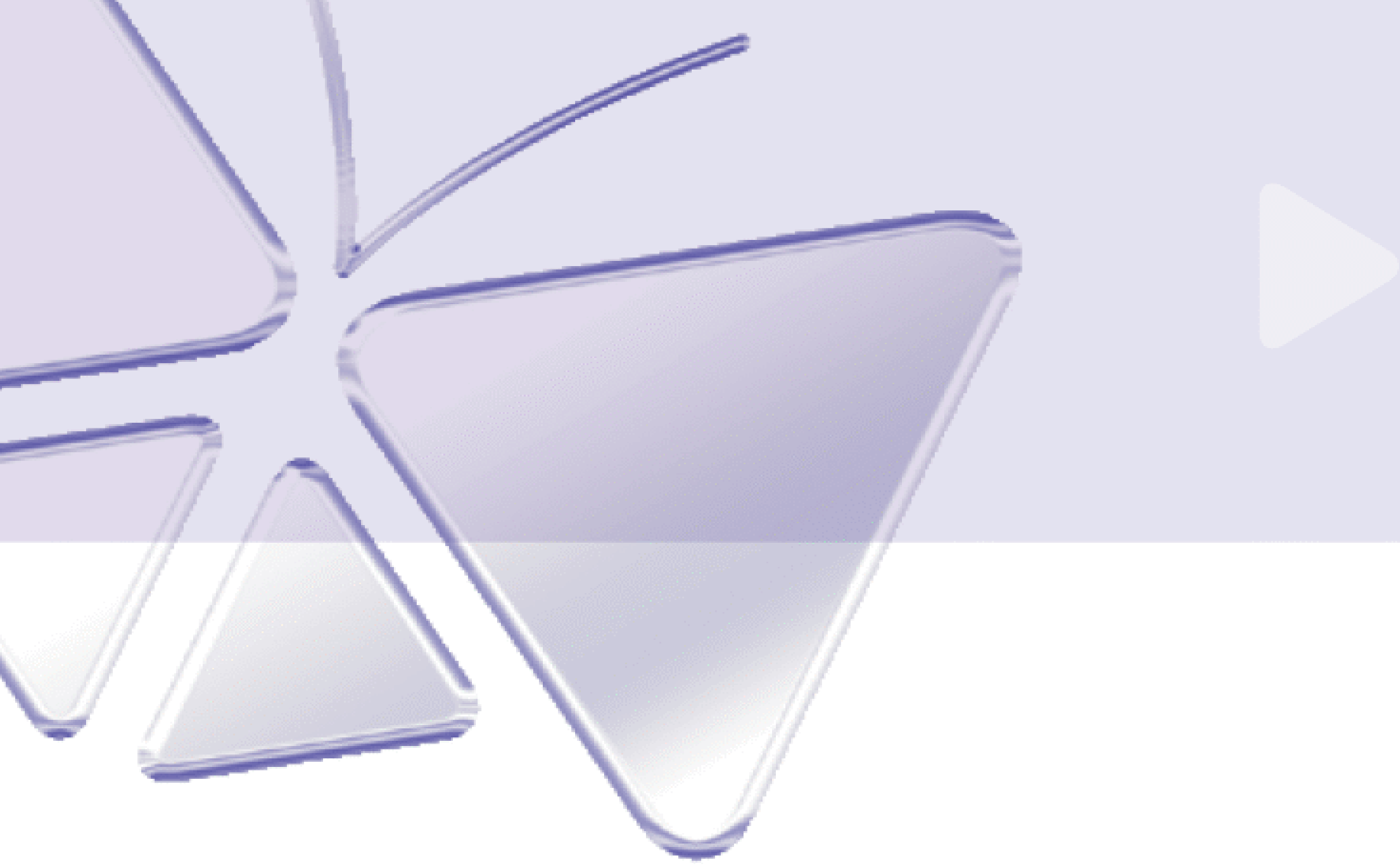


ACTi SDK-10000
C Library Edition
V1.2

Programming Guide

ACTi

www.acti.com

ACTi SDK-10000

Information provided in this manual is intended to be accurate and reliable. However, the original manufacturer assumes no responsibility for its use, or for any infringements upon the rights of third parties that may result from its use.

V1.2 Edition Sep. 2007

Table of Contents

1 Overview

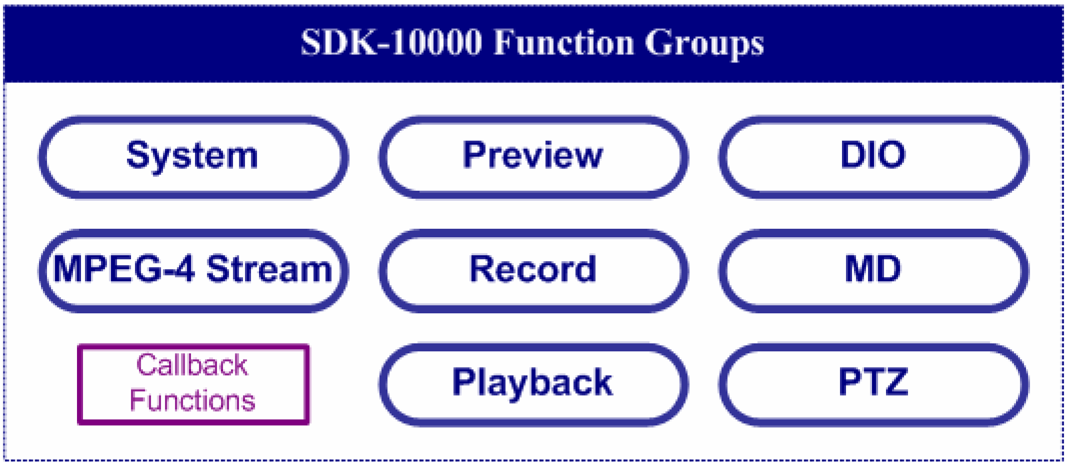
Introduction

This material covers SDK architecture, data structure and procedures to illustrate the mechanisms to integrate the IP Surveillance devices. The content of this material is designed to lead the programmers go through the flow of the SDK and design their own application with supplied functions; they are organized in topics so that programmers may find the topics they want directly.

Please refer to Programming Guide for detailed API references.

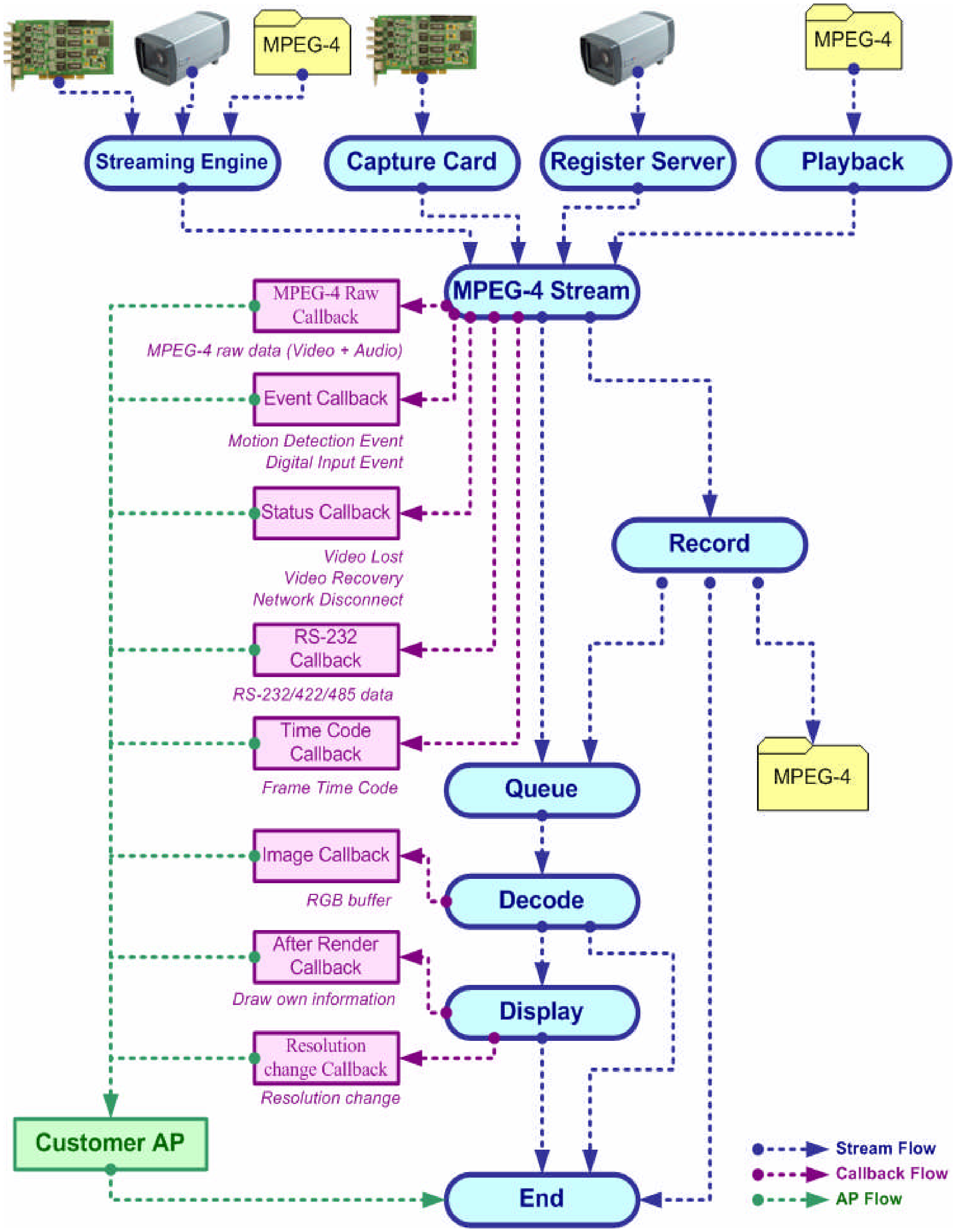
SDK Function Groups

The whole SDK can be divided into following function groups.



Architecture

SDK architecture and data flow is described as follow:



Application Type

Based on the architecture and data flow, users may develop following application type:

1. **Full-featured Surveillance system**: preview, record, playback, DIO event, MD event and PTZ functions
2. **Background recording**: record without preview. The stream can be configured as unicast or multicast mode
3. **Connection with event handling only**: connection only, wait for digital input or motion detection event; when the event triggered, then starts streaming and record the event
4. **Background recording with RGB buffer**: record without preview, receives RGB buffer to run user-defined motion detection algorithm at the same time
5. **Process MPEG-4 video stream**: advanced users may acquire MPEG-4 video stream and process by themselves. Related video, audio and audio+video callback functions are provided
6. **User-defined information on screen**: user may use after render callback function to draw user-defined information on preview window, including OSD text, draw video intelligence information

Topics

Streaming Client Library is developed for MPEG-4 Video Network Streaming Application. It contains following abilities:

- Registration with Unicast / Multicast
- Preview / Record / Playback
- DIO Event Handling
- Motion Detection Event Handling
- PTZ Integration
- Status Callback
- IP Quad Integration
- Advanced Topics
    - Gets MPEG-4 data via MPEG-4 callback function
    - Gets RGB via image callback function
    - ACTi MPEG-4 Time code format
    - Decode I Frame Only
    - Save ACTi MPEG-4 raw data into AVI format
    - Gets RGB via image callback function

What’s New?

Following lists the new contents in this release:

- (v1.0.07) Add ATCP10, AMCST10, A4100 adaptor
- (v1.0.07) Add FAVI adaptor: may records AVI file format
- (v1.2.08) Add support to megapixel MPEG-4 decoding
- (v1.2.08) Add support to MJPEG decoding
- (v1.2.08) Add support to Intel IPP decoder
- (v1.2.08) Add video image flip and mirror
- (v1.2.08) Add DI handling function

Compiling and Linking

This section describes the compiling and linking options.

Include Files ${SDK DIR}\SDK\Include

File	Description
`SDK10000.h`	SDK 10000 include file.

Library Files ${SDK DIR}\SDK\LIB

File	Description
`KMpeg4.lib`	SDK 10000 library file.

Runtime DLL Files ${SDK DIR}\SDK\DLL

File	Description
`KMpeg4.dll`	MPEG-4 Kernel dll.
`ATCP10.dll`	AVC adaptor on networking module for TCP 10 data.
`ATCP20.dll`	AVC adaptor on networking module for TCP 20 data.
`AMCST10.dll`	AVC adaptor on networking module for Multicast 10 data.
`AMCST20.dll`	AVC adaptor on networking module for Multicast 20 data.
`ARAW`	AVC adaptor for playback.
`FRAW.dll`	File adaptor on raw data format
`FFMCODEC.dll`	MPEG-4 software CODEC
`XVIDCODEC.dll`	MPEG-4 software CODEC

.

