ITX DVR RTSP API

RTSP API Development Guide

Date 2013-11-18

Version 2.0

Table of Contents

Copyright Notice

HTTP API LICENSE AGREEMENT
This is a legal agreement (the "License Agreement") between you (either individual or an entity) and ITX Security.

1. GRANT OF LICENSE
ITX Security hereby grants to you the right to use the INTERFACE and the INTERFACE DESCRIPTION for the sole and limited purpose of creating, manufacturing and developing a solution that integrates any unit or portion included in the product range of ITX Security Digital Video Recorder, ITX Security Network Video Recorder, ITX Security Network Camera, ITX Security video encoders and ITX Security video decoders (as defined by ITX Security at its discretion) and to market, sell and distribute any such solution.

2. COPYRIGHT
The INTERFACE and the INTERFACE DESCRIPTION are owned by ITX Security and are protected by copyright laws and international treaty provisions. Any use of the INTERFACE and/or the INTERFACE DESCRIPTION outside the limited purpose set forth in Section 1 above is strictly prohibited.

3. NO REVERSE ENGINEERING
You may not reverse engineer, decompile, or disassemble the INTERFACE except to the extent required to obtain interoperability with other independently created computer programs as permitted by mandatory law.

4. TERMINATION
This License is effective until terminated. Your rights under this License will terminate automatically without notice from ITX Security if you fail to comply with any term(s) of this License.
Upon the termination of this License, you shall cease all use and disposition of the INTERFACE and/or THE INTERFACE DESCRIPTION whether for the purpose set forth in Section 1 above or not.

5. DISCLAIMER
5.1. THE INTERFACE AND THE INTERFACE DESCRIPTION ARE DELIVERED FREE OF CHARGE AND "AS IS" WITHOUT WARRANTY OF ANY KIND. THE ENTIRE RISK AS TO THE USE, RESULTS AND PERFORMANCE OF THE INTERFACE AND THE INTERFACE DESCRIPTION IS ASSUMED BY THE USER/YOU. ITX SECURITY DISCLAIMS ALL WARRANTIES,
WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT AND PRODUCT LIABILITY, OR ANY WARRANTY ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE WITH RESPECT TO THE INTERFACE AND THE INTERFACE DESCRIPTION.

5.2. YOU ARE YOURSELF RESPONSIBLE FOR EXAMINING WHETHER THE INTERFACE AND THE INTERFACE DESCRIPTION ARE ENCUMBERED BY OR INFRINGES UPON A RIGHT HELD BY A THIRD PARTY. ITX SECURITY, WHO HAS NOT UNDERTAKEN ANY SUCH INVESTIGATIONS, HAS NO KNOWLEDGE OF NOR DOES ITX SECURITY ACCEPT ANY LIABILITY FOR ANY SUCH ENCUMBRANCES OR INFRINGEMENTS.

5.3. YOU UNDERTAKE NOT TO PURSUE ANY CLAIMS WHATSOEVER AGAINST ITX SECURITY RELATING TO OR EMANATING FROM THE INTERFACE AND THE INTERFACE DESCRIPTION.

5.4. ITX SECURITY SHALL NOT BE LIABLE FOR LOSS OF DATA, LOSS OF PRODUCTION, LOSS OF PROFIT, LOSS OF USE, LOSS OF CONTRACTS OR FOR ANY OTHER CONSEQUENTIAL, ECONOMIC OR INDIRECT LOSS WHATSOEVER IN RESPECT OF USE OR DISPOSITION OF THE INTERFACE AND THE

INTERFACE DESCRIPTION.

5.5 YOU SHALL INDEMNIFY AND HOLD ITX SECURITY HARMLESS FROM ANY CLAIMS WHATSOEVER FROM ANY THIRD PARTY AGAINST ITX SECURITY RELATING TO OR EMANATING FROM YOUR USE OF THE INTERFACE AND THE INTERFACE DESCRIPTION UNDER THIS LICENSE AGREEMENT. THE FOREGOING INDEMNIFICATION INCLUDES BUT IS NOT LIMITED TO ANY AND ALL DAMAGES, COSTS AND EXPENSES (INCLUDING REASONABLE ATTORNEYS' FEES).

1. Overview

1.1 Description

2. This document defines RTSP API(Application Programming Interface).
3. RTSP(Real Time Streaming Protocol) is a protocol which controls media stream sent from the media server.
4. RTSP provides play and pause which are "remote control" commands.
5. It can specify the detailed play state on live or playback by using parameter.

