6000 Control Channel Signaling

Version 1.0d

2007/11/28

Revision History

version	Issue date	author	comment
1.0a	2005/06/28	Albus	
1.0b	2005/06/30	Joe Wu	Revised some explanations
1.0c	2006/03/29	Kate Hsu	Add missing headers,CSeq and Session
1.0d	2007/11/27	Kate Hsu	Remove unimplemented ALERT_MSG

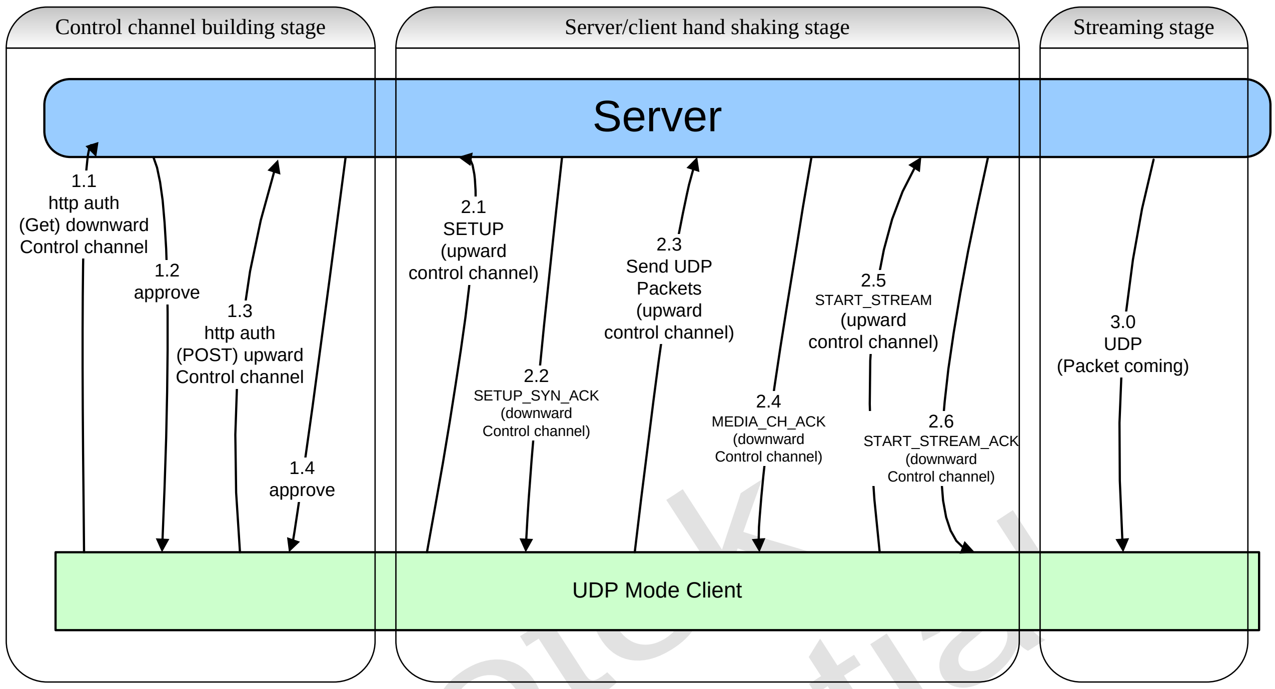
TABLE of CONTENTS

Overview

This document describes signaling sequence and format of the control messages of control channel of Vivotek 6000 series products. Control channel exchange the capabilities of codec between server and client as well as the transport information that tells how media data can be delivered via network. All the control channels are based on HTTP protocol for signaling. The control channel over HTTP is based on the idea of tunneling RTSP through HTTP to get through firewall. This is achieved by setting up two sessions, one GET (for downward control message) and one POST (for upward control message). The control messages 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 UDP streaming mode

The table next page describes the UDP mode signaling for Vivotek 6000 series products.


Control channel building stage
Server/client hand shaking stage
Streaming stage
Server
1.1 http auth (Get) downward Control channel
1.2 approve
1.3 http auth (POST) upward Control channel
1.4 approve
2.1 SETUP (upward control channel)
2.2 SETUP_SYN_ACK (downward Control channel)
2.3 Send UDP Packets (upward control channel)
2.4 MEDIA_CH_ACK (downward Control channel)
2.5 START_STREAM (upward control channel)
2.6 START_STREAM_ACK (downward Control channel)
3.0 UDP (Packet coming)
UDP Mode Client

UDP streaming mode

Message of 1.1

```
Get /cgi-bin/control.cgi HTTP/1.0
User-Agent: VVTK (ver=40)
x-sessioncookie: ABCDEFGHIJK…XY
Accept: application/x-vvtk-tunnelled
Pragma: no-cache
Cache-Control: no-cache
```

The message is a HTTP format message.