Sample Codes ${SDK DIR}\SDK\Samples

SDK-10000 v1.2 sample programs can be reached at **${SDK Directory}\SDK\Samples**

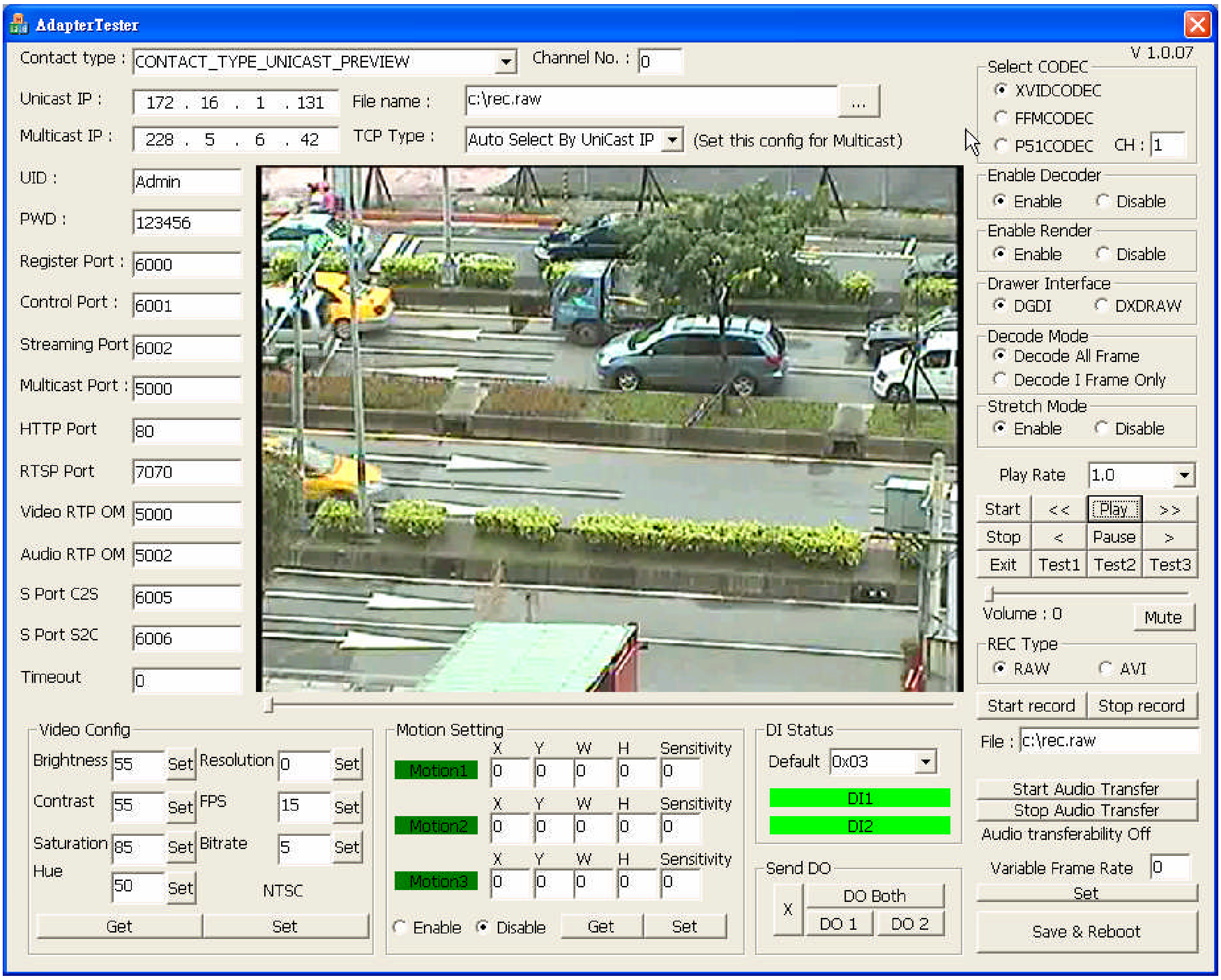
SDK-10000 v1.2 provides 8 sample codes:

1. StreamSample : preview, record, motion, DIO, etc.
2. PlaybackSample: play forward/backward, fast forward/backward, step-by-step, etc.
3. DecodeSample: connects to the device, receives MPEG-4 raw data, decode it to RGB buffer, display the RGB buffer, save to BMP file.
4. URLSample: Allow you to send URL request and receive URL response from video server.
5. ArchivePlay: Allow you to preview a raw/mp4 file with playback functions.
6. SearchSample: Search for connectable devices.
7. PTZSample: To demonstrate how you can get PTZ command from PTZParser library and send it through SDK-10000.
8. MediaConverter: To demonstrate how you can convert raw data file to avi.

StreamSample Program

StreamSample codes demonstrate following functions:

1. Search Server
2. Connection mode: unicast, multicast
3. Preview, Record
4. Motion Detection set up and trigger
5. DI trigger and sends DO
6. Audio functions



AdapterTester

V 1.0.07

Contact type : CONTACT_TYPE_UNICAST_PREVIEW
Channel No. : 0
Unicast IP : 172 . 16 . 1 . 131
File name : c:\rec.raw
Multicast IP : 228 . 5 . 6 . 42
TCP Type : Auto Select By UniCast IP (Set this config for Multicast)

UID : Admin
PWD : 123456
Register Port : 6000
Control Port : 6001
Streaming Port 6002
Multicast Port : 5000
HTTP Port 80
RTSP Port 7070
Video RTP OM 5000
Audio RTP OM 5002
S Port C2S 6005
S Port S2C 6006
Timeout 0

Select CODEC
● XVIDCODEC
○ FFMCODEC
○ P51CODEC CH : 1

Enable Decoder ● Enable ○ Disable
Enable Render ● Enable ○ Disable
Drawer Interface ● DGDI ○ DXDRAW
Decode Mode
● Decode All Frame
○ Decode I Frame Only
Stretch Mode ● Enable ○ Disable

Play Rate 1.0
Volume : 0
REC Type ● RAW ○ AVI
File : c:\rec.raw
Audio transferability Off
Variable Frame Rate 0

Video Config
Brightness 55 Set Resolution 0 Set
Contrast 55 Set FPS 15 Set
Saturation 85 Set Bitrate 5 Set
Hue 50 Set NTSC

Motion Setting

	X	Y	W	H	Sensitivity
Motion1	0	0	0	0	0
Motion2	0	0	0	0	0
Motion3	0	0	0	0	0

○ Enable ● Disable

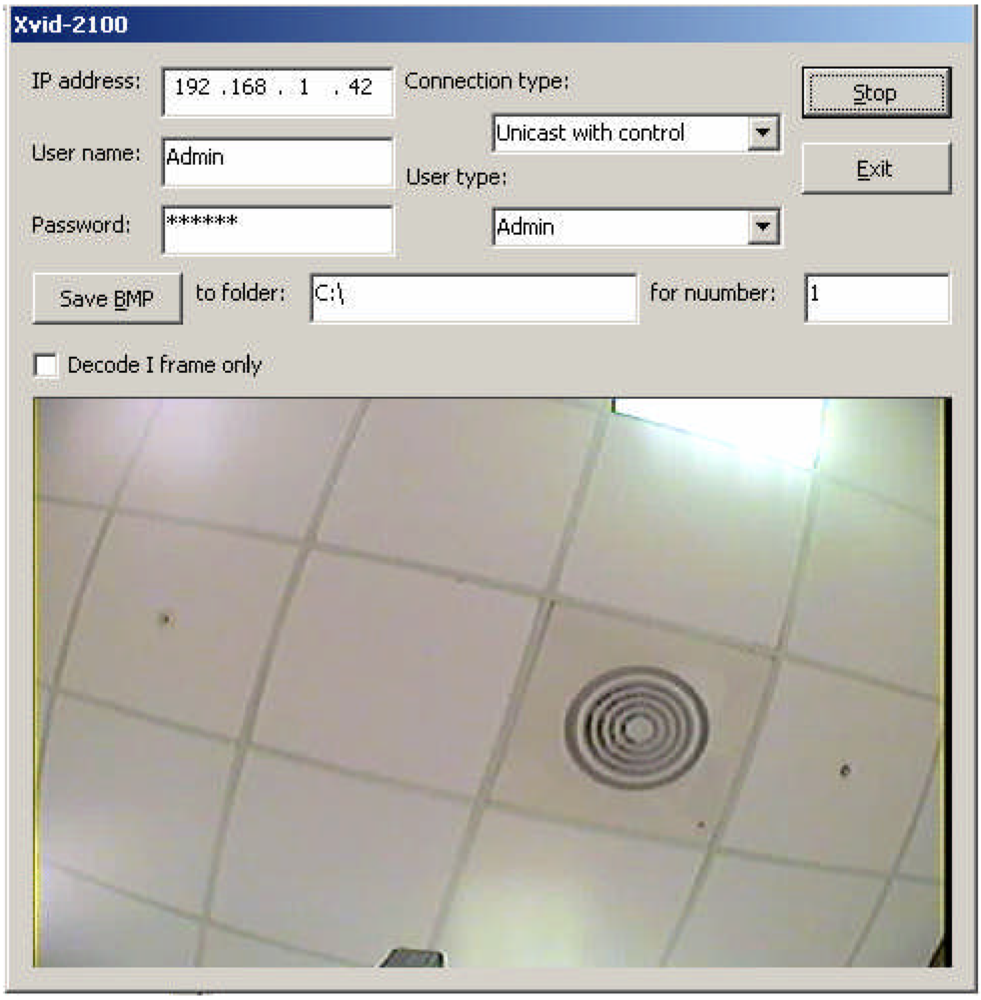
DI Status
Default 0x03
DI1
DI2

Send DO

DecodeSample Program

PlaybackSample codes demonstrate following functions:

1. Decode MPEG-4 into RGB buffer
2. Display RGB buffer onto screen
3. Save RGB buffer to BMP file
4. Decode I-frame only



Xvid-2100

IP address: 192 .168 . 1 . 42

Connection type: Unicast with control

User name: Admin

User type: Admin

Password: ******

to folder: C:\

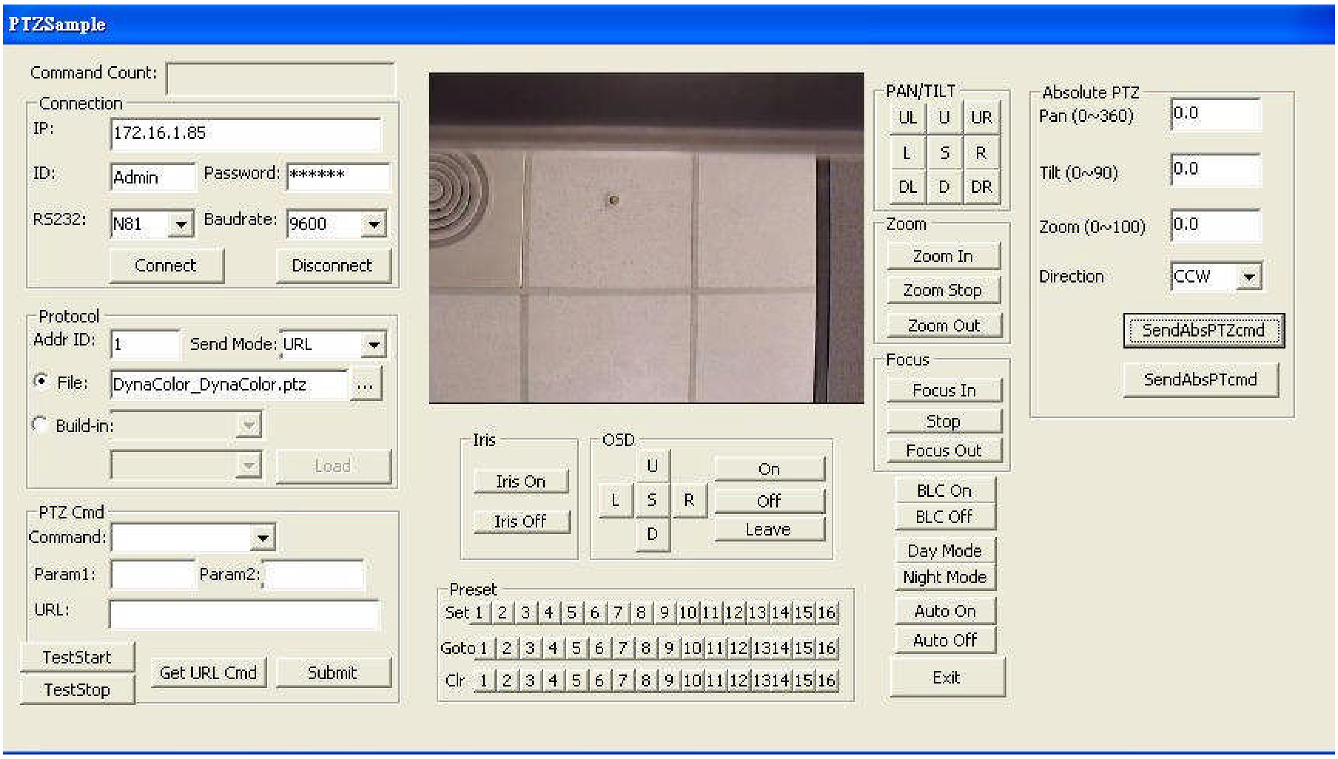
for nuumber: 1

□ Decode I frame only

PTZSample Program

PTZSample codes demonstrate following functions:

1. Read PTZ protocol files
2. Operate PTZ functions.(Most PTZ functions were updated since V1.2)
3. Demonstrate Pan, Tilt, Zoom, Focus, Iris, Preset, OSD, and Absolute PTZ functions. (Absolute PTZ functions only work with DynaColor protocols now.)
4. URL Command to send PTZ commands.
5. Get PTZ command using PTZParser library. (PTZParser was integrated into SDK V1.2 , so that is major change of PTZ APIs.).