1.2 History

Version	Date	Comment
2.0	2013-11-18	Resolution added for S1 VA CAM in the 4. VIDEO FRAME HEADER FORMAT.
1.9	2013-08-30	4. Frame Type Field updated
1.8	2012-01-04	Section 6. Security Tab added
1.7	2012-09-14	FAQ added
1.6	2012-07-25	Sending audio data added Reserved field information in the Video frame header added
1.4	2011-06-06	I Frame only and Stream Select added Video frame header resolution added
1.3	2010-09-03	User authorization added In 3.4.1 Pb_is_start_end parameter added
1.2	2010-04-15	video frame header format added
1.1	2010-04-01	English version added
1.0	2010-02-01	V1.0 Release Timezone Idx added Reviewer: DongUk Park
0.1a	2009-06-05	Initial version

2. RTSP Command Format Definition

This chapter defines basic format of commands to control the media stream.

2.1 REQUEST MESSAGE

[Description]

This section defines basic format of request messages.

[Syntax]

```
COMMAND rtsp://<servername>/PLAY_STATUS
[?<parameter>=<value>[&<parameter>=<value>...]] RTSP/1.0<CRLF>
Headerfield1: val1<CRLF>
Headerfield2: val2<CRLF>
...
<CRLF>
[Body]
```

*

[Example]

```
OPTIONS rtsp://192.168.100.134/live RTSP/1.0₩r₩n
CSeq: 1₩r₩n
User-Agent: ITX Security ₩r₩n
```

- COMMAND can be DESCRIBE, SETUP, OPTIONS, PLAY, PAUSE, TEARDOWN, GET_PARAMETER.
- servername means host name or ip address of server.
- Parameter starts with '?'.
- Each parameter is separated by '&'.
- PLAY_STATUS can be live or playback and followed by 'track number' when COMMAND is SETUP.
  ex) SETUP rtsp://192.168.100.134/live/track0 RTSP/1.0₩r₩n
- The numbers in the parameter value can be represented by hex and decimal.
- Chapter 3 describes supported parameters and values.
- If the parameter value is the string, it follows the RFC 1738.
- Header field used for specific command refers to each command.
- RTSP server provides the user authorization when the client connects to the server.
- RTSP user authorization process follows the RTSP standard.
- Below are the header fields which can be used for every command.

[Header field]

Header Field	Description

Authorization	Authorization information of client.
CSeq	Request sequence number.
Session	Session identifier (This field is contained when receiving SETUP response from server.)
Content-Length	content length.
Content-Type	content media type.
User-Agent	Information about the client that initiates the request.

2.2 RESPONSE MESSAGE

[Description]

This section defines basic format of response messages.

[Syntax]

```
RTSP/1.0 <Status Code> <Reason Phrase> <CRLF>
Headerfield3: val3<CRLF>
Headerfield4: val4<CRLF>
...
<CRLF>
[Body]
```

- Status Code and Reason Phrase follows the RFC 2326.
- The Header field used for specific command refers to the command.
- The Header fields which can be used for every command are listed below.

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

[Example]

```
RTSP/1.0 200 OK₩r₩n
CSeq: 1₩r₩n
Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, GET_PARAMETER₩r₩n
```

3. RTSP COMMANDS

This chapter defines ITX RTSP API supported commands.

3.1 RTSP OPTION

[Description]

It responds to RTSP Command List which is supported by ITX RTSP API.

3.1.1 REQUEST Message

[Example]

```
OPTIONS rtsp://192.168.100.134/live RTSP/1.0₩r₩n
CSeq: 1₩r₩n
```

3.1.2 RESPONSE Message

[Example]

```
RTSP/1.0 200 OK₩r₩n
CSeq: 1₩r₩n
Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, GET_PARAMETER₩r₩n
```

- OPTION command header field contains public.

Header field	Description
Public	Specify the supported RTSP commands.

3.2 RTSP DESCRIBE

[Description]

- DESCRIBE command is used when requesting the SDP of media stream.
- Session Description Protocol(SDP) follows RFC 2327.
- The server can determine the permission of the client's access to the server by checking the user's id and password in the authentication field.
- If the id or password is incorrect, server returns Code 401(Unauthorized) to the client.

3.2.1 REQUEST Message

[Example]

```
DESCRIBE rtsp://192.168.100.134/live RTSP/1.0₩r₩n
CSeq: 2₩r₩n
Accept: application/sdp₩r₩n
User-Agent: ITX Security ₩r₩n
```

- The header field contains the accept field when requesting DESCRIBE.

Header Field	Description
Accept	client supported content type list application/sdp are supported.

3.2.2 RESPONSE Message

[Example]

```
RTSP/1.0 200 OK₩r₩n
CSeq: 2₩r₩n
Content-Base: rtsp://192.168.100.134/live/₩r₩n
Content-Type: application/sdp₩r₩n
Content-Length: 499₩r₩n
₩r₩n
v=0₩r₩n
o=- 1 1 IN IP4 127.0.0.1₩r₩n
s=Test₩r₩n
a=type:broadcast₩r₩n
t=0 0₩r₩n
c=IN IP4 0.0.0.0₩r₩n
m=video 0 RTP/AVP 96₩r₩n
a=rtpmap:96 MP4V-ES/90000₩r₩n
a=fmtp:96 profile-level-id=3;config=
000001B003
000001B509
00000100
00000120008440FA28A021E0A31F
000001B243616D54696D3A20323030392D30312D31362046726920313233A30313A31330D0A46
726D526174653A2031360D0A54696D5374616D703A203030303333337333838300D0A43616D50
6F733A20313131303266336665653030303030530D0A416C6D4576656E743A20303030303030303
03030303030303030300D0A0000..
a=control:track0₩r₩n
```

- DESCRIBE response message contains header fields listed below.