The ***x-sessioncookie*** should be exactly 22 characters. It is used for distinguish different clients. Users can have their own rule to generate this ID.

This connection is used to establish the downward control channel.

Message of 1.2

```
HTTP/1.1 200 OK
Content-Type: application/x-vvtk-tunnelled
Date: Thu, 19 Aug 1982 18:30:00 GMT
Cache-Control: no-store
Pragma: no-cache
Content-Language: en
SID: 0
```

```
Audio Mode: Full Duplex
Privilege : 143
Server: Network Camera
Connection: close
```

The message is a HTTP format message.

Audio Mode	Description
Full Duplex	Can listen and talk simultaneously
Half Duplex	Can listen and talk but not at the same time
Talk Only	Talk only
Listen Only	Listen only

The privilege is a 16-bits integer.

Bit position	Description
0	DI/DO privilege
1	Listen privilege
2	Talk privilege
3	Camera control privilege
7	Configuration privilege
Others	Reserved and should be 0

Message of 1.3

```
POST /cgi-bin/control.cgi HTTP/1.0
User-Agent: VVTK (ver=40)
x-sessioncookie: ABCDEFGHIJK…XY
Content-Type: application/x-vvtk-tunnelled
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 32767
Expires: Sun, 9 Jan 1972 00:00:00 GMT
```

The message is a HTTP format message.

The ***x-sessioncookie*** should be exactly 22 characters. It is used for distinguish different clients. Users can have their own rule to generate this ID.

This connection is used to establish the upward control channel. The x-sessioncookie should be the same as that of downward control channel.

Message of 1.4

OPEN VCM/1.0

Only method 1.1, 1.2 and 1.3 are completed, the server will send out this message.

Format of 2.1

Control channel message: SETUP

This is message is a Base64 encoding data. Please refer to Chapter 5 for more details.

Format of 2.2

Control channel message: SETUP_SYN_ACK

Please refer to Chapter 5 for more details.

Format of 2.3

This is an UDP packet which contains the sid number that server sent to client in method 1.2. Please send the sid number in binary and big endian format.

When the packet is sent to video port which is received from method 2.2, it means we're going to request video channel and data.

When the packet is sent to audio port which is received from method 2.2, it means we're going to request audio channel and data.

Format of 2.4

Control channel message: MEDIA_CH_ACK

Please refer to Chapter 5 for more details.

Format of 2.5

Control channel message: START_STREAM

This is message is a Base64 encoding data. Please refer to Chapter 5 for more details.