PTZSample

Command Count:

Connection

IP: 172.16.1.85

ID: Admin Password: ******

RS232: N81 Baudrate: 9600

Protocol

Addr ID: 1 Send Mode: URL

● File: DynaColor_DynaColor.ptz

○ Build-in:

PTZ Cmd

Command:

Param1: Param2:

URL:

Absolute PTZ

Pan (0~360) 0.0

Tilt (0~90) 0.0

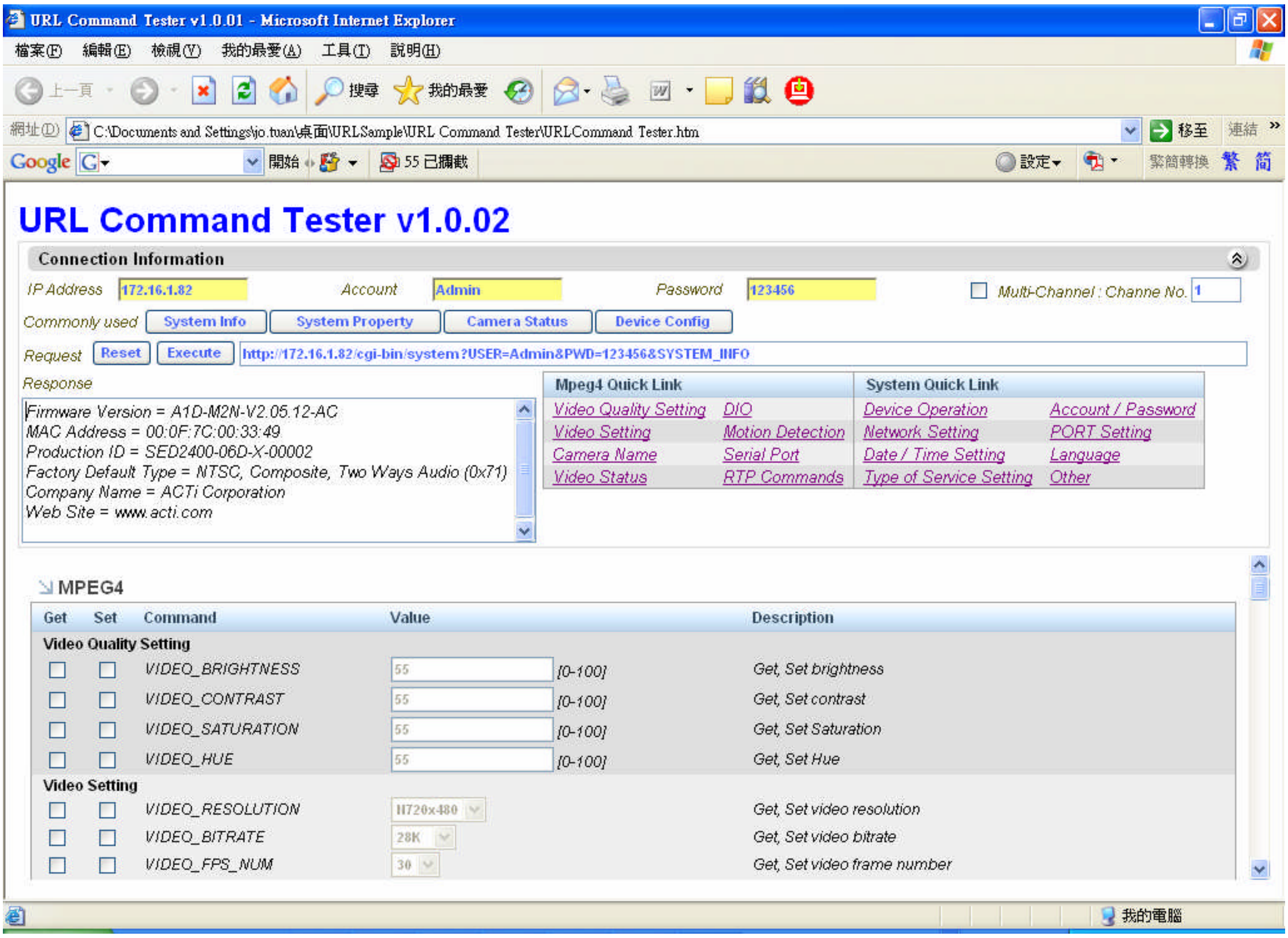
Zoom (0~100) 0.0

Direction CCW

URLSample Program

URLSample codes demonstrate following functions:

1. Send URL command request.
2. Receive URL response


URL Command Tester v1.0.01 - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
網址(D)
C:\Documents and Settings\jo.tuan\桌面\URLSample\URL Command Tester\URLCommand Tester.htm
移至
連結
Google
開始
55 已攔截
設定
繁簡轉換 繁 簡
我的電腦

URL Command Tester v1.0.02

Connection Information

IP Address 172.16.1.82 Account Admin Password 123456 □ Multi-Channel : Channe No. 1

Commonly used [System Info] [System Property] [Camera Status] [Device Config]

Request [Reset] [Execute] http://172.16.1.82/cgi-bin/system?USER=Admin&PWD=123456&SYSTEM_INFO

Response

```
Firmware Version = A1D-M2N-V2.05.12-AC
MAC Address = 00:0F:7C:00:33:49
Production ID = SED2400-06D-X-00002
Factory Default Type = NTSC, Composite, Two Ways Audio (0x71)
Company Name = ACTi Corporation
Web Site = www.acti.com
```

Mpeg4 Quick Link		System Quick Link	
Video Quality Setting	DIO	Device Operation	Account / Password
Video Setting	Motion Detection	Network Setting	PORT Setting
Camera Name	Serial Port	Date / Time Setting	Language
Video Status	RTP Commands	Type of Service Setting	Other

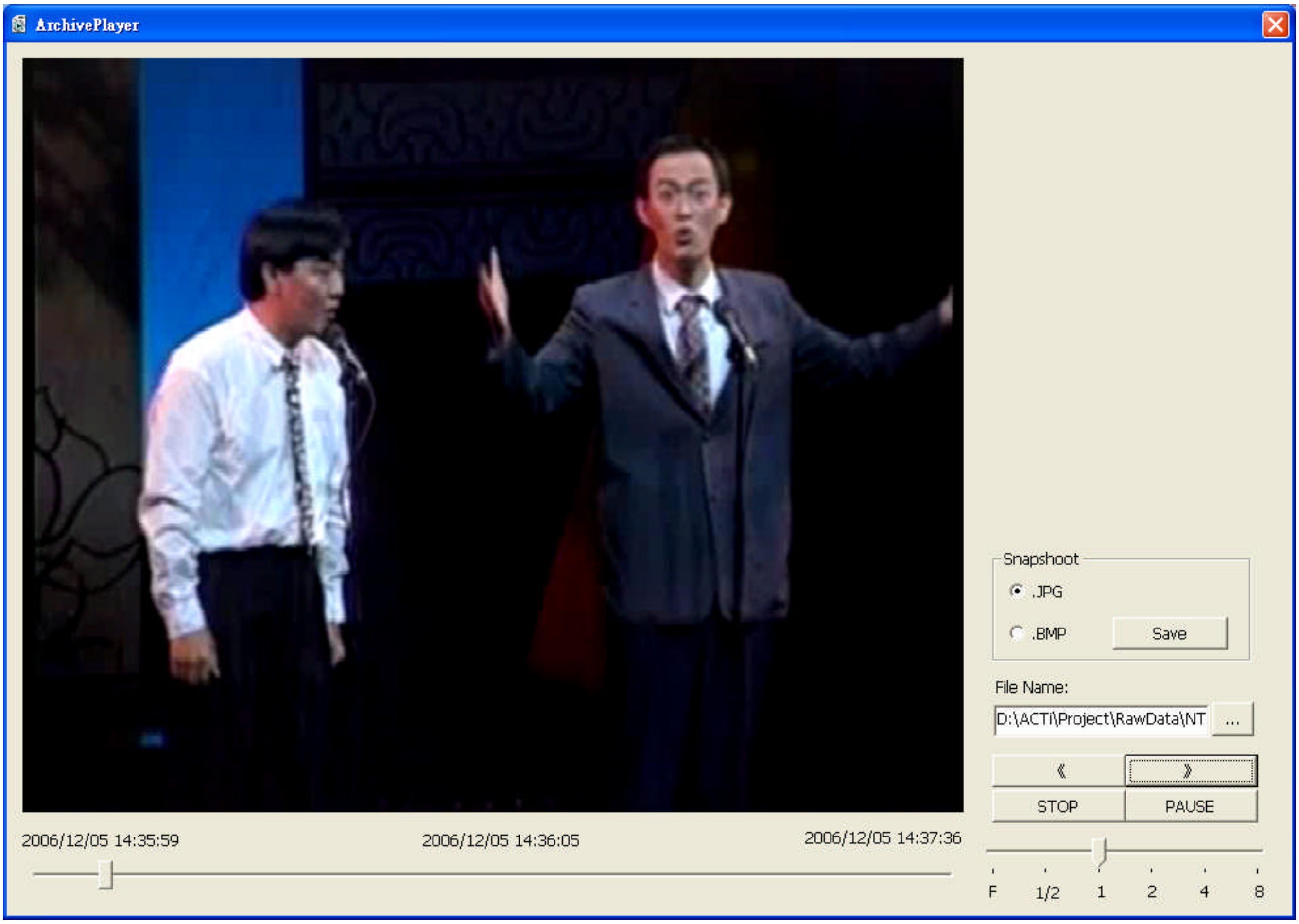
MPEG4

Get	Set	Command	Value	Description
Video Quality Setting				
□	□	VIDEO_BRIGHTNESS	55 [0-100]	Get, Set brightness
□	□	VIDEO_CONTRAST	55 [0-100]	Get, Set contrast
□	□	VIDEO_SATURATION	55 [0-100]	Get, Set Saturation
□	□	VIDEO_HUE	55 [0-100]	Get, Set Hue
Video Setting				
□	□	VIDEO_RESOLUTION	N720x480	Get, Set video resolution
□	□	VIDEO_BITRATE	28K	Get, Set video bitrate
□	□	VIDEO_FPS_NUM	30	Get, Set video frame number

ArchivePlayer Program

ArchivePlayer codes demonstrate following functions:

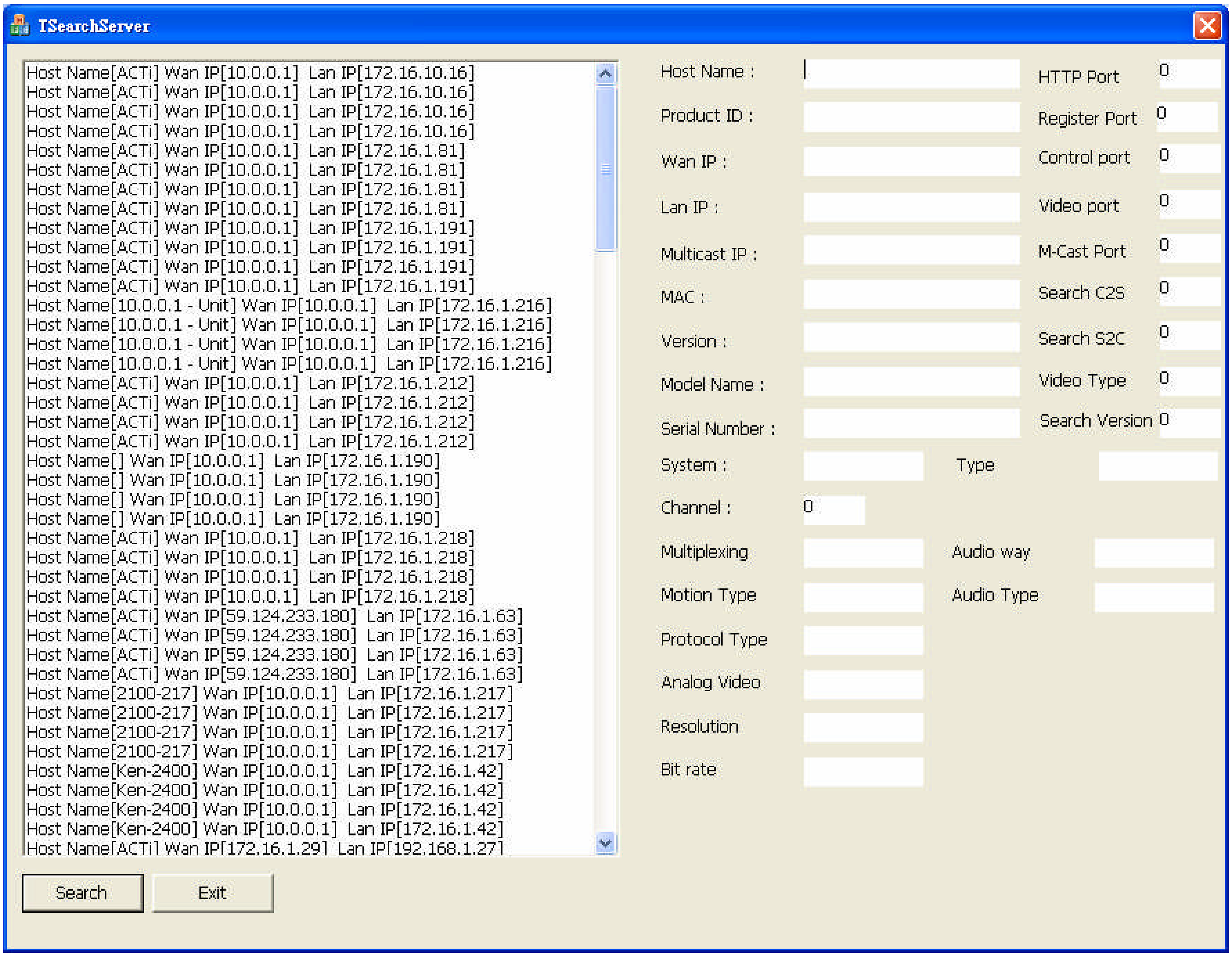
1. Snapshot with JPG&BMP format.
2. Play with different speed.
3. Preview with frame by frame.
4. Pause.
5. Seek into random position.
6. Allow to play raw/mp4 file.
7. Display text on video frame.