Header Field	Description

Content-Type	Type of content (application/sdp).
Content-Length	Length of SDP description.
Content-Base	base URL.

3.3 RTSP SETUP

[Description]

- It determines the method of sending data.
- It gives the Session ID.
- By checking the user id and password, the server prevents unregistered user from accessing to the server.
- If id or password is incorrect, server returns code 401(Unauthorized) to the client.
- In case of requesting playback or archiving, server checks user authority and prevents user without authority from playback and archiving.

3.3.1 REQUEST Message

[Example]

```
SETUP rtsp://192.168.100.134/live/track0 RTSP/1.0₩r₩n
CSeq: 3₩r₩n
Transport: RTP/AVP;unicast;client_port=1722-1723₩r₩n
User-Agent: ITX Security ₩r₩n
```

- The track number obtained from DESCRIBE is contained in the request message.
- The transport field is contained in the header field when requesting SETUP.

Header field	Description
Transport	It determines the way how client receives data stream. below are supported the transmission format RTP/AVP;unicast;client_port=port1-port2 RTP/AVP;multicast;client_port=port1-port2 RTP/AVP/TCP;unicast

3.3.2 RESPONSE Message

[Example]

```
RTSP/1.0 200 OK₩r₩n
CSeq: 3₩r₩n
Transport: RTP/AVP;unicast;client_port=1722-1723;server_port=58106-58107₩r₩n
Session: 7A644E944BC1ED63B8EE4B9107D2C2₩r₩n
```

- Server_port is sent to client.

3.4 RTSP PLAY

[Description]

- This command is used when requesting PLAY.
- Parameters sending from client to server are different according to LIVE and PLAYBACK.
- When changing the parameter set configuration in the play state, sending parameter is achieved using PLAY command without shutting down the session.
- If parameter is changed during the playback, Play MUST be sent after sending PAUSE command.

3.4.1 REQUEST Message

[LIVE Example]

```
PLAY
rtsp://192.168.100.134/live?Live_video_channel_mask=0x0000FFFF&Live_audio_channel_mask=0
x00000000&Iframe_only=0&Stream_index=2/ RTSP/1.0₩r₩n
CSeq: 4₩r₩n
Session: 7A644E944BC1ED63B8EE4B9107D2C2₩r₩n
Range: npt=0.000-₩r₩n
User-Agent: ITX Security ₩r₩n
```

- Live parameters are used for purpose listed below.

Parameter	Value	Description
Live_video_channel_mask	0x00000000 ~0xFFFFFFFF	- it represents the client receiving video channel by bit(LSB first). ex)if client is supposed to receive 1~6 channels. Live_video_channel_mask=0x0000003f
Live_audio_channel_mask	0x00000000 ~0xFFFFFFFF	- it represents client receiving audio channel by bit(LSB first). - if set this parameter '0', it is auto mute.
Iframe_only	0,1	- 0 : I Frame and P Frame send. - 1 : Only I Frame send
Stream_index	0~3	- 0: Auto - 1: Main Stream - 2: Second Stream

[PLAYBACK Example]

```
PLAY rtsp://192.168.100.236/play
back?Pb_video_channel_mask=0x0000FFFF&Pb_audio_channel_mask=0x00000000&Pb_start_
time=1241474509&Pb_end_time=1241479509&Pb_direction=Forward&Pb_speed=1&Iframe_on
ly=0&Stream_index=2₩r₩n
CSeq: 4₩r₩n
Session: 7A644E944BC1ED63B8EE4B9107D2C2₩r₩n
Range: npt=0.000-₩r₩n
User-Agent: ITX Security ₩r₩n
```

- Playback parameters are used for purpose listed below.

Parameter	Value	Description
Pb_video_channel_mask	0x00000000 ~0xFFFFFFFF	- it represents the client receiving video channel by bit(LSB first). ex)if client is supposed to receive 1~7 channels. Pb_video_channel_mask =0X0000007F
Pb_audio_channel_mask	0x00000000 ~0xFFFFFFFF	- it represents client receiving audio channel by bit(LSB first). - if set this parameter '0', it is auto mute.
Pb_start_time	UTC Time Value	- Playback start time. - it represents in UTC. - if send value '0', playback starts from the beginning of the data stream.
Pb_end_time	UTC Time Value	- Playback end time. - it represents in UTC. - if send value '0', playback continues until the end of the data stream.
Pb_direction	Forward, Backward	- Playback direction.
Pb_speed	1~64	- Playback Speed
Pb_is_start_end	0~3	- 0 : indicates middle channel in case of multiple channel archiving. - 1 : indicates first channel in case of multiple channel archiving.

