r\n
Public: DESCRIBE, SETUP, TEARDOWN, PLAY, OPTIONS, SET_PARAMETER, GET_PARAMETER\r\n
\r\n

DESCRIBE rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 2\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 2\r\n
Content-Length: <Length>\r\n
Content-Type: application/sdp\r\n
Content-Base: rtsp://<camera_address>/video1/\r\n
\r\n

v=0\r\n
o=- <SessionIdForOrigin> 1 IN IP4 <camera_address>\r\n
s=<SessionName>\r\n
t=0 0\r\n
a=range:npt=now-\r\n
c=IN IP4 <ConnectionAddress>\r\n
m=<MediaNameAndTransportAddress>\r\n
a=rtpmap:<PayloadType> <EncodingName>/<ClockRate>\r\n
a=control:trackID=1\r\n
a=framerate:<FrameRate>\r\n
a=fmtp:<Format> <FormatSpecificParameters>\r\n
\r\n

SETUP rtsp://<camera_address>/video1/trackID=1 RTSP/1.0\r\n
CSeq: 3\r\n
Transport:
RTP/AVP;multicast;destination=<DestinationAddress>;client_port=<McPort>-<McPort
+1>\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 3\r\n
Session: <SessionId>[;timeout=<SessionTimeout>]\r\n
Cache-Control: must-revalidate\r\n
```

```
Transport:
RTP/AVP;multicast;destination=<DestinationAddress>;port=<McPort>-<McPort+1>;ttl
=<MulticastTimeToLive>\r\n
\r\n

PLAY rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 4\r\n
Session: <SessionId>\r\n
Range: npt=0.000-\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 4\r\n
Session: <SessionId>\r\n
RTP-Info: url=trackID=1;seq=<SequenceNumber>;rtptime=0\r\n
\r\n

<Video stream in RTP payload over UDP multicast>

GET_PARAMETER rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 5\r\n
Session: <SessionId>\r\n
User-Agent: <UA>\r\n
\r\n

<Video stream in RTP payload over UDP multicast>

TEARDOWN rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 6\r\n
Session: <SessionId>\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 6\r\n
Session: <SessionId>\r\n
```

**<Note>**

To get video and/or audio bit stream over UDP multicast, the camera requires you to enable multicast streaming function in advance. Otherwise, a request for multicast streaming fails during RTSP communication. For details about ways to enable multicast streaming function of the camera, please refer to relevant documents — “User’s Guide” and “CGI Commands Technical Guide”.

## 5.2. Getting Audio Stream

The following captured packets in this section show an example of getting an audio bit stream from the camera over TCP in a situation where audio codec is enabled and a client requests an audio stream.

Detailed explanations about RTSP audio streaming over UDP unicast and UDP multicast are left out — these can be known by analogy of the cases of video streaming described in section 5.1.2 “UDP Unicast Bit Stream (Video)” and 5.1.3 “UDP Multicast Bit Stream (Video)”.

```
OPTIONS rtsp://<camera_address>/audio RTSP/1.0\r\n
CSeq: 1\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 1\r\n
Public: DESCRIBE, SETUP, TEARDOWN, PLAY, OPTIONS, SET_PARAMETER, GET_PARAMETER\r\n
\r\n

DESCRIBE rtsp://<camera_address>/audio RTSP/1.0\r\n
CSeq: 2\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 2\r\n
Content-Length: <Length>\r\n
Content-Type: application/sdp\r\n
Content-Base: rtsp://<camera_address>/audio/\r\n
\r\n

v=<ProtocolVersion>\r\n
o=- <SessionIdForOrigin> 1 IN IP4 <camera_address>\r\n
s=<SessionName>\r\n
t=0 0\r\n
a=range:npt=now-\r\n
c=IN IP4 <ConnectionAddress>\r\n
m=<MediaNameAndTransportAddress>\r\n
a=rtpmap:<PayloadType> <EncodingName>/<ClockRate>[/<EncodingParameters>]\r\n
a=control:trackID=2\r\n
\r\n

SETUP rtsp://<camera_address>/audio/trackID=2 RTSP/1.0\r\n
CSeq: 3\r\n
```

```
Transport: RTP/AVP/TCP;unicast;interleaved=0-1\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 3\r\n
Session: <SessionId>[;timeout=<SessionTimeout>]\r\n
Cache-Control: must-revalidate\r\n
Transport: RTP/AVP/TCP;interleaved=0-1;ssrc=<SSRC>\r\n
\r\n

PLAY rtsp://<camera_address>/audio RTSP/1.0\r\n
CSeq: 4\r\n
Session: <SessionId>\r\n
Range: npt=0.000-\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 4\r\n
Session: <SessionId>\r\n
RTP-Info: url=trackID=2;seq=<SequenceNumber>;rtptime=0\r\n
\r\n

<Audio stream in RTP payload over TCP>

GET_PARAMETER rtsp://<camera_address>/audio RTSP/1.0\r\n
CSeq: 5\r\n
Session: <SessionId>\r\n
User-Agent: <UA>\r\n
\r\n

<Audio stream in RTP payload over TCP>

TEARDOWN rtsp://<camera_address>/audio RTSP/1.0\r\n
CSeq: 6\r\n
Session: <SessionId>\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 6\r\n
Session: <SessionId>\r\n
```

## 5.3. Getting Both Video and Audio Bit Streams

In a situation where audio codec is enabled and a client requests just a video stream, the camera transmits the video stream and also an audio stream at a time in an RTSP session. The following captured packets in this section show an example of getting both a video bit stream and an audio bit stream at a time from the camera.

As you can see in the example, session descriptions in RTSP response to "DESCRIBE" request contains two media descriptions — the first one is for video, the second one is for audio.