SearchSample Program

SearchSample codes demonstrate following functions:

1. Search for connectable devices..



```
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.10.16]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.10.16]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.10.16]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.10.16]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.81]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.81]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.81]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.81]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.191]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.191]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.191]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.191]
Host Name[10.0.0.1 - Unit] Wan IP[10.0.0.1]  Lan IP[172.16.1.216]
Host Name[10.0.0.1 - Unit] Wan IP[10.0.0.1]  Lan IP[172.16.1.216]
Host Name[10.0.0.1 - Unit] Wan IP[10.0.0.1]  Lan IP[172.16.1.216]
Host Name[10.0.0.1 - Unit] Wan IP[10.0.0.1]  Lan IP[172.16.1.216]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.212]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.212]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.212]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.212]
Host Name[] Wan IP[10.0.0.1]  Lan IP[172.16.1.190]
Host Name[] Wan IP[10.0.0.1]  Lan IP[172.16.1.190]
Host Name[] Wan IP[10.0.0.1]  Lan IP[172.16.1.190]
Host Name[] Wan IP[10.0.0.1]  Lan IP[172.16.1.190]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.218]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.218]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.218]
Host Name[ACTi] Wan IP[10.0.0.1]  Lan IP[172.16.1.218]
Host Name[ACTi] Wan IP[59.124.233.180]  Lan IP[172.16.1.63]
Host Name[ACTi] Wan IP[59.124.233.180]  Lan IP[172.16.1.63]
Host Name[ACTi] Wan IP[59.124.233.180]  Lan IP[172.16.1.63]
Host Name[ACTi] Wan IP[59.124.233.180]  Lan IP[172.16.1.63]
Host Name[2100-217] Wan IP[10.0.0.1]  Lan IP[172.16.1.217]
Host Name[2100-217] Wan IP[10.0.0.1]  Lan IP[172.16.1.217]
Host Name[2100-217] Wan IP[10.0.0.1]  Lan IP[172.16.1.217]
Host Name[2100-217] Wan IP[10.0.0.1]  Lan IP[172.16.1.217]
Host Name[Ken-2400] Wan IP[10.0.0.1]  Lan IP[172.16.1.42]
Host Name[Ken-2400] Wan IP[10.0.0.1]  Lan IP[172.16.1.42]
Host Name[Ken-2400] Wan IP[10.0.0.1]  Lan IP[172.16.1.42]
Host Name[Ken-2400] Wan IP[10.0.0.1]  Lan IP[172.16.1.42]
Host Name[ACTi] Wan IP[172.16.1.29]  Lan IP[192.168.1.27]
```

Host Name :   HTTP Port 0

Product ID :   Register Port 0

Wan IP :   Control port 0

Lan IP :   Video port 0

Multicast IP :   M-Cast Port 0

MAC :   Search C2S 0

Version :   Search S2C 0

Model Name :   Video Type 0

Serial Number :   Search Version 0

System :   Type

Channel : 0

Multiplexing   Audio way

Motion Type   Audio Type

Protocol Type

Analog Video

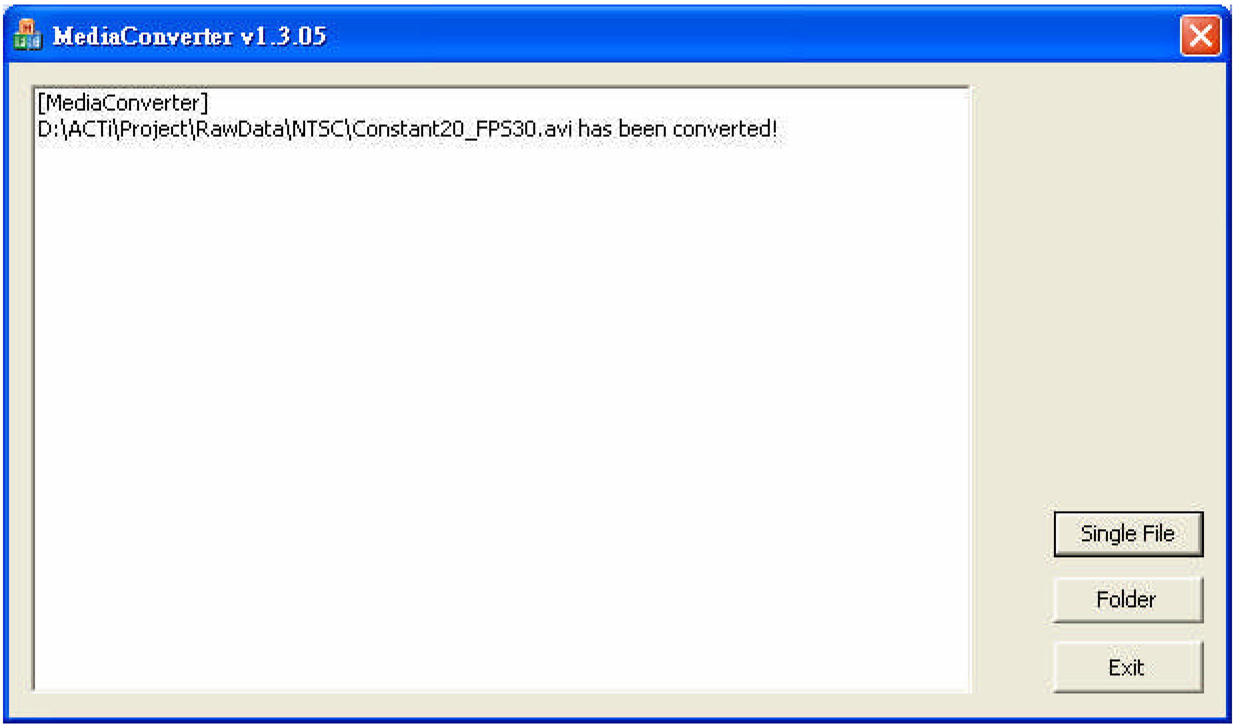
Resolution

Bit rate

MediaConverter Program

MediaConverter codes demonstrate following functions:

1. Convert raw file to avi..



```
[MediaConverter]
D:\ACTi\Project\RawData\NTSC\Constant20_FPS30.avi has been converted!
```

2 Search Device

Device Locator Architecture

The section describes the mechanism on how to search ACTi's IP surveillance products on network. With this mechanism, you can locate the devices on the network, then use URL commands to operate or manage those devices.

The function sends out a broadcast message, ACTi's devices respond with detailed information, application then parse the replied information and parse the content with **NET_SEARCHSERVER** data structure.

Search Device

Steps to detect ACTi IP Surveillance products are listed as follow:

1. Call `netSearchServer()`
2. Receive and decodes with `NET_SEARCHSERVER`

> ⚠ **NOTE:** The second parameter of `netSearchServer()` indicates the maximum total number to be reached in the network; for example, if this parameter is set to 10, and there are 20 devices in the same network, then this function returns when it reaches the first 10 devices in the network.
>
> Default timeout value is 20 seconds

```
typedef struct tagSearchServer {
     char szHostName[24];        // [OUT] Host Name          : ASCII Z STRING
     char szProductID[8];        // [OUT] ProductID          : ASCII Z STRING
     char szWanIp[16];           // [OUT] WAN IP             : ASCII Z STRING
     char szLanIp[16];          // [OUT] LAN IP         : ASCII Z STRING
     char szMultiCastIp[16];     // [OUT] MULTICAST IP       : ASCII Z STRING
     char szMac[32];             // [OUT] MAC            : ASCII Z STRING
     char cType;                 // [OUT] Bit0~3         : 1: Composite, 2: S-Video
                                 // [OUT] Bit4~7       : 1: Video Server, 2: IPCam
     char    dummy1;
     char    dummy2;
     char    dummy3;
     char    Version[32];
     WORD wHPort;
     WORD wSPortC2S;             // [IN]  Search   Port (Client to Server)
     WORD wSPortS2C;             // [IN]  Search   Port (Server to Client)
     WORD wRPort;                // [IN]  Register Port
```

```
    WORD wCPort;                   // [IN]  Control  Port
    WORD wVPort;                   // [IN]  Video    Port
    WORD wMPort;                   // [IN]  MultiCastPort
    WORD    dummy4;

} NET_SEARCHSERVER;



WORD dwRet ;
NET_SEARCHSERVER ServerList[MAXSERVERLIST];
    // Receive data Structure

DWORD dwTotalNum = MAXSERVERLIST ;

dwRet = netSearchServer((char*) ServerList, &dwTotalNum);

for (DWORD i = 0; i< dwTotalNum; i++) {
    szHostName[i]     = ServerList[i].szHostName ;
        // Get the Host Name From Result Structure
    szProductID[i]         = ServerList[i].szProductID ;
        // Get the Product ID From Result Structure
    szWanIp[i]             = ServerList[i].szWanIp ;
        // Get the WanIp From Result Structure
    szLanIp[i]             = ServerList[i].szLanIp
        // Get the LanIp From Result Structure
    szMultiCastIp[i]  = ServerList[i].szMultiCastIp ;
        // Get the MultiCastIp From Result Structure
    szMac[i]          = ServerList[i].szMac ;
        // Get the Mac Address From Result Structure
    szVersion[i]      = ServerList[i].Version ;
        // Get the Firmware Version From Result Structure
    wRPort[i]              = ServerList[i].wRPort;
        // Get the Register Port From Result Structure
    wCPort[i]              = ServerList[i].wCPort;
        // Get the control Port From Result Structure
    wVPort[i]              = ServerList[i].wVPort;
        // Get the Streaming Port From Result Structure
    wMPort[i]              = ServerList[i].wMPort;
        // Get the Multicast Port From Result Structure
    wHPort[i]              = ServerList[i].wHPort;
        // Get the Http Port From Result Structure
}
```

How to detect device

This section describes how to detect, manage and configure IP devices. All commands are operated with URL Commands, you can use the functions we suggested (xmlhttp) or you can find HTTP-related functions by yourselves.

Please also refer to the Appendix for the complete ACTi URL Command listing.

System Information

Steps to detect product System Information are listed as follow:

Sample:

```
// you should get HANDLE by KOpenInterface before Preview
     HANDLE hK =  KOpenInterface();

// Prepare USER_INFO data structure by filling IP address, account, password.
     MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
     …
     KSetMediaConfig(hK, &mcc);
     KConnect(hK);

     strURL = 'http://192.168.1.100:80' ;
     strURL = '/cgi-bin/system?USER=Admin&PWD=123456&SYSTEM_INFO' ;

     char szResultBuf[1024] = {0};
     DWORD dwResultLen;
     KSendURLCommand( hK, strURL, szResultbuf, dwResultLen) ;

//   Firmware Version = A1D-M2N-V2.03.02-NB
//   MAC Address = 00:0F:7C:00:1A:47
//   Production ID = SED2400-05I-1-00034
//   Factory Default Type = NTSC, Composite, Two Ways Audio (0x71)
```

System Property

Steps to detect product System Property are listed as follow:

Sample:

```
// you should get HANDLE by KOpenInterface before Preview
     HANDLE hK =  KOpenInterface();

// Prepare USER_INFO data structure by filling IP address, account, password.
     MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
     …
     KSetMediaConfig(hK, &mcc);
     KConnect(hK);

     strURL = 'http://192.168.1.100:80' ;
     strURL = '/cgi-bin/system?USER=Admin&PWD=123456& SYSTEM_PROPERTY ' ;

     char szResultBuf[1024] = {0};
     DWORD dwResultLen;
     KSendURLCommand( hK, strURL, szResultbuf, dwResultLen) ;

//   SYSTEM='E'
//   TYPE='A'
//   NO_OF_CHANNEL='01'
//   MULTIPLEXING='X'
//   NO_OF_AUDIO_WAYS='2'
//   AUDIO_TYPE='PCM'
//   MOTION_TYPE='0'
//   PROTOCOL_TYPE='2'
```

Video Color Adjustments

This section describes on how to adjust video color using URL Commands.