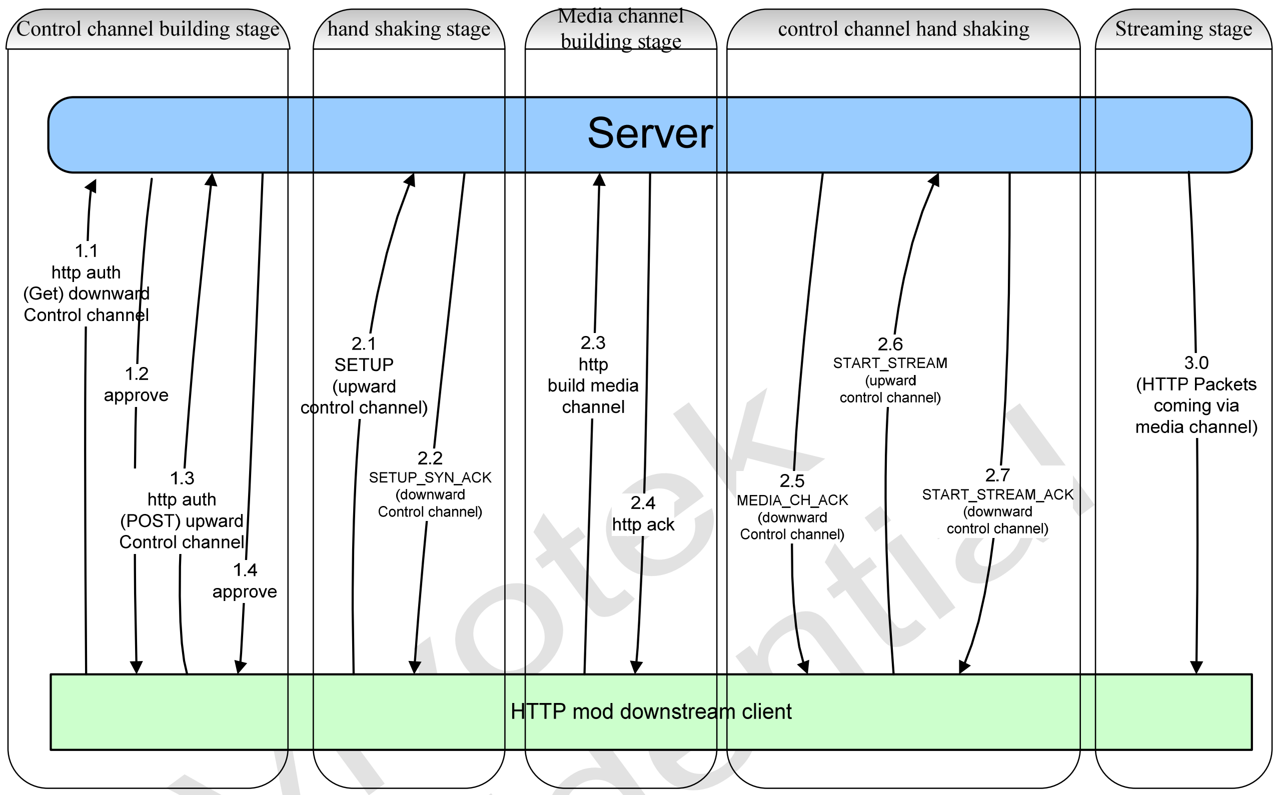
Format of 2.6

Control channel message: START_STREAM_ACK

Please refer to Chapter 5 for more details.

The HTTP down-streaming mode

The table next page describes the HTTP down-streaming mode signaling for Vivotek 6000 series products.


Control channel building stage
hand shaking stage
Media channel building stage
control channel hand shaking
Streaming stage
Server
1.1 http auth (Get) downward Control channel
1.2 approve
1.3 http auth (POST) upward Control channel
1.4 approve
2.1 SETUP (upward control channel)
2.2 SETUP_SYN_ACK (downward Control channel)
2.3 http build media channel
2.4 http ack
2.5 MEDIA_CH_ACK (downward Control channel)
2.6 START_STREAM (upward control channel)
2.7 START_STREAM_ACK (downward control channel)
3.0 (HTTP Packets coming via media channel)
HTTP mod downstream client

HTTP down streaming mode

Message of 1.1

```
Get /cgi-bin/control.cgi HTTP/1.0
User-Agent: VVTK (ver=40)
x-sessioncookie: ABCDEFGHIJK…XY
Accept: application/x-vvtk-tunnelled
Pragma: no-cache
Cache-Control: no-cache
```

The message is a HTTP format message.

The ***x-sessioncookie*** should be exactly 22 characters. It is used for distinguish different clients. Users can have their own rule to generate this ID.

Message of 1.2

```
HTTP/1.1 200 OK
Content-Type: application/x-vvtk-tunnelled
Date: Thu, 19 Aug 1982 18:30:00 GMT
Cache-Control: no-store
Pragma: no-cache
Content-Language: en
SID: 0
Audio Mode: Full Duplex
Privilege : 143
Server: Network Camera
Connection: close
```

The message is a HTTP format message.

Message of 1.3

```
POST /cgi-bin/control.cgi HTTP/1.0
User-Agent: VVTK (ver=40)
x-sessioncookie: ABCDEFGHIJK…XY
Content-Type: application/x-vvtk-tunnelled
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 32767
Expires: Sun, 9 Jan 1972 00:00:00 GMT
```

The message is a HTTP format message.

The ***x-sessioncookie*** should be exactly 22 characters. It is used for distinguish different clients. Users can have their own rule to generate this ID.

Message of 1.4

```
OPEN VCM/1.0
```

Only method 1.1, 1.2 and 1.3 are completed, the server will send out this message.

Format of 2.1

Control channel message: SETUP

This is message is a Base64 encoding data. Please refer to Chapter 5 for more details.

Format of 2.2

Control channel message: SETUP_SYN_ACK

Please refer to Chapter 5 for more details.