```
DESCRIBE rtsp://<camera_address>/video1 RTSP/1.0\r\n
CSeq: 2\r\n
User-Agent: <UA>\r\n
\r\n

RTSP/1.0 200 OK\r\n
Server: <ServerName>\r\n
CSeq: 2\r\n
Content-Length: <Length>\r\n
Content-Type: application/sdp\r\n
Content-Base: rtsp://<camera_address>/video1/\r\n
\r\n

v=0\r\n
o=- <SessionIdForOrigin> 1 IN IP4 <camera_address>\r\n
s=<SessionName>\r\n
t=0 0\r\n
a=range:npt=now-\r\n
c=IN IP4 <ConnectionAddress>\r\n

m=video 0 RTP/AVP 105\r\n
a=rtpmap:105 H264/90000\r\n
a=control:trackID=1\r\n
a=framerate:60.0\r\n
a=fmtp:105 packetization-mode=1; profile-level-id=2742e0; sprop-parameter-sets
=Z0LgH41oBQBbsBbIAAAfSAADqYNaAD0IAQBXvdQ8UIqA,KM4ESSAAAA==\r\n

m=audio 0 RTP/AVP 101\r\n
a=rtpmap:101 mpeg4-generic/16000/1\r\n
a=control:trackID=2\r\n
a=fmtp:101 profile-level-id=15; streamtype=5; mode=AAC-hbr; config=1408; SizeL
ength=13; IndexLength=3;IndexDeltaLength=3; constantDuration=1024; Profile=1;
bitrate=64000; \r\n
\r\n
```

## 5.4. rtpmap Attribute

Values of "`rtpmap`" attribute(s) in RTSP response to "`DESCRIBE`" request vary with codec of media stream(s). Here are examples with the cameras.

| Codec | rtpmap Attribute Value |
|---|---|
| H.264 | `a=rtpmap:105 H264/90000\r\n` |
| JPEG | `a=rtpmap:26 JPEG/90000\r\n` |
| G.711 | `a=rtpmap:0 PCMU/8000\r\n` |
| G.726 (40 kbps) | `a=rtpmap:97 G726-40/8000\r\n` |
| G.726 (32 kbps) | `a=rtpmap:98 G726-32/8000\r\n` |
| G.726 (24 kbps) | `a=rtpmap:99 G726-24/8000\r\n` |
| G.726 (16 kbps) | `a=rtpmap:100 G726-16/8000\r\n` |
| AAC (64 kbps) | `a=rtpmap:101 mpeg4-generic/16000/1\r\n` |
| AAC (128 kbps) | `a=rtpmap:102 mpeg4-generic/48000/1\r\n` |

# 6. Bit-Packing Formats of G.726 Audio

The cameras support the following two bit-packing formats for G.726 audio streaming.

| Format | Description |
|---|---|
| RFC 3551 compliant | G.726 audio bit stream in this bit-packing format from the camera is compatible with ONVIF conformant clients.<br>Non-ONVIF clients (such as generic media players) may not be able to correctly recognize G.726 audio bit stream in this format. |
| ITU-T compliant | G.726 audio bit stream in this bit-packing format from the camera is compatible with ITU-T recommendation.<br>It is expected that many of generic media players can recognize G.726 audio bit stream in this format. |

The cameras automatically choose one of the two formats listed above according to string in "`User-Agent:`" field in RTSP request.

When the user agent string in RTSP request matches with "`RTSPReUserAgentG726Itu`" CGI parameter in regular expression, the cameras transmit G.726 audio bit stream in ITU-T compliant bit-packing format. Otherwise, the cameras transmit RFC 3551 compliant bit-packing format. "`RTSPReUserAgentG726Itu`" CGI parameter is "`LibVLC.*`" by factory default. This CGI parameter can be changed by using "`camera.cgi`" CGI command. Please refer to relevant document "CGI Commands Technical Guide" for details about `camera.cgi` and `RTSPReUserAgentG726Itu`.

# 7. Source-Specific Multicast

To be supported.

# 8. RTSP/RTP Tunneling over HTTP

To be supported.

# Revision History

| Date | Revision | Description |
|---|---|---|
| 2012/10/26 | 1.0.0 | First edition. |

## Disclaimer

This document, in whole or in part, may not be reproduced or transferred for any purpose without prior written approval from Sony Corporation.

Sony Corporation reserves the right to make any modification to this document or the information contained herein at any time without notice.

Sony Corporation shall not bear any responsibility or liability for any damage, lost earning, and third party claim, resulting from the products and related documents.

## Copyright