Hue, Brightness, Contrast Setting

Steps to Gets/Sets product Video Property are listed as follow:

1. Initial `KMpeg4` Object
2. Gets color setting.
3. Set new setting

Sample:

```
typedef struct structural_MEDIA_VIDEO_CONFIG
{
DWORD dwTvStander;          ///< 0:NTSC 1:PAL
DWORD dwVideoResolution;    ///< See the definition above
DWORD dwBitsRate;           ///< See the definition above
DWORD dwVideoBrightness;    ///< 0 ~ 100 : Low ~ High
DWORD dwVideoContrast;      ///< 0 ~ 100 : Low ~ High
DWORD dwVideoSaturation;    ///< 0 ~ 100 : Low ~ High
DWORD dwVideoHue;           ///< 0 ~ 100 : Low ~ High
DWORD dwFps;                ///< 0 ~ 30 frame pre second
} MEDIA_VIDEO_CONFIG;

// you should get HANDLE by KOpenInterface before Preview
    HANDLE hK =  KOpenInterface();

// Prepare USER_INFO data structure by filling IP address, account, password.
    MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
    …

    KSetMediaConfig(hK, &mcc);
    KConnect(hK);

// Get current color setting
    MEDIA_VIDEO_CONFIG mvc;
    KGetVideoConfig(hK, &mvc);
```

```
//   To Set the Video Property
     KSetHue(hK, 10)
     KSetBrightness(hK, 20);
     KSetContrast(hK, 30);
```

Video Setting Configuration

Setup Resolution, Frame Rate, Bit Rate

Steps to Gets/Sets product Video Setting are listed as follow:

Sample:

```
enum BITRATE_TYPES    /** Bitrate Types */
{
BITRATE_28K,          ///< #0# - 28K Bits per second
BITRATE_56K,          ///< #1# - 56K Bits per second
…
BITRATE_3000K         ///< #12# - 3M Bits per second
}

enum RESOLUTION_TYPES /** Resolution Types */
{
NTSC_720x480,         ///< #0# - NTSC - 720 x 480
NTSC_352x240,         ///< #1# - NTSC - 352 x 240
…
PAL_176x144           ///< #5# - PAL  - 176 x 144
}

typedef struct structural_MEDIA_VIDEO_CONFIG
{
DWORD dwTvStander;        ///< 0:NTSC 1:PAL
DWORD dwVideoResolution;  ///< See the definition above
DWORD dwBitsRate;         ///< See the definition above
DWORD dwVideoBrightness;  ///< 0 ~ 100 : Low ~ High
DWORD dwVideoContrast;    ///< 0 ~ 100 : Low ~ High
DWORD dwVideoSaturation;  ///< 0 ~ 100 : Low ~ High
DWORD dwVideoHue;         ///< 0 ~ 100 : Low ~ High
DWORD dwFps;              ///< 0 ~ 30 frame pre second
} MEDIA_VIDEO_CONFIG;

// you should get HANDLE by KOpenInterface before Preview
    HANDLE hK =  KOpenInterface();

// Prepare USER_INFO data structure by filling IP address, account, password.
    MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
    …
    KSetMediaConfig(hK, &mcc);
    KConnect(hK);

// Get current color setting
    MEDIA_VIDEO_CONFIG mvc;
    KGetVideoConfig(hK, &mvc);

//  To Set the Video Property
    KSetResolution(hK, 10)     // 0~5
    KSetFPS(hK, 30);
    KSetBitRate(hK, 30);       // 0~12
```

Save and Reboot

The section describes the mechanism on how to search ACTi's IP surveillance products on network. With this mechanism, you can locate the devices on the network, then use URL commands to operate or manage those devices.

The function sends out a broadcast message, ACTi's devices repond with detailed information, application then parse the replied information and parse the content with NET_SEARCHSERVER data structure.

Execute Save and Reboot Command

Steps to execute Save and Reboot Video device are listed as follow:

Sample:

```
// you should get HANDLE by KOpenInterface before Preview
     HANDLE hK =  KOpenInterface();

// Prepare USER_INFO data structure by filling IP address, account, password.
     MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
     …
     KSetMediaConfig(hK, &mcc);
     KConnect(hK);

     KSaveReboot(hK);
```

3 Preview / Record / Playback

Preview / Record Architecture

This material covers SDK architecture, data structure and sample programs to illustrate the methods to integrate ACTi's IP Surveillance products.

Register to IP devices

Steps to register to ACTi's device:

1. Call `KOpenInterface()` to get KMpeg4 handle.
2. Prepare IP address, port number, account, password, contact type..
3. Call `KSetMediaConfig(HANDLE, MEDIA_CONNECTION_CONFIG)` to set connect config.
4. Call `KConnect(HANDLE).`
5. Call `KStartStreaming(HANDLE)` to get ready to receive.

```
// you should get HANDLE by KOpenInterface before Preview
     HANDLE hK = KOpenInterface();

// Set call back functions
     KSetRawDataCallback(hK, id, fnRawCallback);

// Prepare USER_INFO data structure by filling IP address, account, password.
     MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
     …
     KSetMediaConfig(hK, &mcc);
     KConnect(hK);

// Start Streaming
     KStartStreaming(hK);
```

Preview Operations

Preview with Unicast Mode

Steps to start preview with unicast mode include:

1. Set contact type as `CONTACT_TYPE_UNICAST_PREVIEW`;
2. Register to the IP devices
3. Call `KPlay(HANDLE)` to start receive data.

```
// you should get HANDLE by KOpenInterface before Preview
     HANDLE hK =  KOpenInterface();

// Set call back functions
     KSetRawDataCallback(hK, id, fnRawCallback);

// Prepare USER_INFO data structure by filling IP address, account, password.
     MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
     mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
     …
     KSetMediaConfig(hK, &mcc);
     KConnect(hK);

// Start Streaming
     KStartStreaming(hK);
// Start receiving data from KMpeg4
     KPlay(hK);
```

Preview with Audio

Steps to register to ACTi's device:

1. Call `KOpenInterface()` to get KMpeg4 handle.
2. Prepare IP address, port number, account, password, contact type..
3. Call `KSetMediaConfig(HANDLE, MEDIA_CONNECTION_CONFIG)` to set connect config.
4. Call `KConnect(HANDLE).`
5. Call `KStartStreaming(HANDLE)` to get ready to receive.
6. Call `KPlay(HANDLE)` to start receive data.
7. Set mute mode to false with `KSetMute(HANDLE, BOOL)` function
8. Set audio volume with `KSetVolume(HANDLE, int, int)` function

NOTE:

```
/
// Register to the device
// Start Preview

//---- Set volume
     KSetVolume( hK , lLeftVolume , lReightVolume );  // set volume

//---- set to mute
     KSetMute(hK, true);        // audio is off

//---- turn audio back on
     KSetMute(hK, false);       // audio is on
```

Preview with 2-way audio

Steps to preview with 2-way audio include:

1. Call `KOpenInterface()` to get KMpeg4 handle.
2. Prepare IP address, port number, account, password, contact type..
3. Call `KSetMediaConfig(HANDLE, MEDIA_CONNECTION_CONFIG)` to set connect config.
4. Call `KConnect(HANDLE).`
5. Call `KStartStreaming(HANDLE)` to get ready to receive.
6. Call `KPlay(HANDLE)` to start receive data.
7. Start preview
8. Get Audio Token
9. Send audio sound from PC side to the device with `KStartAudioTransfer(HANDLE)` function. This function opens the speaker connected on the PC, and grab sound from the speaker and transmit to the device
10. Stop sending audio sound from PC side to the device with `KStopAudioTransfer(HANDLE)` function

IMPORTANT: One IP device has only 1 audio token; if the token is taken by one application, then no other application may acquire the audio token again. Remember to free audio token after the 2-way audio function is done.

```
// Register to the device

// Get the Audio Token
    bool bAudioToken = KGetAudioToken( hK );

// check the return value , if you get the audio token success.
    if ( bAudioToken )
    {
        KStartAudioTransfer(hK);
// start sending audio from PC to the device
// this function turns on speaker, the audio will be captured
// and transferred to the devices
    }
    KStopAudioTransfer(hK);
// Free the Audio Token Before you close connection.
    KFreeAudioToken(hK);
```

Preview with I-Frame Decoding only

This chapter describes a mechanism on how to decrease CPU loading. With this mechanism, MPEG-4 software decoder will decode I-Frame only and drops all P-Frame before decoding.

Steps to preview with I-Frame decoding only include:

1. Register to the IP device
2. Preview with `KPlay(HANDLE)`
3. Set to I-Frame decoding only with `KSetDecodeIFrameOnly(HANDLE, BOOL)` function

NOTE: With `KSetDecodeIFrameOnly(HANDLE, BOOL)` function, the CPU loading can be decreased dramatically.

IMPORTANT: `KSetDecodeIFrameOnly(HANDLE, BOOL)` function only affects preview and CPU loading; recording still records with I-frame and P-frame as setup.

```
// you should get HANDLE by KOpenInterface and Start Preview First
    KPlay(hK);

// [1] If you are handling raw data yourself by using call back function then you
//     have to filter the frames and decide which frame your are going to process.
//     This is because KMpeg4 will pass all the frames to call back function.

// Determine the frame type I or P frame.

    If (!bDecodeI )
    {
        // Decode All of Frames you receive
    } else {
        // Check the frame type
        // Decode I Frame Only
    }

//------------------------------------------------------------------------------

// [2] If KMpeg4 is handling the raw data for you then you can call
//     KSetDecodeIFrameOnly(HANDLE, BOOL) to decode I frame only
    KSetDecodeIFrameOnly(hK, true);
```

Draw your own information on the preview window

This chapter describes a mechanism on how you can draw your own information on the preview window, including OSD information, timecode or video intelligence information.

Steps to draw your own information on the preview window:

1. Register to the IP device
2. Setup after render callback function (`KSetAfterRenderCallback()`)
3. When preview window is painted, SDK will calls after render callback function
4. Draw your own information in the after render callback function

⚠ **NOTE:** When you hook up `KSetAfterRenderCallback()` function, the callback function will be called 30 times per second, if the frame rate is set to 30 FPS.

```
/
// register to the device

// Setup after render callback function
    KSetAfterRenderCallback( hK, dwCallbackID, AfterRenderCallback );


AfterRenderCallback(DWORD dwCallbackID)
{
//---- draw your own information over here,
//     including OSD, time code or video intelligence information
}
```

Record Operations

Background record with multicast mode

Streaming Client Library is developed for MPEG-4 Video Network Streaming Application.

Steps to start preview with multicast mode without preview include:

1. Set Contact type as `CONTACT_TYPE_MULTICAST_PREVIEW` or `CONTACT_TYPE_MULTICAST_WOC_PREVIEW`
2. Register to the IP devices
3. Start recording

NOTE: Application may start recording without preview.

```
/
// Get KMpeg4 handle
     HANDLE hK =  KOpenInterface();

// Prepare USER_INFO data structure by filling IP address, account, password.
     MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
     mcc.ContactType = CONTACT_TYPE_MULTICAST_PREVIEW;
     …
     KSetMediaConfig(hK, &mcc);
     KConnect(hK);

// Start Streaming
     KStartStreaming(hK);
// Start receiving data from KMpeg4
     KPlay(hK);

// Start recording with record file name.
     KStartRecord(hK, “c:\\rec.raw”);

// Finish recording
// You can retrive the recording information by passing MP4FILE_RECORD_INFO
     MP4FILE_RECORD_INFO mri;
     KStopRecord(hK, &mri);
```

Alarm Recording with DI event

Steps to start alarm recording include:

1. Setup pre-event recording time and post-event recording time
2. Register to the IP devices
3. Setup event callback
4. Start alarm recording
5. Stop alarm recording

```
// Setup digital input callback function
    KSetDICallback( hK, dwCallbackID, DIOCallback );

    KSetPrerecordTime(hK, 5);      // set pre-event time as 5 seconds

// Register to the device

//----- in call back function

DIOCallback(DWORD dwCallbackID, bool bDI1, bool bDI2 )
{
    if ( bDI1 || bDI2)         //---- DI 1 or DI 2 is on
    {
        KStartRecord (hK, "C:\\AlarmREC.raw" );
        Sleep( 10000 );            // records for 10 seconds
        KStopRecord( hK, NULL );   // in total it records 15 seconds
    }
}
```

Playback Operations

Steps to operate playback functions include:

1. Call `KOpenInterface()` to get KMpeg4 handle.
2. Prepare file name and set contact type to `CONTACT_TYPE_PLAYBACK`
3. Call `KSetMediaConfig(HANDLE, MEDIA_CONNECTION_CONFIG)` to set connect config.
4. Call `KConnect(HANDLE).`
5. Call `KStartStreaming(HANDLE)` to get ready to receive.
6. Call `KPlay(HANDLE)` to start receive data.
7. Sets playback play speed
8. Calls playback operation, including play forward, play backward, seed operation

Open and close a raw data file

```
// Get KMpeg4 SDK handle
     HANDLE hK =  KOpenInterface();

// Prepare playback file name.
     MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
     mcc.ContactType = CONTACT_TYPE_PLAYBACK;
     strcpy(mcc.PlayFileName, “c:\\test.raw”);
     …
     KSetMediaConfig(hK, &mcc);
// Open file.
     KConnect(hK);
// Start Streaming
     KStartStreaming(hK);


// Stop streaming
     KStopStream( hK );
// Close file
     KDisconnect( hK );
```

Play forward, backward

```
// Get KMpeg4 SDK handle
     HANDLE hK =  KOpenInterface();

// Set render information.
     MEDIA_RENDER_INFO mri;
     mri.RenderInterface = DGDI;
     mri.hWnd = m_hWnd;                  // Windows’ handle to draw
     mri.rec = m_rec;                    // rec information.
     KSetRenderInfo(hK, &mri);

// Prepare playback file name.
     MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
     mcc.ContactType = CONTACT_TYPE_PLAYBACK;
     strcpy(mcc.PlayFileName, “c:\\test.raw”);
     …
     KSetMediaConfig(hK, &mcc);
// Open file.
     KConnect( hK );
// Start Streaming
     KStartStreaming(hK);
// Play forward
     KPlay( hK );

// Play backward
     KSetPlayDirection(hK, false);
```

Play frame by frame

```
// Play step by step

// Open file and play
   ...

// need to set play status pause for play step frame
     KPause(hK);

// Step to next frame
    KStepNextFrame(hK);

// Step to previous frame
    KStepPrevFrame(hK);
```

4 Event Handling

Digital I/O Architecture

This material covers SDK architecture, data structure and sample programs to illustrate the methods to integrate ACTi's IP Surveillance products.