		- 2 : indicates last channel in case of multiple channel archiving. - 3 : indicates single channel archiving.
Iframe_only	0,1	- 0 : I Frame and P Frame send. - 1 : Only I Frame send
Stream_index	0~3	- 0: Auto - 1: Main Stream - 2: Second Stream

3.4.2 RESPONSE Message

[Example]

```
RTSP/1.0 200 OK₩r₩n
CSeq: 4₩r₩n
Session: 7A644E944BC1ED63B8EE4B9107D2C2
```

3.5 RTSP TEARDOWN

[Description]

- This command is used to terminate the data delivery from the server.

3.5.1 REQUEST Message

[Example]

```
TEARDOWN rtsp://192.168.100.134/live/ RTSP/1.0₩r₩n
CSeq: 5₩r₩n
Session: 7A644E944BC1ED63B8EE4B9107D2C2₩r₩n
User-Agent: ITX Security ₩r₩n
```

3.5.2 RESPONSE Message

[Example]

```
RTSP/1.0 200 OK₩r₩n
CSeq: 6₩r₩n
Session: 7A644E944BC1ED63B8EE4B9107D2C2₩r₩n
```

3.6 RTSP PAUSE

[Description]

- This command is used to change from a play state to a stop state.

- This command is usable only in playback state.
- Next action MUST be taken after the server receives response to PAUSE.
  ex) When changing parameter during the playback, the client MUST send PAUSE command to the server and then receives its response and retries play.

3.6.1 REQUEST Message

[Example]

```
PAUSE rtsp://192.168.100.134/live RTSP/1.0₩r₩n
CSeq: 5₩r₩n
Session: 7A644E944BC1ED63B8EE4B9107D2C2₩r₩n
User-Agent: ITX Security ₩r₩n
```

3.6.2 RESPONSE Message

[Example]

```
RTSP/1.0 200 OK₩r₩n
CSeq: 6₩r₩n
Session: 7A644E944BC1ED63B8EE4B9107D2C2₩r₩n
```

3.7 RTSP GET_PARAMETER

[Description]

- This command is used to receive DST and timezone information from the server.
- The number of parameters can be increased according to necessity.

3.7.1 REQUEST Message

[Example]

```
PAUSE rtsp://192.168.100.134/live RTSP/1.0₩r₩n
CSeq: 5₩r₩n
Session: 7A644E944BC1ED63B8EE4B9107D2C2₩r₩n
dst_onoff:₩r₩n
time_zone:₩r₩n
```

3.7.2 RESPONSE Message

[Example]

```
RTSP/1.0 200 OK₩r₩n
CSeq: 6₩r₩n
Session: 7A644E944BC1ED63B8EE4B9107D2C2₩r₩n
dst_onoff:1₩r₩n
```

```
time_zone:2₩r₩n
```

[Parameters]

Parameter	Value	Description
dst_onoff	0, 1	0 : DST OFF 1 : DST ON
time_zone	0x00~0x21	refer to [Time Zone Definition] table.

[Time Zone Definition]

code(value)	time zone	time
0x00	GMT-12	GMT-12
0x01	Pacific/Midway	GMT-11.00
0x02	Pacific/Hawaii	GMT-10.00
0x03	America/Anchorage	GMT-09.00
0x04	America/LA	GMT-08.00
0x05	America/Phoenix	GMT-07.00
0x06	America/CST	GMT-06.00
0x07	America/EST	GMT-05.00
0x08	America/Halifax	GMT-04.00
0x09	America/St.Johns	GMT-03.30
0x0a	America/Sao.Paulo	GMT-03.00
0x0b	Mid_Atlantic	GMT-02.00
0x0c	Atlantic/Azores	GMT-01.00
0x0d	Europe/London	GMT+00.00
0x0e	Europe/Berlin	GMT+01.00
0x0f	Europe/Istanbul	GMT+02.00
0x10	Africa/Cairo	GMT+02.00
0x11	Europe/Moscow	GMT+03.00
0x12	Asia/Tehran	GMT+03.30
0x13	Asia/Muscat	GMT+04.00
0x14	Asia/Kabul	GMT+04.30
0x15	Asia/Karachi	GMT+05.00
0x16	Asia/Calcutta	GMT+05.30
0x17	Asia/Katmandu	GMT+05.45
0x18	Asia/Dhaka	GMT+06.00

