



RTSP API Specification

Version	Date	Changes
1.0	2010/5/27	Created
1.1	2010/8/3	Modify for 3gpp, audio, event of RTSP command
1.2	2010/11/18	Modify for RTSP API
1.3	2011/3/29	Add RTSP URL Command



Table of Content

1. Introduction	3
2. References.....	3
2.1 RTSP protocol	3
2.2 SDP protocol.....	3
2.3 HTTP protocol	3
2.4 RTSP over HTTP	3
3. RTSP API	4
3.1 RTSP DESCRIBE	5
3.2 RTSP OPTIONS.....	8
3.3 RTSP SETUP	8
3.4 RTSP PLAY.....	9
3.5 RTSP PAUSE	11
3.6 RTSP TEARDOWN	11
3.7 RTSP over HTTP	12



1. Introduction

This document specifies the external RTSP-based application programming interface of the 3S cameras and video servers

2. References

2.1 RTSP protocol

[Real Time Streaming Protocol - RFC 2326](#)

2.2 SDP protocol

[Session Description Protocol - RFC 2327](#)

2.3 HTTP protocol

[Hypertext Transfer Protocol -- HTTP/1.0](#)

2.4 RTSP over HTTP

Using well-known concept to tunnel RTSP and RTP Over HTTP that was announced in QuickTime Project.
http://developer.apple.com/documentation/QuickTime/QTSS/Concepts/chapter_2_section_14.html

3. RTSP API

The RTSP URL is **rtsp://<server name>/<camera number>/<codec or special transport>** where <server name> is the host name or IP address of the server, <camera number> is cam1 to cam4 which depend on device supported channel number, and <codec or special transport> is mpeg4, mjpeg, h264, audio, or h264/multicast...etc which depend on device supported codec. All of RTSP URLs could fetch a SDP included video, audio and event tracks except audio RTSP URL. The DESCRIBE, SETUP, OPTIONS, PLAY, PAUSE and TEARDOWN methods are supported. The RTSP protocol is described in RFC 2326.

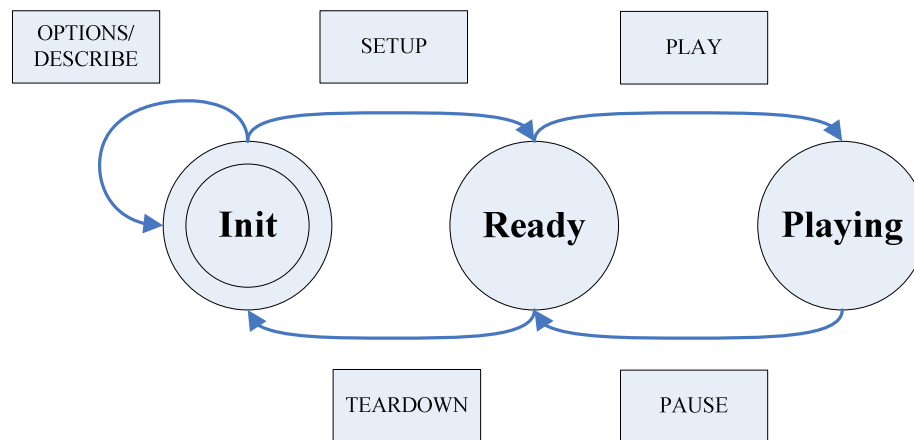


Fig. The finite state machine for RTSP protocol

The following table summarizes the purpose of methods

Method	Description
OPTIONS	Request the server options
DESCRIBE	Retrieves the description of media object from a server
SETUP	Specifies the transport mechanism to be used for the streamed media
PLAY	Tell the server to start sending data via the mechanism specified in SETUP
TEARDOWN	Request stops the stream delivery



PAUSE

The mechanism of RTSP pause then play in video streaming.