Receives Digital Input Event

Steps to receive digital input event include:

1. Register to the IP devices
2. Setup digital event callback
3. Process digital input event in the callback function

```
// Get Mpeg4 SDK handle
// Setup digital input callback function
    KSetDICallback( hK, dwCallbackID, DICallback );

    KSetPrerecordTime(hK, 5);      // set pre-event time as 5 seconds

// Register to the device

//----- in call back function

DICallback( DWORD dwCallbackID, bool bDI1, bool bDI2 )
{
    if ( bDI1 || bDI2 )        //---- DI 1 or DI 2 is on
    {
        KStartRecord (hK, "C:\\AlarmREC.raw" );
        Sleep( 10000 );             // records for 10 seconds
        KStopRecord( hK, NULL );   // in total it records 15 seconds
    }
}
```

Send Digital Output

Steps to receive digital input event include:

1. Register to the IP devices
2. Call `KSendDO(HANDLE, BYTE)` function to send event to the digital output device

Send DO 1

```
// Register to device.

#define DO_OUTPUT_1    0X01
#define DO_OUTPUT_2    0X02

// Send DO 1
     KSendDO( hK, DO_OUTPUT_1);
```

Send DO 2

```
// Register to device.

// Send DO 1
     KSendDO( hK, DO_OUTPUT_2);
```

Motion Detection Event Handling

Sets Motion Detection parameters

Steps to setup motion detection parameters include:

1. Register to the IP devices
2. Setup motion detection callback function
3. Sets motion detection parameters
4. Process motion detection event in the callback function

⚠ **NOTE:** The parameter to set the range of the motion detection window has to be the multiplier of 16, if not, the number will be align to the multiplier of 16. For example, if the application set the range as 125, then it will be align to 128.

Set MD Range to Range1

```
typedef struct structural_MEDIA_MOTION_INFO
{
DWORD dwEnable;
DWORD dwRangeCount;
DWORD dwRange[3][4];
DWORD dwSensitive[3]; ///< 0 - 100
} MEDIA_MOTION_INFO;

// Register to the IP devices

// Prepare you own callback function

// Plug function after KOpenInterface()
    KSetMotionDetectionCallback(hK, dwCallbackID, MDCallBack);

// Set motion detection structure
    MEDIA_MOTION_INFO mmi;
    mmi.dwEnable = 1;                   // Enable MD
    mmri.dwRangeCount = 1;              // Just 1 range for MD
    mmi.dwSensitive[0] = 100;           // Sensitive of range 1
    mmi.dwRange[0][0] = 0;              // Left position
    mmi.dwRange[0][1] = 0;              // Top position
    mmi.dwRange[0][2] = 128;            // Width of range 1
    mmi.dwRange[0][3] = 128;            // Height of range 1

// Set motion detection information.
    KSetMotionInfo( hK, mmi);
```

Gets Motion Detection Settings

Get MD Range Setting

```
//Prepare structure for get MD information
    MEDIA_MOTION_INFO mmi;

// One function to get all data
    KGetMotionInfo(hK, &mmi);
```

Receives Motion Detection Trigger Event

To Plug You Own Callback Function for MD

```
Void MDCallBack(DWORD dwCallbackID, bool bMotion1, bool bMotion2, bool bMotion3)
{
    if( bMotion1 )
    {
// Motion 1 Event occuring
    }

    if( bMotion2 )
    {
// Motion 2 Event occuring
    }

    if( bMotion3 )
    {
// Motion 3 Event occuring
    }
}
```

Status Callback – video lost, recovery, disconnect event

Status callback includes:

1. Video Lost event
2. Video Recorvery event
3. Network disconnect event

Steps to implement status callback are listed as follow:

1. Register to the device
2. Setup appropriate callback function (`KSetVideoLossCallback()`, `KSetVideoRecoveryCallback()`, `KSetNetworkLossCallback()`)
3. Event handling in the status callback function

```
//---- prepare status callback here
// Video lost
void VideoLossCallBack(DWORD dwCallbackID)
{
// To Do: Add your video loss handle code here.
}

// Video recovery
void VideoRecoveryCallBack(DWORD dwCallbackID)
{
// To Do: Add your video recovery handle code here.
}

// Disconnect
void NetworkLossCallback(DWORD dwCallbackID)
{
// To Do: Add your network loss handle code here.
}



//---- register to the server
// Set video loss call back
    KSetVideoLossCallback( hK, dwCallbackID, VideoLossCallBack);
```

```
// Set video recovery call back
    KSetVideoRecoveryCallback( hK, dwCallbackID, VideoRecoveryCallBack);

// Set network loss (disconnect) call back
    KSetNetworkLossCallback(hK, dwCallbackID, NetworkLossCallback);
```

5 PTZ Integration

PTZ Integration Architecture

This material covers how to integrate PTZ protocol with prepared information.

In the product architecture, the PTZ operation is defined as transparent tunnel; in this way, the PTZ protocol information does not keep in the firmware, and user's application has to parse and prepare PTZ commands in the application side.

To shorten the integration process, SDK provides implemented and tested PTZ protocol files, so that application may just utilize the PTZ protocols that has been prepared.

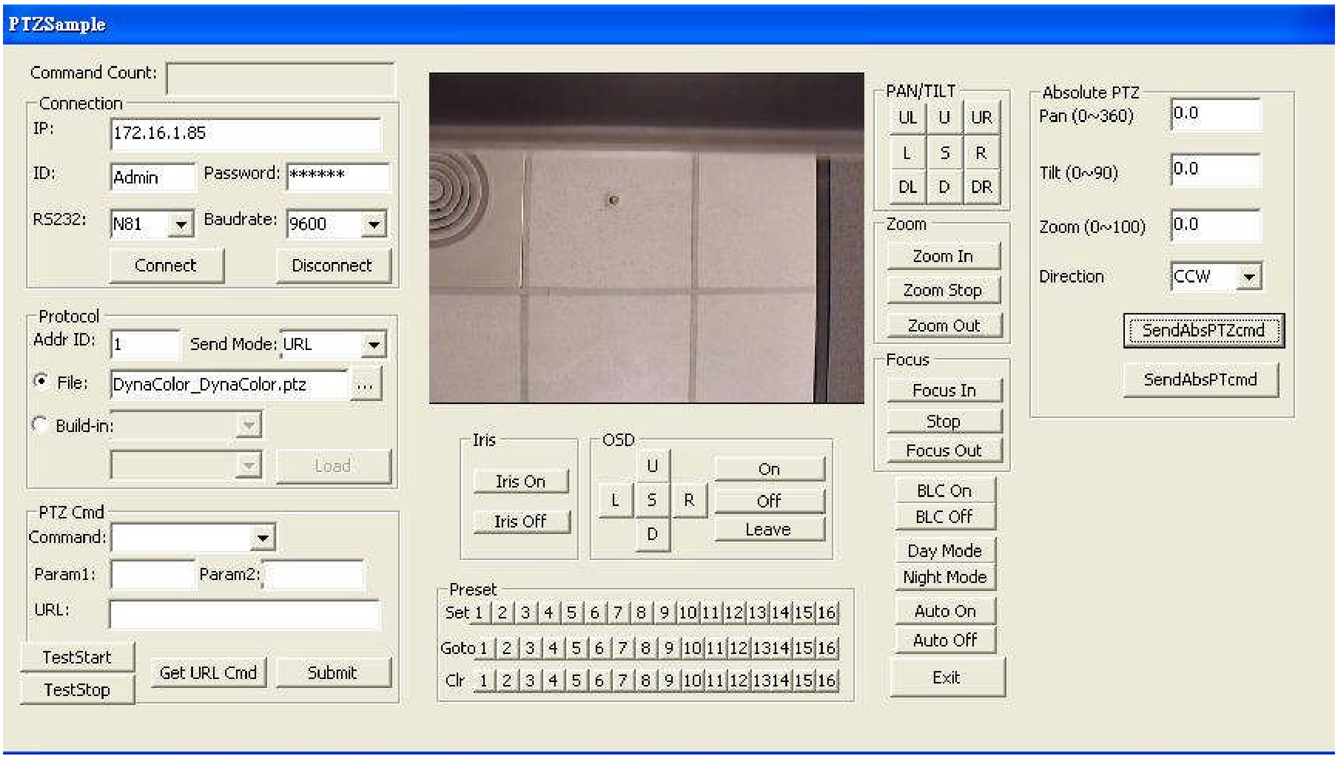
NOTE: Firmware does not contain PTZ protocol information. User's application has to prepare the PTZ command string and execute the string directly

The benefits of the PTZ Integration architecture are listed as follow:

- Utilize tested protocols
- Provides PTZ operation command strings
- Provides important commands like Day and Night switch, Patrol, Pattern, IR, etc
- Provides OSD operation

PTZ Parser Source Code

Please refer to `${SDK-DIR}\SDK\PTZSample` for sample source code. Also, ACTi provides integrated PTZ protocol files under `${SDK-DIR}\PTZ-Protocol`.



PTZSample

Command Count:

Connection
IP: 172.16.1.85
ID: Admin
Password: ******
RS232: N81
Baudrate: 9600

Protocol
Addr ID: 1
Send Mode: URL
● File: DynaColor_DynaColor.ptz
○ Build-in:

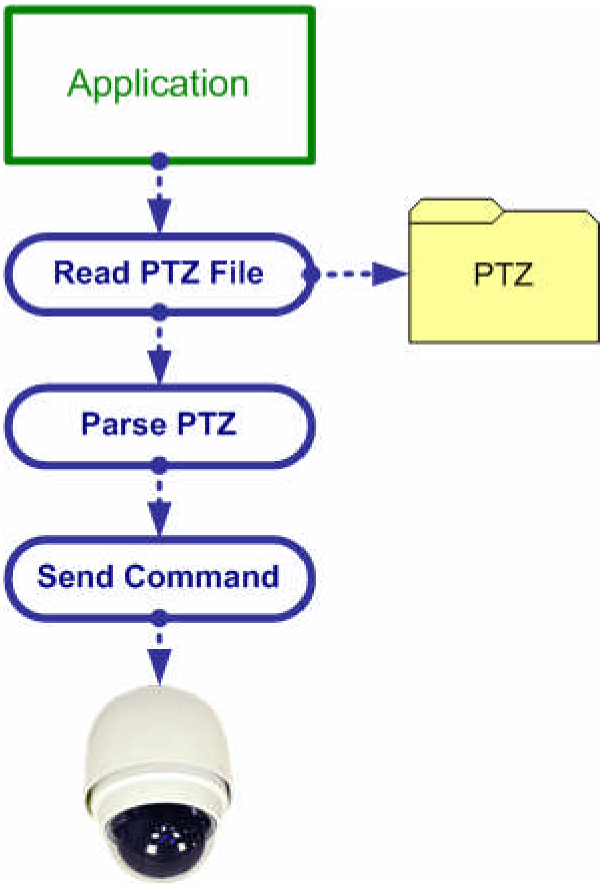
PTZ Cmd
Command:
Param1:
Param2:
URL:

Absolute PTZ
Pan (0~360) 0.0
Tilt (0~90) 0.0
Zoom (0~100) 0.0
Direction CCW

Steps to integrate a PTZ protocol include:

1. **Read PTZ File**: read PTZ protocol file specified
2. **Parse PTZ command**: parse the PTZ command rules, calculate the checksum and prepare the PTZ command
3. **Send Command**: sends PTZ command out with URL command or `netSend2ServerSerialPort()` function

(Most of new PTZ APIs in SDK 10000 V1.2 proceed step 1 and 2 at the same time)



PTZ Protocol Files ${SDK-DIR}\PTZ-Protocol

This section describes the definition of PTZ protocol files. Please get these files from `${SDK-DIR}\PTZ-Protocol\` directory. A sample fragment of the protocol file looks like follow

```
ADDRIDSTART; 1; 0;;;;
ADDRIDPOS; 2; 0;;;;
CHECKSUM; $B7=$B2+$B3+$B4+$B5+$B6;;;
INTERVAL;0;0;;;;
PANTILT;-5;-5;0xFF,0x01,0x00,0x14,0x3F,0x3F,0x93;;;
OSDON;0;0;0xFF,0x01,0x00,0x03,0x00,0x5F,0x63;;;
OSDUP;0;0;0xFF,0x01,0x00,0x08,0x00,0x0C,0x15;;;
OSDENTER;0;0;0xFF,0x01,0x02,0x00,0x00,0x00,0x03;;;
```

The protocol file contains following commands:

1. `ADDRIDSTART`: indicates the starting number of the address ID. Take above sample as an example (`ADDRIDSTART;1;0;;;;`), if the application is set to address ID as 3, then it starts at 1, so the calculated address ID is 3 (`0x03`);

2. `ADDRIDPOS`: indicates the position to replace with calculated address ID. Take above sample as an example (`ADDRIDPOS; 2; 0;;;;`), the address ID is at 2nd position of the command string. So, `PANTILT; -5, -5` command (`PANTILT;-5;-5;0xFF,0x01,0x00,0x14,0x3F,0x3F,0x93;;;`) will be replace as (`PANTILT;-5;-5;0xFF,0x03,0x00,0x14,0x3F,0x3F,0x93;;;`)

3. `CHECKSUM`: indicates the checksum rule, `+` is to run `AND` operation, `|` is to run `OR` operation, `^` is to run `XOR` operation. Take above sample as an example (`CHECKSUM; $B7=$B2+$B3+$B4+$B5+$B6;;;`), the checksum rule is to run `AND` operation for byte 2, byte 3, byte 4, byte 5 and byte 6, and the result is placed at byte 7. Then this becomes a final `PTZ command string`