Format of 2.3

This is a HTTP 1.0 request for http://<ip address>/ cgi-bin/downstream.cgi, in GET method.

Format of 2.4

This is a standard HTTP response.

Format of 2.5

Control channel message: MEDIA_CH_ACK

Please refer to Chapter 5 for more details.

Format of 2.6

Control channel message: START_STREAM

This is message is a Base64 encoding data. Please refer to Chapter 5 for more details.

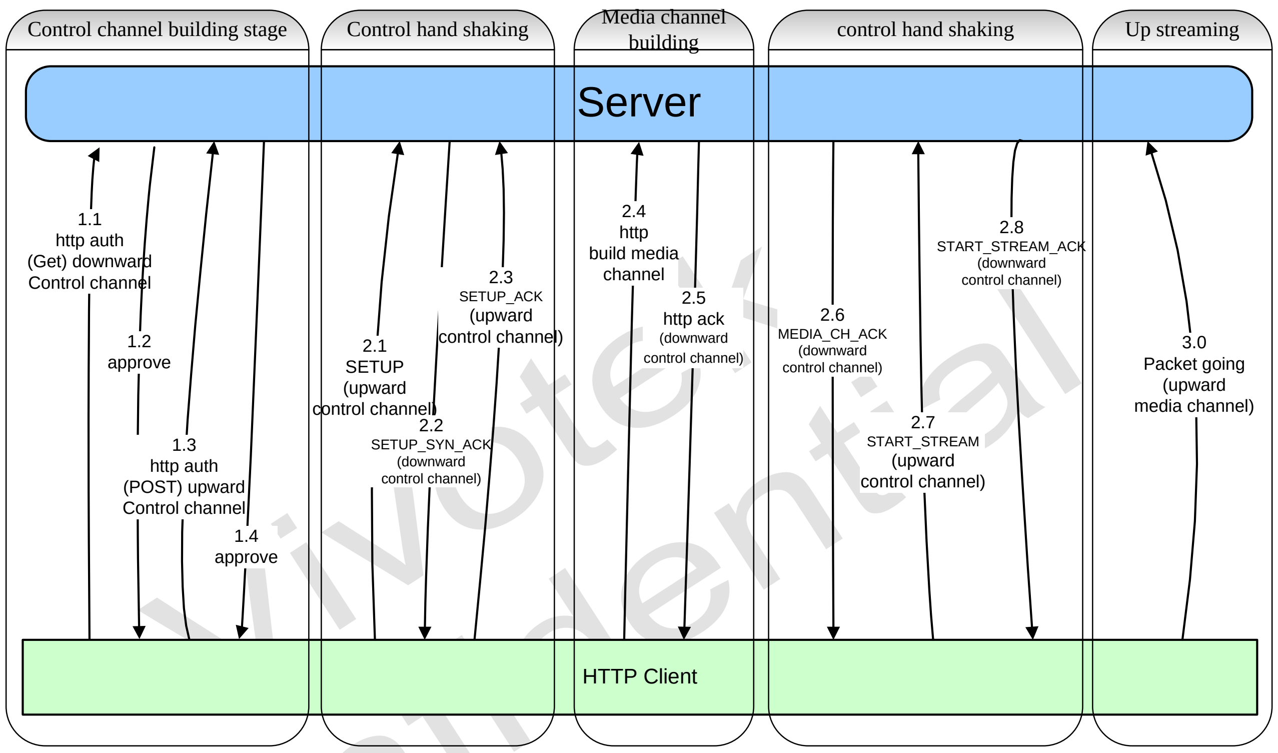
Format of 2.7

Control channel message: START_STREAM_ACK

Please refer to Chapter 5 for more details.

The HTTP up-streaming mode

The table next page describes the HTTP up-streaming mode signaling for Vivotek 6000 series products.


Control channel building stage
Control hand shaking
Media channel building
control hand shaking
Up streaming
Server
1.1 http auth (Get) downward Control channel
1.2 approve
1.3 http auth (POST) upward Control channel
1.4 approve
2.1 SETUP (upward control channel)
2.2 SETUP_SYN_ACK (downward control channel)
2.3 SETUP_ACK (upward control channel)
2.4 http build media channel
2.5 http ack (downward control channel)
2.6 MEDIA_CH_ACK (downward control channel)
2.7 START_STREAM (upward control channel)
2.8 START_STREAM_ACK (downward control channel)
3.0 Packet going (upward media channel)
HTTP Client

HTTP up streaming mode

Message of 1.1

```
Get /cgi-bin/control.cgi HTTP/1.0
User-Agent: VVTK (ver=40)
x-sessioncookie: ABCDEFGHIJK…XY
Accept: application/x-vvtk-tunnelled
Pragma: no-cache
Cache-Control: no-cache
```

The message is a HTTP format message.