0x19	Asia/Rangoon	GMT+06.30
0x1a	Asia/Bangkok	GMT+07.00
0x1b	Asia/Beijing	GMT+08.00
0x1c	Asia/Tokyo	GMT+09.00
0x1d	Asia/Seoul	GMT+09.00
0x1e	Australia/Darwin	GMT+09.30
0x1f	Australia/Adelaide	GMT+09.30
0x20	Australia/Brisbane	GMT+10.00
0x21	Pacific/Noumea	GMT+11.00
0x22	Newzealand	GMT+12.00
0x23	Australia/Perth	GMT+08.00
0x24	Middle East/Jordan	GMT+02.00
0x25	Middle East/Lebanon	GMT+02.00
0x26	Middle East/Syria	GMT+02.00
0x27	Middle East/SaudiArabia	GMT+03.00
0x28	Middle East/Iraq	GMT+03.00
0x29	Middle East/Iran	GMT+03.30
0x2a	Middle East/UAE	GMT+04.00
0x2b	Australia/Sydney	GMT+10.00

- The standard of TIME in table Time Zone Definition is GMT TIME.
- TIME in table Time Zone Definition can be changed.

4. VIDEO FRAME HEADER FORMAT

This chapter defines ITX Video (Audio) frame header format.

- The video frame header is contained in RTP payload followed by frame data.
- The video frame is sent in below format.

[Sending packet format]

RTSPHEADER	RTPHEADER	ICODEC_HEADER	FRAMEDATA

- ICODEC_HEADER is ITX Video frame header.
- If sending packet is split into more than one packet, the rest of the packets except first are sent in below format.

[Middle packet format]

RTSPHEADER	RTPHEADER	FRAMEDATA

- ICODEC_HEADER can be described in C/C++ code listed below.

```
typedef struct _ICODEC_HEADER_T {
        unsigned char    chan;
        unsigned char    codec;
        unsigned char    flags       : 2;
        unsigned char    rec_reason : 6;
        unsigned char    version;
        unsigned int     frame_size;
        unsigned char    frame_type;
        unsigned char    timestamp;
        unsigned char    resolution;
        unsigned char    frame_rate;
        unsigned int     timestamp;
        unsigned int     reserved[2];
} ICODEC_HEADER;
```

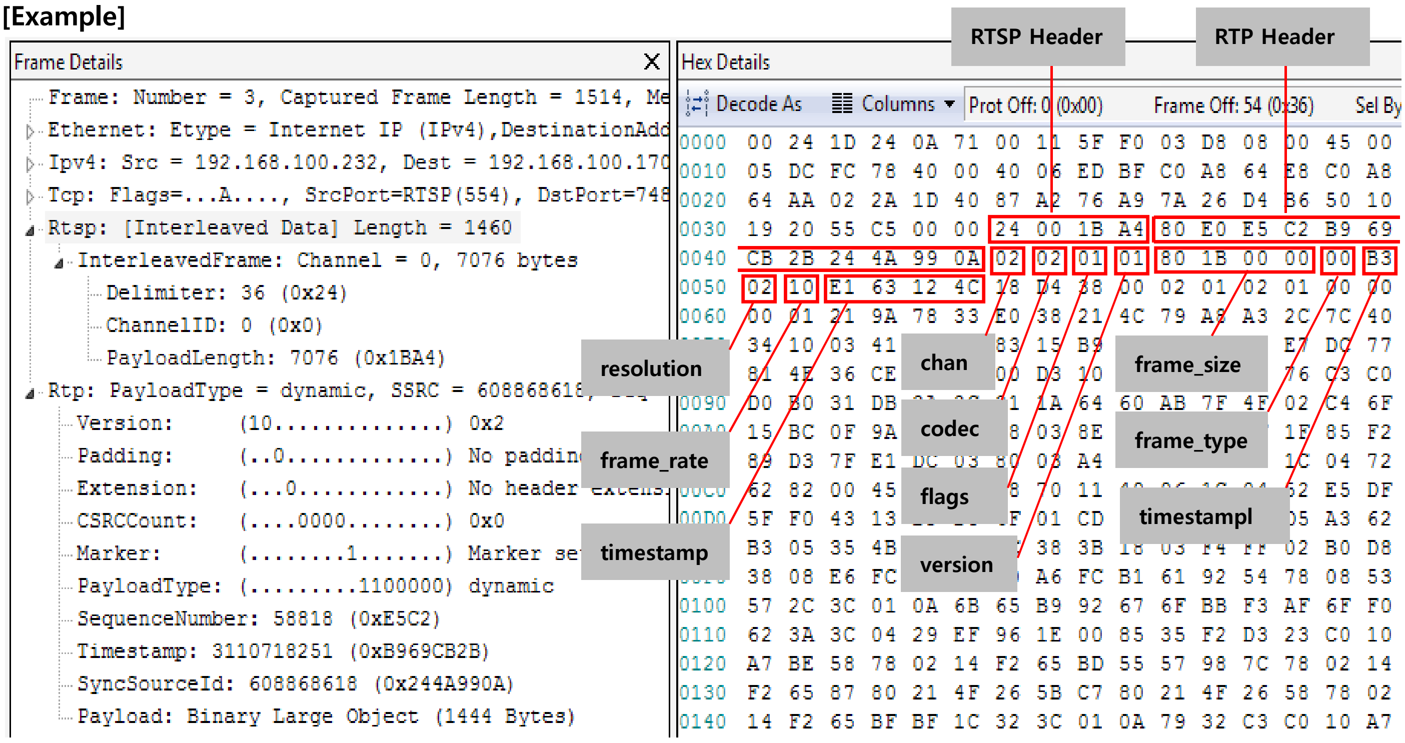
[Header field]