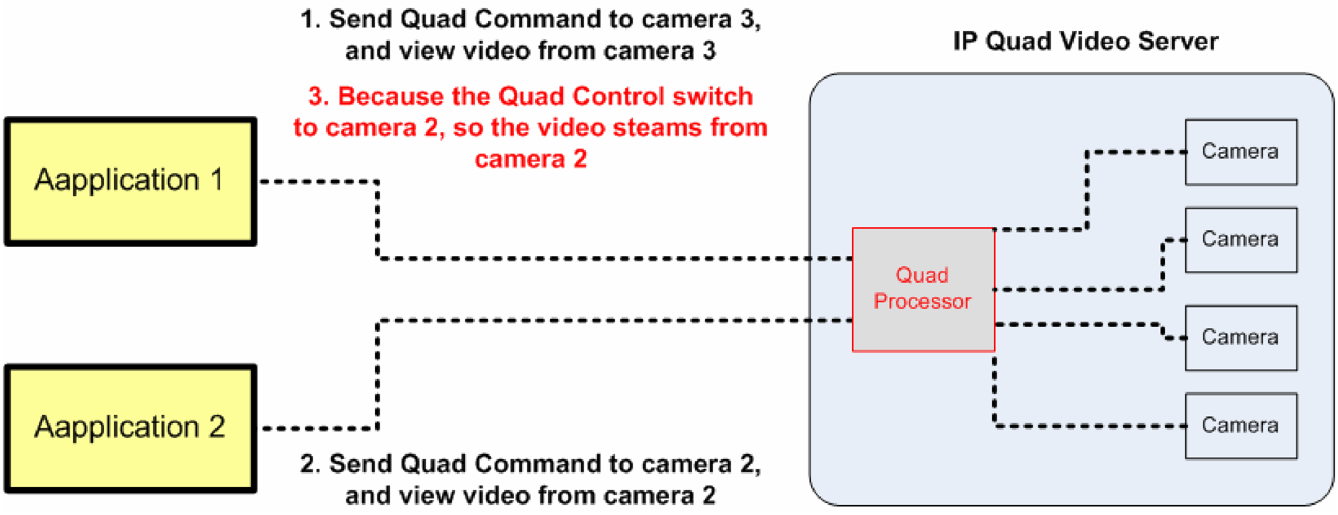
4. Application then sends the calculated `PTZ command string` out via normal serial port operation or URL command.

6 IP Quad Video Server Integration

IP Quad Architecture

IP Quad is a Quad processor which connects to 4 analog video sources then multiplexed by a quad processor; in this way, an IP Quad video server may generates 1 Full D1 video stream or 4 CIF video streams at the same time

IP Quad video server firmware contains URL commands, so that application may simply sends out the URL command to control the behavior of it.


1. Send Quad Command to camera 3, and view video from camera 3
3. Because the Quad Control switch to camera 2, so the video steams from camera 2
IP Quad Video Server
Aapplication 1
Aapplication 2
Quad Processor
Camera
Camera
Camera
Camera
2. Send Quad Command to camera 2, and view video from camera 2

NOTE: There is only one quad processor in the device, so when an application sends a URL command to the IP Quad video server, then the quad processor will execute the commands specified, and all connected application will receive the same result from quad processor.

IP Quad URL Commands

Application may just use URL Command to perform these tasks to setup and control Quad Video Server; for information that needs to retrieve from Quad Video Server (e.g. Retrieve MPEG-4 stream, record to files, motion detection event, digital input event), the calling methods are all the same as SDK-2000 v1.0.

IP Quad's quad control is based on URL Command, which means that you need to send out the URL Command to IP Quad to set certain parameters.

HTTP Code Status

HTTP Code	HTTP Text	Description
200	OK	The request has succeeded, but an application error can still occur, which will be returned as an application error code.
204	No Content	The server has fulfilled the request, but there is no new information to send back.
400	Bad Request	The request had bad syntax or was inherently impossible to be satisfied.
401	Unauthorized	The request requires user authentication or the authorization has been refused.
404	Not Found	The server has not found anything matching the request.
409	Conflict	The request could not be completed due to a conflict with the current state of the resource.
500	Internal Error	The server encountered an unexpected condition which prevented it from fulfilling the request.
503	Service Unavailable	The server is unable to handle the request due to temporary overload.

Example :

Return success http context

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
 \n
```

Return failed http context

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
 \n
ERROR: error description
```

How to set display mode

Syntax	`http://192.168.1.1/cgi-bin/quad?DISPLAY=n`

How to get display mode

Syntax	http://192.168.1.1/cgi-bin/quad?DISPLAY

<parameter>	**<values>**	**Description**
DISPAY	n: 0~4	0: quad display 1: display channel 1 2: display channel 2 3: display channel 3 4: display channel 4

How to set osd enabled

Syntax	http://192.168.1.1/cgi-bin/quad?OSD_ENABLED=0xnn

How to get osd enabled status

Syntax	http://192.168.1.1/cgi-bin/quad?OSD_ENABLED

<parameter>	**<values>**	**Description**
OSD_ENABLED	0xnn : hexadecimal	BIT0: 1:title name enabled BIT1: 1:video loss enabled BIT2: 1:motion detect enabled BIT3: 1:date time enabled BIT4: 1:DIO status enabled BIT5: Reserved BIT6: Reserved BIT7: Reserved

How to set motion detect enabled

Syntax	http://192.168.1.1/cgi-bin/quad?MOTION_ENABLED=0xnn

How to get motion enabled status

Syntax	http://192.168.1.1/cgi-bin/quad?MOTION_ENABLED

<parameter>	**<values>**	**Description**
MOTION_ENABLED	0xnn : hexadecimal	BIT0: 1:channel 1 motion detect enabled BIT1: 1:channel 2 motion detect enabled BIT2: 1:channel 3 motion detect enabled BIT3: 1:channel 4 motion detect enabled BIT4: Reserved BIT5: Reserved BIT6: Reserved BIT7: Reserved

How to set sensitive for motion detect

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&SENSITIVE=m

How to get sensitive setting

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&SENSITIVE

<parameter>	**<values>**	**Description**
CHANNEL	n: 1~4	channel number
SENSITIVE	m: 0~15	0: more sensitive . .. 8: middle sensitive . .. 15: less sensitive

7 Advanced Topics

Callback Functions

This section lists the callback functions and its explanation for references.

Category	Function	Description
Decode	`KSetImageCallback()`	Callback functions to receive RGB buffer.
Event	`KSetDICallback()`	DI event triggers
Event	`KSetMotionDetectionCallback()`	Motion detection event triggers
MPEG-4	`KSetRawDataCallback()`	MPEG-4 raw data including Video and Audio. All data are in TCP v2.0 format.
MPEG-4	`KSetTimeCodeCallback()`	Timecode is sent to this callback function every time a frame arrives
Preview	`KSetAfterRenderCallback()`	Callback functions are called every time a frame is drawn on the screen. This is useful when user wants to draw their own OSD, Text or video intelligence information overlay on the preview window
Preview	`KSetResolutionChangeCallback()`	Callback function is called when resolution is changed.
RS-232	`KSetRS232DataCallback()`	RS-232/RS-422/RS-485 data arrives
System	`KSetVideoLossCallback()`	Video loss event triggers.
System	`KSetVideoRecoveryCallback()`	Video recovery event triggers.
System	`KSetNetworkLossCallback()`	Network loss is sent if disconnect.

Deals with MPEG-4 Stream

This section describes the ways to deal with MPEG-4 stream, including:

- MPEG-4 raw data callback (Video and Audio)
- How to detects I-Frame
- Decode I-Frame only

MPEG-4 Raw Data Format in TCP 2.0

Please refer your request to our Sales representative for detailed protocol and MPEG-4 data format specification.

MPEG-4 stream raw data format (video and audio) is described as follow:

Video Data: **I-Frame Data Structure** or **P-Frame Data Structure**

Audio Data: **Audio frame**

Get MPEG-4 Raw Data (Video + Audio)

Steps to get MPEG-4 raw data include:

1. Register to the IP devices
2. Setup MPEG-4 callback function

```
//---- prepare callback function when MPEG-4 raw data arrives

#define DATA_TYPE_VIDEO    0x01
#define DATA_TYPE_AUDIO    0x02

void RawDataCallBack(DWORD id, DWORD dwDataType, BYTE* buf, DWORD len )
{
    switch (dwDataType)
    {
      Case DATA_TYPE_VIDEO:
//do something for video stream
      break;
      Case DATA_TYPE_AUDIO:
//do something for audio stream
      break:
    }
}
// Prepare yourself callback function first


//---- register server
    HANDLE hK =  KOpenInterface();
// you should get HANDLE by KOpenInterface before Preview

// Set call back functions
    KSetRawDataCallback(hK, id, RawDataCallBack);

// Prepare USER_INFO data structure by filling IP address, account, password.
    MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
    mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
    strcpy(mcc.UserID, “Your ID”);
    strcpy(mcc.Password, “Your Password”);
    mcc.RegisterPort = 6000;
    mcc.ControlPort = 6001;
    mcc.StreamingPort = 6002;
    mcc.MultiCastPort = 5000;
    mcc.HTTPPort = 80;
    strcpy(mcc.UniCastIP, “172.16.1.81”);
    strcpy(mcc.MultiCastIP, “225.5.6.81);

// Set media configuration file.
    KSetMediaConfig(hK, &mcc);
// Register
    KConnect(hK);
// Start Streaming
    KStartStreaming(hK);
// Start receiving data from KMpeg4
    KPlay(hK);



//---- below list some step if you need terminate whole process
    KStop(hK);
    KStopStreaming(hK);
    KDisconnect(hK);
    KCloseInterface(hK);
```

Detect I-Frame (key frame)

Steps to detect I-Frame in MPEG-4 raw data include:

1. Process in MPEG-4 raw data callback function
2. Check the MPEG-4 raw data format

Video data structure:

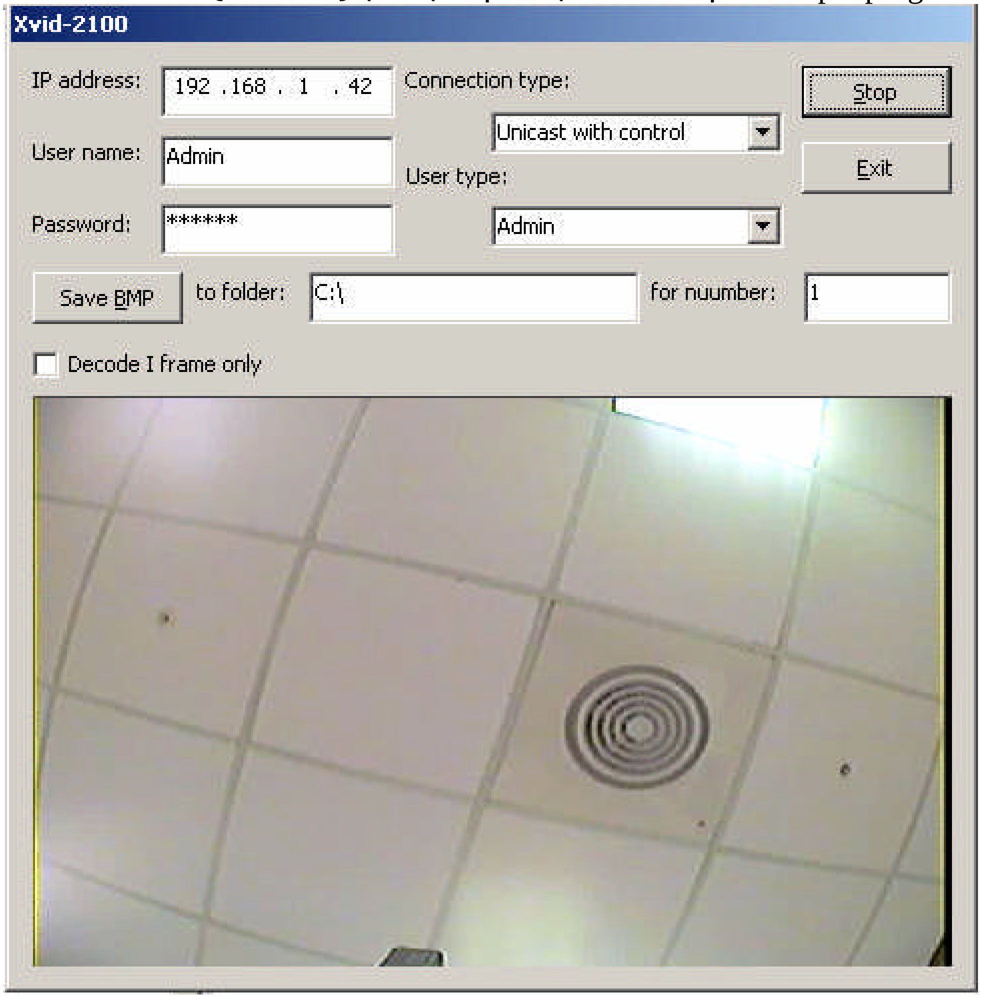
I Frame = **User Data** + **Bitstream Data** + **I-Frame Data**

```
// in C++ language example, here shows to know an I-Frame
// We suppose BYTE* buf is a continuous raw data for one frame
// compare 0xB3010000 with 4 bytes from the 75th byte in BYTE* buf
     DWORD f;
     CopyMemory( (BYTE*)&f, (buf+75), sizeof( DWORD ) );

     // an I-Frame
     if( f == 0xB3010000 )
     {
     }
     else ; //---- P-Frame
```

Decode MPEG-4 Stream with Xvid

ACTi MPEG-4 stream complies with standard ISO-14496-2 format and can be decoded with open source MPEG-4 software decoders, including FFMPEG, Xvid, DivX, etc.