The ***x-sessioncookie*** should be exactly 22 characters. It is used for distinguish different clients. Users can have their own rule to generate this ID.

Message of 1.2

```
HTTP/1.1 200 OK
```

```
Content-Type: application/x-vvtk-tunnelled
Date: Thu, 19 Aug 1982 18:30:00 GMT
Cache-Control: no-store
Pragma: no-cache
Content-Language: en
SID: 0
Audio Mode: Full Duplex
Privilege : 143
Server: Network Camera
Connection: close
```

The message is a HTTP format message.

Message of 1.3

```
POST /cgi-bin/control.cgi HTTP/1.0
User-Agent: VVTK (ver=40)
x-sessioncookie: ABCDEFGHIJK…XY
Content-Type: application/x-vvtk-tunnelled
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 32767
Expires: Sun, 9 Jan 1972 00:00:00 GMT
```

The message is a HTTP format message.

The ***x-sessioncookie*** should be exactly 22 characters. It is used for distinguish different clients. Users can have their own rule to generate this ID.

Message of 1.4

```
OPEN VCM/1.0
```

Only method 1.1, 1.2 and 1.3 are completed, the server will send out this message.

Format of 2.1

Control channel message: SETUP

This is message is a Base64 encoding data. Please refer to Chapter 5 for more details.

Format of 2.2

Control channel message: SETUP_SYN_ACK

Please refer to Chapter 5 for more details.

Format of 2.3

Control channel message: SETUP_ACK
This is message is a Base64 encoding data. Please refer to Chapter 5 for more details.

Format of 2.4

This is a HTTP 1.0 request for http://<ip address>/ cgi-bin/upstream.cgi?SID=<sid>, in POST method.
The <sid> is the number which received from method 1.2.

Format of 2.5

This is a standard HTTP response.

Format of 2.6

Control channel message: MEDIA_CH_ACK
Please refer to Chapter 5 for more details.

Format of 2.7

Control channel message: START_STREAM
Please refer to Chapter 5 for more details.

Format of 2.8

Control channel message: START_STREAM_ACK
Please refer to Chapter 5 for more details.

Vivotek Control Message 1.0 (VCM)

Vivotek Control Message is text-based, and it is always exchanged via control channels.

1. Basic Syntax

OCTET	=	<any 8-bit sequence of data>
CHAR	=	<any US-ASCII character (octets 0 - 127)>
UPALPHA	=	<any US-ASCII uppercase letter "A".."Z">
LOALPHA	=	<any US-ASCII lowercase letter "a".."z">
ALPHA	=	UPALPHA | LOALPHA
DIGIT	=	<any US-ASCII digit "0".."9">
CR	=	<US-ASCII CR, carriage return (13)>
LF	=	<US-ASCII LF, linefeed (10)>
SP	=	<US-ASCII SP, space (32)>
<">	=	<US-ASCII double-quote mark (34)>
CRLF	=	CR LF
HEX	=	"A" | "B" | "C" | "D" | "E" | "F"| "a" | "b" | "c" | "d" | "e" | "f" | DIGIT

2. Message Type

VCM messages consist of requests and responses from client and server.

```
    VCM-message    = Request | Response      ; VMC/1.0 messages
```

request and response messages use the generic message format as follows. Both types of message consist of a start-line, one or more header fields (also known as "headers") and an empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields.

```
    generic-message   =    start-line
                           *message-header
                           CRLF
    start-line        =    Request-Line | Response-Line
```

3. Message Header

Each header field consists of a name followed by a colon (":") and the field value. Field names are case-insensitive.

```
message-header    = field-name ":" [ field-value ] CRLF
field-name        = token
field-value       = *( field-content | SP)
field-content     = <the OCTETs making up the field-value
                    and consisting of either *TEXT or combinations
                    of token, tspecials, and quoted-string>
```

4. VCM version

VCM uses a "<major>.<minor>" numbering scheme to indicate versions of the message. The version of a Vivotek control message is indicated by an VCM-Version field in the first line of the message.

```
VCM -Version    = "VCM" "/" 1*DIGIT "." 1*DIGIT
    Example:
        VCM/1.0
```