Header Field	Length	Description
chan	1 byte	channel number, it ranges from 0
codec	1 byte	codec type 1 : H.264 Base Profile, P-ref 2 : H.264 Base Profile, I-ref(key frame mode) 3 : JPEG 4 : MPEG4 I-ref 5 : MPEG4 P-ref 6 : H.264 Main Profile, P-ref 7 : H.264 Main Profile, I-ref if frame_type is audio 0 : uraw 1 : araw
flags	2 bits	0x01 : do not show 0x02 : start of recblock
rec_reason	6 bits	record reason 0x00 : none 0x01 : by timer (scheduler) 0x02 : by alarm 0x03 : by motion 0x04 : by user event 0x05 : by manual (panic) 0x06 : pre recorded
version	1 bytes	the header version
frame_size	4 bytes	frame size excluding header size
frame_type	1 byte	0 : P frame 1 : I frame 2 : Null frame 3 : Start frame 4 : End frame (end of clip) 5 : Reverse I frame 8 : end data(end data of entire frame) 9 : start data(first start data of entire frame) 10 : Audio frame 14 : overlapped data (If system tries to overwrite the current playback frame)
timestampl	1 bytes	timestamp (millisecond part, 5 milliseconds unit, GMT)

resolution	1 bytes	0x00 : NTSC_NONE 0x01 : NTSC_CIF 0x02 : NTSC_2CIF 0x04 : NTSC_4CIF (704x480) 0x84 : NTSC_4CIFP (Progressive) 0x11 : PAL_CIF 0x12 : PAL_2CIF 0x14 : PAL_4CIF (704x576) 0x94 : PAL_4CIFP (Progressive) 0x85 : 640x480 0x86 : 720x480 0x87 : 720x576 0x88 : 800x600 0x89 : 1024x768 0x8A : 1280x1024 0x8B : 1600x1200 0x8C : 1280x720 0x8D : 1920x1080 0x8E : 640x352 0x8F : 640x360 0x90 : 640x360I (Interlaced) 0x92 : 1280x720I (Interlaced) 0x93 : 1920x1080I (Interlaced) 0xA0 : 640x400 0xA1 : 800x450 0xA2 : 1440x900 0xA3 : 320x180 0x41 : 960H_NTSC_CIF 0x42 : 960H_NTSC_2CIF 0x44 : 960H_NTSC_4CIF 0xC4 : 960H_NTSC_4CIFP (Progressive) 0x51 : 960H_PAL_CIF 0x52 : 960H_PAL_2CIF 0x54 : 960H_PAL_4CIF 0xD4 : 960H_PAL_4CIFP (Progressive)
frame_rate	1 bytes	encoding rate – 1,2,4,8,16,32 fps
timestamp	4 bytes	timestamp (second part, gmt)
reserved[1]	4 bytes	1. This field saves the index information to verify

		whether the P-frame is included or not. 2. Adapted Models : OTM(Product Code:16XXX), SNF(Product Code:18XXX), STM(Product Code:19XXX) - Note: Product Code contains three digits in place of XXX. 3. Usage - Recording on the local set, frame check when in playback state. - Frame check in the Live view and playback state on the WebRA 4. Composition of data - Upper 2 bytes: GOP index (GOP index start from 0 and increase by increments of 1 at the next I-frame. If the encoding setting is changed, it will initialize by 0.) - Lower 2 bytes: index within GOP(the number in the GOP start from 0. This increase by increments of 1 and initialized by 0 at the next I-frame.
reserved[2]	4 bytes	reserved