RTSP URL Command:

```
rtsp://192.168.20.100/cam1/mjpeg
rtsp://192.168.20.100/cam1/mpeg4
rtsp://192.168.20.100/cam1/h264      (main streaming)
rtsp://192.168.20.100/cam1/h264-1  (sub streaming)
rtsp://192.168.20.100/cam1/mjpeg/multicast
rtsp://192.168.20.100/cam1/mpeg4/multicast
rtsp://192.168.20.100/cam1/h264/multicast
rtsp://192.168.20.100/cam1/audio  (alone audio does not support multicast
transport)
rtsp://192.168.20.100/cam1/3gpp
rtsp://192.168.20.100/cam1/event      (N9071 support of FW v1.02 later)
rtsp://192.168.20.100/cam1/onvif-h264 (remove event track)
rtsp://192.168.20.100/cam1/onvif-h264-1 (remove event track)
rtsp://192.168.20.100/cam1/onvif-mjpeg (remove event track)
```

Note:

- 3gpp : Use sub video / audio stream and remove event track to provide low bandwidth streaming for handheld devices which compatible with 3GPP protocol.
- audio : Only provide audio steam when user do not want receive video and event track.
- event : Only provide event track for user who want to receive event information without video and audio data
- onvif-h264, onvif-h264-1, onvif-mjpeg : Use video / audio stream and remove event track.

*user can use Properties.RTSP.Paths to get supported RTSP paths

(EX: Properties.RTSP.Paths="/h264/h264-1/mjpeg/3gpp/audio/event/onvif-h264/onvif-h264-1/onvif-mjpeg")

Request syntax:

```
COMMAND URI RTSP/1.0<CRLF>
Headerfield1: val1<CRLF>
Headerfield2: val2<CRLF>
...
```



<CRLF>

Response syntax:

```
RTSP/1.0 ResultCode ResultString<CRLF>
Headerfield3: val3<CRLF>
Headerfield4: val4<CRLF>
...
<CRLF>
```

The following header fields are accepted by all commands. Other header fields are silently ignored (unless stated otherwise in the sections below).

Field	Description
Authorization	Authorization information from the client.
CSeq	Request sequence number.
Session	Session identifier (returned by server in SETUP response).
Content-Length	Length of content.

The following header fields can be generated for all responses by the RTSP server:

Field	Description
CSeq	Response sequence number (matches the sequence number of the request).
Session	Session identifier.
WWW-Authenticate	Authentication from client requested.

3.1 RTSP DESCRIBE

The DESCRIBE command returns the SDP (RFC 2327) description for the URI. The DESCRIBE command accepts the following additional header field:



Accept	List of content types that client supports (application/sdp is the only supported type).
--------	--

The DESCRIBE command generates the following additional header fields:

Content-Type	Type of content (application/sdp).
Content-Length	Length of SDP description.
Content-Base	If relative URLs are used in the SDP description, then this is the base URL.

Example:

```
DESCRIBE rtsp://192.168.2.123:554/cam1/h264 RTSP/1.0
CSeq: 9
Accept: application/sdp
Authorization: Digest username="root", realm="support@3svision.com.tw",
nonce="496ece8a08f5e9e25e5360aad2b2d085bdc56aae16036540437e0088f22b0c7a",
uri="rtsp://192.168.2.123:554/cam1/h264",
response="60aeb9a29fd2edd92f76d6d27fce28a7"
User-Agent: LIVE555 Streaming Media v2007.10.31
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 2
Content-Base: rtsp://192.168.2.123:554/cam1/h264/
Content-Type: application/sdp
Content-Length: 448

v=0
o=- 2208990726 2208990736 IN IP4 127.0.0.1
s=IP Camera Session
c=IN IP4 0.0.0.0
t=0 0
a=control:*
a=range:npt=0-
m=video 0 RTP/AVP 36
a=rtpmap:36 MP4V-ES/90000
a=fmtp:36 profile-level-
id=1;config=000001B001000001B58913000001000000012000C488BA98514043C1463F
a=control:trackID=1
m=audio 0 RTP/AVP 97
```



```
a=rtpmap:97 G726-24/8000
a=control:trackID=2
m=event 0 RTP/AVP 69
a=rtpmap:69 EVET/90000
a=fmtp:69 cps=30
a=control:trackID=3
```

3.2 RTSP OPTIONS

The OPTIONS command returns a list of supported RTSP commands.