5. Request Line

Request-Line = Method SP VCM-Version CRLF

Method	OPEN
	SETUP
	START_STREAM
	STOP_STREAM
	PACKET_LOSS
	FORCE_INTRA
	KEEP_ALIVE
	GET_LOCATION
	CLOSE

6. Response Line

Response-Line = Acknowledge SP VCM-Version [SP Status-Code] CRLF

Acknowledge	SETUP_SYN_ACK

	SETUP_ACK
	MEDIA_CH_ACK
	START_STREAM_ACK
	STOP_STREAM_ACK
	GET_LOCATION_ACK
	VER_NOT_SUPPORT

7. Header

7.1. ChDir

This field describes the media channel direction client want to build.

ChDir = "ChDir" ":" "upstream" | "downstream"

Example:

ChDir: downstream

7.2. Accept

This filed indicates the client or server can accept or will accept what kind of data. In this version (VCM/1.0) only media data is defined.

Accept = "Accept" ":" (media-type)

media-type= (type "/" subtype) *(";" parameter)

7.3. Submit

This field indicates the client or server will send what kind of data. In this version (VCM/1.0) only media data is defined.

Submit = "Submit" ":" (media-type)

media-type= (type "/" subtype) *(";" parameter)

7.4. ChNo

This field indicates the number of media channel to build.

ChNo = “ChNo” “:” “avboth” | “vonly” | “aonly” | “none”

Example:

ChNo: vonly

7.5. Transport

This field describes the transport protocol of media channel and the port of audio and video UDP packet.

Transport = “Transport” “:” (“UDP”|“HTTP”) | (“vport”=*DIGIT,“aport”=*DIGIT)

Example:

Transport: vport=5001, aport=5002

Transport: HTTP

7.6. Bandwidth

This field indicates the bandwidth of the client.

Bandwidth = "Bandwidth" ":" 1*DIGIT

Example:

Bandwidth: 4000

7.7. Location

This field indicates the location of the camera.

Location: ALPHA|DIGIT

Location = " Location " ":" ALPHA|DIGIT

Example:

Location: Door2

7.8 CSeq

This field indicates the sequence number of setup sequence message.
The sequence number of request and acknowledge message is the same.

CSeq = "CSeq" ":" *DIGIT

Example:

CSeq: 2

7.9 Session

This field indicates the session ID of setup sequence message.
The session ID is the same for all messages in a setup sequence.

Session = "Session" ":" *DIGIT

Example:

Session : 21567843

8. Media Type

This field describes the media type of the client or server can send or receive.

media-type = type "/" subtype *(";" parameter)

type = “audio” | “video”

audio subtype = “g7221” | “g729”

audio parameter = (“bitrate”=1*DIGIT) | (“freq”=1*DIGIT) | (“chno” = 1*DIGIT)

bitrate : the audio bitrate

freq: the sample frequency

chno: channel number

video subtype = “jpeg” | “h263” | “mp4v”

video has no parameter in this version.

Example:

audio/g7221;bitrate=24000,

audio/g729,

video/jpeg, video/h263, video/mp4v

9. Status Code

The Status-Code element is a 3-digit integer result code.

Status-Code	Description
200	OK
400	Syntax Error
401	Insufficient information in the request
402	Bad sequence of request
410	Not Authorized

411	Audio Mode Not Support
412	Audio upstream channel is occupied
420	UDP Not Support
421	No HTTP Socket Built
422	Establish UDP Failed
430	Audio Codec Not Support
431	Video Codec Not Support
440	Mismatch With Setup Information
500	Internal Server Error
501	Service Unavailable

10. Example

(1) Setup an audio and video downstream channel.

```
C->S:  SETUP VCM/1.0
       CSeq: 1
       ChDir: downstream
       Accept: audio/g7221, video/jpeg, video/h263
       ChNo: avboth
       Transport: UDP

S->C:  SETUP_SYN_ACK VCM/1.0 200
       Session: 4086
       CSeq: 1
       Submit: audio/g7221, video/h263
       Transport: aport=5112, vport=5113
```

If the client builds two UDP channels successfully,

```
S->C: MEDIA_CH_ACK VCM/1.0 200
      Session: 4086
      CSeq: 2
      ChDir: downstream

C->S: START_STREAM VCM/1.0
       Session: 4086
```

```
            CSeq: 3
            ChDir: downstream

   S->C: START_STREAM_ACK   VCM/1.0   200
            Session: 4086
            CSeq: 3
            ChDir: downstream
```