[Example]



```
Frame: Number = 3, Captured Frame Length = 1514, Me
Ethernet: Etype = Internet IP (IPv4),DestinationAdd
Ipv4: Src = 192.168.100.232, Dest = 192.168.100.170
Tcp: Flags=...A...., SrcPort=RTSP(554), DstPort=748
Rtsp: [Interleaved Data] Length = 1460
  InterleavedFrame: Channel = 0, 7076 bytes
    Delimiter: 36 (0x24)
    ChannelID: 0 (0x0)
    PayloadLength: 7076 (0x1BA4)
Rtp: PayloadType = dynamic, SSRC = 608868618,
  Version:       (10..............) 0x2
  Padding:       (..0.............) No paddin
  Extension:     (...0............) No header
  CSRCCount:     (....0000........) 0x0
  Marker:        (........1.......) Marker se
  PayloadType:   (.........1100000) dynamic
  SequenceNumber: 58818 (0xE5C2)
  Timestamp: 3110718251 (0xB969CB2B)
  SyncSourceId: 608868618 (0x244A990A)
  Payload: Binary Large Object (1444 Bytes)
```

```
0000  00 24 1D 24 0A 71 00 11 5F F0 03 D8 08 00 45 00
0010  05 DC FC 78 40 00 40 06 ED BF C0 A8 64 E8 C0 A8
0020  64 AA 02 2A 1D 40 87 A2 76 A9 7A 26 D4 B6 50 10
0030  19 20 55 C5 00 00 24 00 1B A4 80 E0 E5 C2 B9 69
0040  CB 2B 24 4A 99 0A 02 02 01 01 80 1B 00 00 00 B3
0050  02 10 E1 63 12 4C 18 D4 38 00 02 01 02 01 00 00
0060  00 01 21 9A 78 33 E0 38 21 4C 79 A8 A3 2C 7C 40
      34 10 03 41 [illegible] [illegible] 83 15 B9 [illegible] [illegible] [illegible] [illegible] E7 DC 77
      81 4E 36 CE [illegible] [illegible] 00 D3 10 [illegible] [illegible] [illegible] [illegible] 76 C3 C0
0090  D0 B0 31 DB [illegible] [illegible] [illegible] 1A 64 60 AB 7F 4F 02 C4 6F
      15 BC 0F 9A [illegible] [illegible] [illegible] 03 8E [illegible] [illegible] [illegible] [illegible] 1F 85 F2
      89 D3 7F E1 DC 03 80 03 A4 [illegible] [illegible] [illegible] [illegible] 1C 04 72
      62 82 00 45 [illegible] [illegible] [illegible] 70 11 [illegible] [illegible] [illegible] [illegible] [illegible] E5 DF
00D0  5F F0 43 13 [illegible] [illegible] [illegible] 01 CD [illegible] [illegible] [illegible] [illegible] [illegible] A3 62
      B3 05 35 4B [illegible] [illegible] [illegible] 38 3B 18 03 F4 FF 02 B0 D8
      38 08 E6 FC [illegible] [illegible] [illegible] A6 FC B1 61 92 54 78 08 53
0100  57 2C 3C 01 0A 6B 65 B9 92 67 6F BB F3 AF 6F F0
0110  62 3A 3C 04 29 EF 96 1E 00 85 35 F2 D3 23 C0 10
0120  A7 BE 58 78 02 14 F2 65 BD 55 57 98 7C 78 02 14
0130  F2 65 87 80 21 4F 26 5B C7 80 21 4F 26 58 78 02
0140  14 F2 65 BF BF 1C 32 3C 01 0A 79 32 C3 C0 10 A7
```

5. SENDING AUDIO DATA

[Description]

- This Chapter defines how to send data from the client to the DVR/NVR.
- The method to send audio data is based on u-law which sends the audio data by 800 bytes every 100 milliseconds. We recommend sending data as u-law because of latency issues

[Sending packet format]

RTSPHEADER	SENDAUDIOHEADER

[Header field]

Header Field	Length	Description
version	1 byte	Default set 0x0f
Type	1 byte	Default set 0x0f
datalen	2 byte	Length of the audio data.

- Sending Audio Data can be described in pseudo code listed below.

```
typedef struct _RTSPHEADER
{
        unsigned char    Delimiter;
        unsigned char    ChannelID;
        short int        PlayloadLen;

} RTSPHEADER;


typedef struct _SENDAUDIOHEADER
{
        unsigned char version;
        unsigned char type;
        short int datalen;

    } SENDAUDIOHEADER;

        unsigned char AudioData[1024*8];
        RTSPHEADER RtspHeader;
        SENDAUDIOHEADER audioheader;

        RtspHeader.Delimiter = 36;
        RtspHeader.ChannelID = 1;
        RtspHeader.PlayloadLen = htons(nSize+sizeof(SENDAUDIOHEADER));

        audioheader.version     = 0x0f;
        audioheader.type        = 0x0f;
        audioheader.datalen     = htons(nSize);

        memcpy(AudioData, &RtspHeader, sizeof(RTSPHEADER));
        memcpy(AudioData+sizeof(RTSPHEADER), &audioheader, sizeof(SENDAUDIOHEADER));
        memcpy(AudioData+sizeof(RTSPHEADER)+sizeof(SENDAUDIOHEADER), pData, nSize);
        int nSendLen = send(SOCKET, (char*)AudioData,
        sizeof(RTSPHEADER)+sizeof(SENDAUDIOHEADER)+nSize, 0);
```