Example:

```
OPTIONS * RTSP/1.0
CSeq: 2
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 2
Public: DESCRIBE, PAUSE, PLAY, SETUP, TEARDOWN
```

3.3 RTSP SETUP

The SETUP command configures the delivery method for the data. The SETUP command requires and generates the following additional header field:

Transport	Specifies how the data stream is transported. Supported variants: RTP/AVP;unicast;client_port=port1-port2 RTP/AVP;multicast;client_port=port1-port2 RTP/AVP/TCP;unicast
-----------	--

The response returns a session identifier that should be used with stream control commands to the server (PLAY, PAUSE, TEARDOWN). If the Session header includes a timeout parameter, then the session needs to be kept alive. This can be done by sending RTSP requests to the server containing the session identifier (e.g. OPTIONS) within the specified timeout time or through the use of RTCP. The RTSP server does not support reconfiguring of the transport parameters.

Example:





```
SETUP rtsp://192.168.2.123:554/cam1/h264/trackID=2 RTSP/1.0
CSeq: 4
Transport: RTP/AVP;unicast;client_port=2570-2571
Session: 4478FA4969E623D01ADA696A7E4C7E
Authorization: Digest username="root", realm="support@3svision.com.tw",
nonce="496ece8433b79950a3e314d3d934f75e4df8221e53d57a2d42e5890e1499b80c",
uri="rtsp://192.168.2.123:554/cam1/h264/",
response="ff882b850bd92389b3bab586d9614fa6"
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 4
Transport: RTP/AVP;unicast;client_port=2570-2571;server_port=53498-53499
Session: 4478FA4969E623D01ADA696A7E4C7E
```

3.4 RTSP PLAY

The PLAY command starts (or restarts if paused) the data delivery to the client. The PLAY command generates the following additional header fields:

Range	Specifies the range of time being played. Since only live streams are used, the specified time will always begin now and have no stop time.
RTP-Info	Information about the RTP stream. More specifically, it includes the next RTP sequence number that will be used.
RTCP-Interval	It specifies interval seconds of sending RTCP report from server.

Example:

```
PLAY rtsp://192.168.2.123:554/cam1/h264/ RTSP/1.0
CSeq: 6
Session: 4478FA4969E623D01ADA696A7E4C7E
Range: npt=0.000-
Authorization: Digest username="root", realm="support@3svision.com.tw",
nonce="496ece8433b79950a3e314d3d934f75e4df8221e53d57a2d42e5890e1499b80c",
uri="rtsp://192.168.2.123:554/cam1/h264/",
response="6c4b45b24c97f3d68de06f46f2b4f8dc"
```



Response example:

```
RTSP/1.0 200 OK
CSeq: 6
Session: 4478FA4969E623D01ADA696A7E4C7E;timeout=80
Range: npt=0-RTP-Info:
url=rtsp://192.168.2.123:554/cam1/h264/trackID=1;seq=46270;rtptime=264780
70;ssrc=2819682976,url=rtsp://192.168.2.123:554/cam1/h264/trackID=2;seq=1
2795;rtptime=2486850436;ssrc=1219700049,url=rtsp://192.168.2.123:554/cam1
/h264/trackID=3;seq=9701;rtptime=3921452014;ssrc=1146565264
RTCP-Interval: 5000
```



3.5 RTSP PAUSE

The PAUSE command pauses the data delivery from the server. We don't support this command now.

Example:

```
PAUSE /cam1/h264 RTSP/1.0
CSeq: 5
Session: 4478FA4969E623D01ADA696A7E4C7E
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 5
Session: 4478FA4969E623D01ADA696A7E4C7E
```

3.6 RTSP TEARDOWN

The TEARDOWN command terminates the data delivery from the server.