(2) Setup an audio upstream channel.

```
   C->S: SETUP VCM/1.0
            CSeq: 1
            ChDir: upstream
            Submit: audio/g7221, audio/g729

   S->C:   SETUP_SYN_ACK VCM/1.0 200
           Session: 7419
           CSeq: 1
           Accept: audio/g7221

   C->S: SETUP_ACK VCM/1.0 200
            CSeq: 1
            Session: 7419
            Submit: audio/g7221
```

(3) Get the camera location

```
   C->S:   GET_LOCATION VCM/1.0

   S->C:   GET_LOCATION_ACK VCM/1.0 200
            Location: BigWindow
```

List of Control Message Type

- All capabilities are exchanged and selection is done in one handshake, <= fast mode.

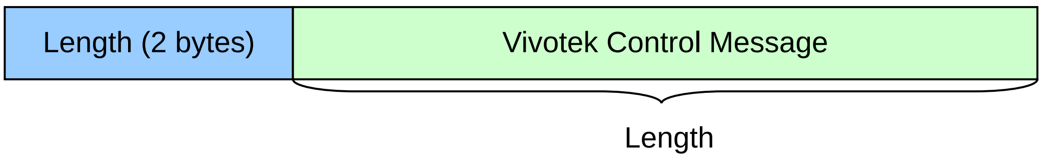
Message Type	Description	Direction
SETUP	■ Sequence number ■ Channel direction	C->S

	■ Upstream ■ Downstream ■ Version ■ Client audio and video decode capability and codec priority (*Downstream Only*) ■ Audio codec type, sampling frequency, bitrate and channel number ◆ G.722.1 ◆ G.729 ■ Video ◆ JPEG ◆ H.263 ◆ MP4V ■ Client A/V encode capability (*Upstream Only*) ■ Channel number (*Downstream Only*) ■ Both audio and video ■ Video only ■ Audio only ■ Transport protocol (*Downstream Only*) ■ UDP ■ HTTP ■ Bandwidth	
SETUP_SYN_ACK	■ Status code ■ Sequence number ■ Session ID ■ Server A/V encode type (*Downstream Only*) ■ Server A/V decode capability (*Upstream Only*) ■ Audio port and video port for UDP	S->C
SETUP_ACK (*Upstream Only*)	■ Status code ■ Sequence number ■ Session ID ■ Client A/V encode type	C->S
MEDIA_CH_ACK	Acknowledge establishing channel for streaming ■ Sequence number ■ Session ID ■ Channel direction ■ Upstream	S->C