6. SECURITY

This device supports several encryption methods. The user can chose an encryption method and a range.
The client goes through the adequate decryption process following the Security-Type within OPTION response header of the device.
The detail explanation of Security-Type is as below:
- Security-Type : <encoding type>/[video1;][video2;][audio;]

<Encoding type>: SEED_128, …

video1: H264 video

video2: JPEG video

Audio: audio input, MIC

e.g) OPTION response

S->C:RTSP/1.0 200 OK[CRLF] CSeq: 10[CRLF]

Public: DESCRIBE, SETUP, TEARDOWN, PLAY[CRLF]
Security-Type: SEED_128/video1;[CRLF]

6.1.1 SEED_128 / H264 Video (I frame, P frame, RI frame)

When SEED_128/video1 is contained in the Security-Type, H.264 Video data is encrypted.
At this time, H264 Video is encrypted 128 bytes after ICODEC_HEADER in the frame.

When the packet type is I frame (RI frame) or P frame, you should decrypt 128 bytes after header, which is not contained.

[Sending packet format]

RTSPHEADER	RTPHEADER	ICODEC_HEADER	128 bytes	FRAMEDATA

Ref)
The middle packet frame data is not encrypted.
When the frame size is less than 128 bytes, the remained data, which are not divided by 16 bytes, are not encrypted.
ex) If the frame size is 120 Bytes, 116 bytes are encrypted, remained 8 bytes is not encrypted.

6.1.2 SEED KEY

The key used for SEED algorithm is consisted by User Password, User ID, MAC Address.
In case of combined characters exceeds 16 bytes, it throw out exceeded characters. However combined characters is less than 16 bytes, it fill the remained characters as a 0x00.

For example, if the information is ID: ADMIN / P.W: 1234 / MAC Address: 00:11:60:FF:00:73, the key value is as below:
0x31 0x32 0x33 0x34 0x41 0x44 0x4d 0x49 0x4e 0x00 0x11 0x60 0xff 0x00 0x73 0x00(padding)

If the information are ID: ADMIN / P.W: ADMIN1234 / MAC Address: 00:11:60:FF:00:73, The key value is as below:
0x41 0x44 0x4d 0x49 0x4e 0x31 0x32 0x33 0x34 0x41 0x44 0x4d 0x49 0x4e 0x00 0x11

7. FAQ

7.1 Why are Images of ANF, ATM and UTM fractured?

ANF, ATM and UTM model are based on I-Frame Reference method which is ITX security Optimized Codec. The reason why we made this strategy is because we want to support many kind of IT client device robustly. If you want to solve fracture problem, you can refer to the example below.
(https://sites.google.com/site/sdkfaq/supportfile/ITXSDKExample_r33177.zip?attredirects=0&d=1)

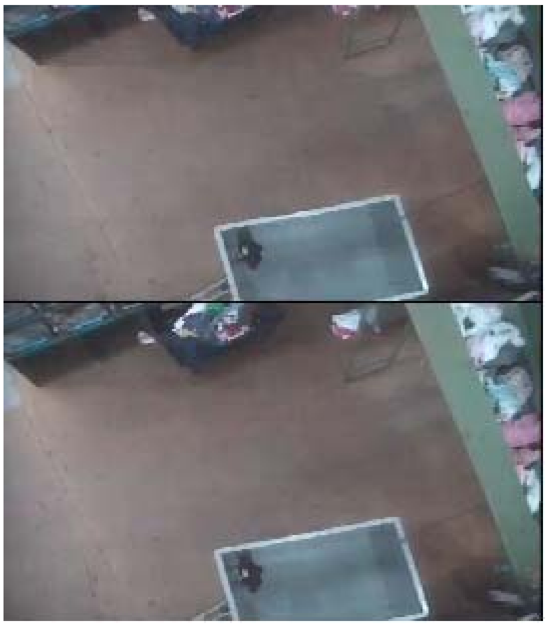
Follow the instruction below

1. Install the ITXCodecInstaller.
2. Start the ITXSDKExample.vcproj
3. Modify the ipAdder, userID, userPass, selectType and rtspPort in the main.cpp
4. Test the example program.

7.2 Why does not play the scene of ANF, ATM, OTM and SNF in the VLC Player?

The initial ITX Security Products does not be concerned about the SPS, PPS Information of H.264 format, so it cannot be played in the VLC Player. If you want to play the scene of those products, you should attach the temporary SPS, PPS information and decode the scene. (Here is the example of the c code. https://sites.google.com/site/sdkfaq/supportfile/itx_sps_pps.c?attredirects=0&d=1)

7.3 Why does the sequence be taken the 2 sub-field in the ANF and ATM models?



The ANF and ATM models are the interlacing model. If you want to see the sequence properly, you should deinterlace the sequence. Refer to below source code.

(https://sites.google.com/site/sdkfaq/supportfile/ITXSDKExample_r33177.zip?attredirects=0&d=1)