Example:

```
TEARDOWN rtsp://192.168.2.123:554/cam1/h264/ RTSP/1.0
CSeq: 7
Session: 4478FA4969E623D01ADA696A7E4C7E
Authorization: Digest username="root", realm="support@3svision.com.tw",
nonce="496ece8433b79950a3e314d3d934f75e4df8221e53d57a2d42e5890e1499b80c",
uri="rtsp://192.168.2.123:554/cam1/h264/",
response="faa81f0e96a6ee642a4508c3cf31912b"
```

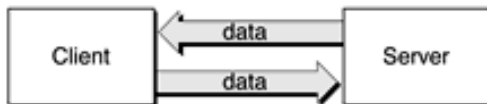
Response example:

```
RTSP/1.0 200 OK
CSeq: 7
Session: 4478FA4969E623D01ADA696A7E4C7E
```

3.7 RTSP over HTTP

It is possible to tunnel RTSP through HTTP to get through firewalls etc. This is achieved by setting up two sessions, one GET (for command replies and stream data) and one POST (for commands). RTSP commands sent on the POST connection are base64 encoded, but replies on the GET connection are plain text. To bind the two sessions together the server needs a unique ID (conveyed in the x-sessioncookie header). The GET and POST request are accepted on the HTTP port (default 80). The deliveries of bitstream are packed within RTSP connection.

Server-to-client connection created by the client's GET request



Client-to-server connection created by the client's POST request

Field	Description
x-sessioncookie	It is a globally unique identifier (GUID). The GUID makes it possible for the server to unambiguously bind the two connections by passing it as an opaque token to the C library strcmp function
Content-Type/Accept	In GET and POST requests, the application/x-rtsp-tunnelled MIME type for both the Content-Type and Accept directives must be specified; this MIME type reflects the data type that is expected and delivered by the client and server
Pragma	no-cache directive tells many HTTP 1.0 proxy servers not to cache the transaction.
Cache-Control	no-cache directive tells many HTTP 1.1 proxy servers not to cache the transaction
Expires	specifies an arbitrary time in the past. This directive is intended to prevent proxy servers from caching the transaction.

The RTSP/HTTP URL is: `http://<server name>/cam1/h264`. Supported methods are GET and POST.



GET example:

```
GET /cam1/h264 HTTP/1.0
User-Agent: MediaClientAxCtrl (LIVE555 Streaming Media v2007.10.31)
x-sessioncookie: b18101b2abe143e371791c6
Accept: application/x-rtsp-tunnelled
Pragma: no-cache
Cache-Control: no-cache
```

Response example:

```
HTTP/1.0 200 OK
Server: httpd
Date: Thu, 15 Jan 2009 06:44:49 GMT
Content-Type: application/x-rtsp-tunnelled
Connection: close
```

Note that the returned Content-Length is an arbitrary number and does not indicate the size of any data.

POST example:

```
POST /cam1/h264 HTTP/1.0
User-Agent: MediaClientAxCtrl (LIVE555 Streaming Media v2007.10.31)
x-sessioncookie: b18101b2abe143e371791c6
Content-Type: application/x-rtsp-tunnelled
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 32767
Expires: Sun, 9 Jan 1972 00:00:00 GMT
```

Note that the x-sessioncookie present in the POST content is the same as the x-sessioncookie in the corresponding GET request. Also note that the POST request must be sent on a separate connection to the server (i.e. it cannot be sent on the same connection as the GET request). No response is sent for the POST request.

RTSP requests made by the client on the POST connection must be encoded using the base64 method. The base64 encoding prevents HTTP proxy server from determining that an embodied RTSP request is a malformed HTTP requests.

Command request (base 64 encoded PLAY on the POST connection):





UExBWSAvbXB1ZzQvMS9tZWRpYS5hbXAglRTUC8xLjANCkNTZXE6IDQNC1N1
c3Npb246IDEzNjk5MTc3MTINCg0K

Which corresponds to this command:

```
PLAY /cam1/h264 RTSP/1.0  
CSeq: 4  
Session: 1369917712
```

Command response (plain-text on the GET connection, server does not need to do base64 encoded on the reply or bitstream):

```
RTSP/1.0 200 OK  
CSeq: 4  
Session: 1369917712
```