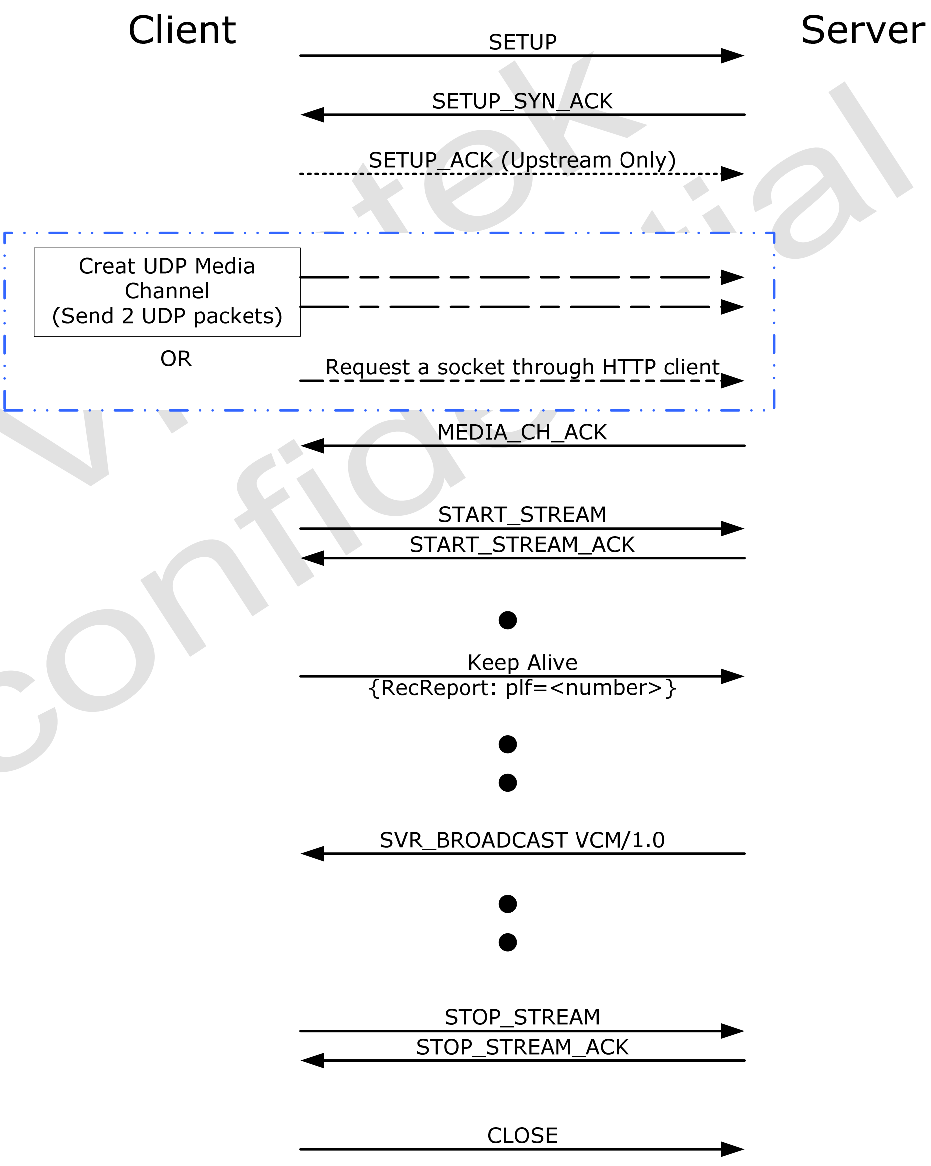
	■ Downstream ■ Status code	
START_STREAM	Start media streaming ■ Sequence number ■ Session ID ■ Channel direction 　■ Upstream 　■ Downstream	C->S
STOP_STREAM	Stop media streaming ■ Sequence number ■ Session ID ■ Channel direction 　■ Upstream 　■ Downstream	C->S
START_STREAM_ACK	Acknowledge to start media streaming from the client ■ Status code ■ Sequence number ■ Session ID ■ Channel direction 　■ Upstream 　■ Downstream	S->C
STOP_STREAM_ACK	Acknowledge to stop media streaming from the client ■ Status code ■ Sequence number ■ Session ID ■ Channel direction 　■ Upstream 　■ Downstream	S->C
PACKET_LOSS	Report the packet loss of the client for the flow control of server	C->S
FORCE_INTRA	Let the server force intra frame	C->S
KEEP_ALIVE	Keep alive	C->S
GET_LOCATION	Get the video text of server	C->S
GET_LOCATION_ACK	● Status code ● Location string	S->C
OPEN	Indicate the completion of control channel setup	S->C
CLOSE	Close control channel	C->S

VER_NOT_SUPPORT	The message version is not support	S->C

- Transmit message type



- Control channel communication:


Client
Server
SETUP
SETUP_SYN_ACK
SETUP_ACK (Upstream Only)
Creat UDP Media Channel (Send 2 UDP packets)
OR
Request a socket through HTTP client
MEDIA_CH_ACK
START_STREAM
START_STREAM_ACK
Keep Alive {RecReport: plf=<number>}
SVR_BROADCAST VCM/1.0
STOP_STREAM
STOP_STREAM_ACK
CLOSE

Notes:

1. Client should send KEEP_ALIVE message every 90 seconds, or the connection will be broken.
2. In KEEP_ALIVE message, if the downstream is in UDP mode, please append the following message to

show UDP packet loss rate:

```
RecReport: plf=<number>
```

3. The UDP packet loss fraction:

plf (integer)	Level of sending packets adjusted in server side
81~100	Decrease by 2
16~80	Decrease by 1
3~15	Keep current level
0~2	Increase by 1

When the codec of video is mpeg-4, there are 10 levels.

When the codec of video is JPEG, there are 7 levels.

That is if the packet loss rate is high, client may want to decrease the level of sending packets to avoid network traffic, then append “RecReport: plf=81” to the KEEP_ALIVE message, the level of sending packets will be decreased, therefore the server will send less packets out for the current connection.

4. When the audio mode selection in server side is changed, the server will send to every control channel with the following message:

```
SVR_BROADCAST VCM/1.0
```