Please refer to `${SDK-DIR}\SDK\Samples\DecodeSample` sample program.



IP address: 192 .168 . 1 . 42
Connection type: Unicast with control
User name: Admin
User type: Admin
Password: ******
to folder: C:\
for nuumber: 1
□ Decode I frame only

Steps to use netSetMpeg4RawDataCallBack and decode by XVID:

1. Link xvidcore.dll.a as Import Lib

2. Put xvidcore.dll in the same directory

3. Include xvid.h

4. Provide following initialize, create, decode, close xvid code.

```
#include "xvid.h"

DWORD m_vWidth;
char  pOutBuf[720*576*3];
```

```
xvid_dec_create_t   m_xvidDecHandle;
xvid_gbl_init_t     xvid_gbl_init;
int                xvidret;

//----------------------------------------------------------------------------
// XVID Decord Init and Create ==>

        memset(&xvid_gbl_init, 0, sizeof(xvid_gbl_init));
        memset(&m_xvidDecHandle, 0, sizeof(m_xvidDecHandle));
        m_xvidDecHandle.version = XVID_VERSION;
        m_xvidDecHandle.height = 0;
        m_xvidDecHandle.width = 0;
        xvid_gbl_init.version = XVID_VERSION;
        xvidret = xvid_global(0, XVID_GBL_INIT, &xvid_gbl_init, 0);
        xvidret = xvid_decore(NULL, XVID_DEC_CREATE, &m_xvidDecHandle, NULL);

//----------------------------------------------------------------------------
//XVID Decord ==>  Put the code into the netSetMpeg4RawCallBack 's CallBack Function

        xvidDecFrame.output.csp = XVID_CSP_BGR;
        xvidDecFrame.general = XVID_LOWDELAY|XVID_DEBLOCKY|XVID_DEBLOCKUV;
        xvidDecFrame.general = XVID_LOWDELAY;
        xvidDecFrame.version = XVID_VERSION;
        xvidDecFrame.output.plane[0] = pOutBuf;               // <<<<<<<

//----------------------------------------------------------------------------
// Output Buffer for the Decord out put

        // <<<<<<< The Video's Width Size => m_vWidth * 3, (a Pixel is 3 Bytes (RGB))
        // <<<<<<< The m_vWidth can get from the Mpeg4 Raw Data
        // <<<<<<< (In the input buffer that first time the callback be called)
        // <<<<<<< Or can assign by yourself if you know what is the video's width
        xvidDecFrame.output.stride[0] = m_vWidth * 3;

        xvidDecFrame.bitstream = pInBuf;     // <<<<<<< The Mpeg4 Raw Data
        xvidDecFrame.length = Len;           // <<<<<<< Mpeg4 Raw Data's Length

        xvidret    =    xvid_decore(m_xvidDecHandle.handle,    XVID_DEC_DECODE,
&xvidDecFrame, 0);
        // Todo : pOutBuf -> Display

//----------------------------------------------------------------------------
// XVID Decord Close ==>

xvidret = xvid_decore(m_xvidDecHandle.handle, XVID_DEC_DESTROY, 0, 0);
```

Get RGB Image Data

Get RGB Image Data with Image Callback Function

Steps to get RGB image data with image callback function:

1. Register to the IP devices
2. Initialize stream
3. Start stream
4. Setup image callback function

```
//---- prepare image callback function

Void ImageCallBack(DWORD id, BYTE* pBuf, DWORD len, long w, long h)
{
// list sample below for save BMP file to “save.bmp” after get RGB data
      LPBITMAPINFO lpbih = (LPBITMAPINFO)pBuf ;
      Long lImageLen=(lpbih->bmiHeader).biSize \ (lpbih->bmiHeader).biSizeImage ;

      BITMAPFILEHEADER oHeader ;
      oHeader.bfType = 0x4d42 ;
      oHeader.bfReserved1 = 0;
      oHeader.bfReserved2 = 0;
      oHeader.bfSize      =      (DWORD)(sizeof(BITMAPFILEHEADER)     \      +
(lpbih->bmiHeader).biSize + (lpbih->bmiHeader).biSizeImage) ;
     oHeader.bfOffBits       =       (DWORD)(sizeof(BITMAPFILEHEADER)       +
(lpbih->bmiHeader).biSize) ;

     CFile oImage ;
     oImage.Open(“save.bmp”, CFile::modeCreate | CFile::modeWrite ) ;
     oImage.Write( &oHeader, sizeof(BITMAPFILEHEADER) );
     oImage.Write( pBuf, (lpbih->bmiHeader).biSize ) ;

     for(int i = lpbih->bmiHeader.biHeight-1 ; i >= 0 ; i--)
          oImage.Write(            (pBuf+(lpbih->bmiHeader).biSize           +
(i*lpbih->bmiHeader.biWidth*4)), lpbih->bmiHeader.biWidth*4) ;

     oImage.Close();
}


     HANDLE hK =  KOpenInterface();
```

```
// you should get HANDLE by KOpenInterface before Preview

// Set call back functions
     KSetRawDataCallback(hK, id, RawDataCallBack);

// Set Display Informationm
     MEDIA_RENDER_INFO mri;
     mri.RenderInterface = DGDI;
     mri.hWnd = m_hWnd;              // Windows’ handle to draw
     mri.rec = m_rec;                // rec information.
     KSetRenderInfo(hK, &mri);

// Prepare USER_INFO data structure by filling IP address, account, password.
     MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
     mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
     strcpy(mcc.UserID, “Your ID”);
     strcpy(mcc.Password, “Your Password”);
     mcc.RegisterPort = 6000;
     mcc.ControlPort = 6001;
     mcc.StreamingPort = 6002;
     mcc.MultiCastPort = 5000;
     mcc.HTTPPort = 80;
     strcpy(mcc.UniCastIP, “172.16.1.81”);
     strcpy(mcc.MultiCastIP, “225.5.6.81);

// Set media configuration file.
     KSetMediaConfig(hK, &mcc);
// Register
     KConnect(hK);
// Start Streaming
     KStartStreaming(hK);
// Start receiving data from KMpeg4
     KPlay(hK);

// Plug Image callback for get RGB Image
     KSetImageCallback(hK, dwCallBackID, ImageCallBack);
.. .. ..
.. .. ..


// below list some step if you need terminate whole process
     Stop(hK);
     KStopStreaming(hK);
     KDisconnect(hK);
     KCloseInterface(hK);
```

Save RGB Data into a BMP file

We can get raw data and save to other file format e.g. if we want to save the current frame to Bitmap file for website image index. Just as like as general computer file format the Bitmap file has itself format. The Bitmap file format has a **BITMAPFILEHEADER**, **BITMAPINFORHEAR** and bitmap bits. Luckily we just have to prepare the header because the bitmap bits that we can get from MPEG-4 Callback function. Below is the 24 bit Bitmap file example.

Steps to save RGB data into a BMP file include:

1. Register to the IP devices
2. Initialize stream
3. Start stream
4. Setup image callback function
5. Create **BITMAPFILEHEADER** data structure and write to file

⚠ **NOTE:** Please refer to **/SDK/Samples/DecodeSample** sample program for full source codes.

F

First we have to create the **BITMAPFILEHEADER** struct and write to file.

```
// Save 24bit BMP
long BufferSize = 720*480*3;
// Write out the file header
//
BITMAPFILEHEADER bfh;
memset( &bfh, 0, sizeof( bfh ) );
bfh.bfType = 'MB';
bfh.bfSize = sizeof( bfh ) + BufferSize + sizeof( BITMAPINFOHEADER );
bfh.bfOffBits = sizeof( BITMAPINFOHEADER ) + sizeof( BITMAPFILEHEADER );

DWORD Written = 0;
WriteFile( hf, &bfh, sizeof( bfh ), &Written, NULL );
```

Second we have to create the **BITMAPINFOHEADER** struct and write to file.

```
// Write the bitmap format
//
BITMAPINFOHEADER bih;
memset( &bih, 0, sizeof( bih ) );
bih.biSize = sizeof( bih );
bih.biWidth = 720;
bih.biHeight = -480;  //Save from down to up
bih.biPlanes = 1;
bih.biBitCount = 24;
Written = 0;
WriteFile( hf, &bih, sizeof( bih ), &Written, NULL );
```

Finally we only need to write the bitmap bits to file and close it.

```
// Write the bitmap bits
//
Written = 0;
WriteFile( hf, xvidDecFrame.output.plane[0], BufferSize, &Written, NULL );

// Close BMP file
CloseHandle( hf );
```

Save Recording to an AVI file

Steps to save recording data into a AVI file include:

1. Register to the IP devices
2. Sets MPEG-4 raw data callback
3. Sets FourCC type as “**vids**”
4. Sets FourCC handle as “**DX50**”
5. Calls AVI functions when receiving frames

NOTE: Please refer to **MSDN** sample or Microsoft web site for reference.

```
A
VIFileInit(); // initializes the AVIFile library


strcpy((char*)g_aviname, m_NormalSaveFile);  // file name
g_aviframesize = (m_width * m_height * 3) / 2;

   // Is the file exist?
FILE *fp = fopen(m_NormalSaveFile, "rb");  if (fp)  {
   fclose(fp);
   DeleteFile(m_NormalSaveFile); // delete it.
}

AVISTREAMINFO g_strhdr_out;
BITMAPINFO g_header;
  // clear the struct
memset(&g_strhdr_out, 0, sizeof(g_strhdr_out));

g_strhdr_out.fccType                 = mmioFOURCC('v', 'i', 'd', 's');// stream type
g_strhdr_out.fccHandler               = mmioFOURCC('D', 'X', '5', '0');
g_strhdr_out.dwScale                 = 1001;
g_strhdr_out.dwRate                  = (DWORD)(m_theFps * 1001);
g_strhdr_out.dwSuggestedBufferSize  = g_aviframesize;

g_header.biSize = 40;
g_header.biWidth = m_width;
g_header.biHeight = m_height;
g_header.biPlanes = 1;
g_header.biBitCount = 0;
g_header.biCompression = g_strhdr_out.fccHandler;
g_header.biSizeImage = g_aviframesize * 2;
g_header.biXPelsPerMeter = 0;
g_header.biYPelsPerMeter = 0;
g_header.biClrUsed =0;
g_header.biClrImportant =0;


   // Create a AVI file.
hr = AVIFileOpen(&m_pAviFile, (char*)g_aviname, OF_WRITE | OF_CREATE, NULL);
if (hr != AVIERR_OK) {
   AVIFileExit();
   return -1;
}

// Create a interface to the new stream.
hr = AVIFileCreateStream(m_pAviFile, &m_pAviVideo, &g_strhdr_out);
if (hr != AVIERR_OK) {
   AVIFileExit();
   return -1;
}
   // sets the format of a stream at the specified position
hr = AVIStreamSetFormat(m_pAviVideo, 0, &g_header, sizeof(g_header));
if (hr != AVIERR_OK) {
   AVIFileExit();
   return -1;
}

m_AviFrameNo = 0;

if (IFrame)
   m_AviFlag = AVIIF_KEYFRAME; // I frame
else
   m_AviFlag = 0;

// write data to stream
hr = AVIStreamWrite(m_pAviVideo, m_AviFrameNo++, 1,
   (LPBYTE) (m_PreSaveFrame[j]),
   m_PreSaveFrameLen[j], m_AviFlag,
   NULL, NULL);

if (hr != AVIERR_OK) {
   return -1; // Record AVIStreamWrite Error6
}

   // Release the Stream
AVIStreamRelease(m_pAviVideo);
```

```
   // Release the file
AVIFileRelease(m_pAviFile);

   // Release the AVIFile Libary
AVIFileExit();
```

Save Recording to an AVI file with SDK Function

Steps to save recording data into a AVI file include:

1. Connect to the IP devices
2. Sets File Writer Type to AVI
3. Start record

```
// Create a interface to the new stream.
HANDLE h = KopenInterface();
KSetMediaConfig( h , cfg );
...
KConnect( h );
KStartStreaming( h );
...

KSetFileWriterType( h, FAVI /* 1 */ );  // To record by AVI mode
                                        // FAVI is a defined macro
KStartRecord( h, “FileName.avi”);
```

Register Control Connection Only

Register to control connection only if you only want to receive events from video server but not video data (for example: motion, DI).   You can also send commands through control connection(for example: PTZ command, set motion…etc).

Steps to register with control connection only:

1. Call `KOpenInterface()` to get KMpeg4 handle.
2. Prepare IP address, port number, account, password, contact type..
3. Call `KSetMediaConfig(HANDLE, MEDIA_CONNECTION_CONFIG)` to set connect config.
4. Set Contact type to `CONTACT_TYPE_CONTROL_ONLY`.
5. Call `KConnect(HANDLE).`
6. Call `KStartStreaming(HANDLE)` to get ready to receive.
7. Call `KPlay(HANDLE)` to start receive.

```
// you should get HANDLE by KOpenInterface before Preview
    HANDLE hK = KOpenInterface();

// Set call back functions
    KSetRawDataCallback(hK, id, fnRawCallback);

// Prepare USER_INFO data structure by filling IP address, account, password.
    MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
    mcc.ContactType = CONTACT_TYPE_CONTROL_ONLY;
    …
    KSetMediaConfig(hK, &mcc);
    KConnect(hK);

// Start Streaming
    KStartStreaming(hK);

// Start Receive
    KPlay(hk);
```

Display text on screen

Steps to display text on screen while previewing.

1. Call `KOpenInterface()` to get KMpeg4 handle.
2. Prepare IP address, port number, account, password, contact type..
3. Call `KSetMediaConfig(HANDLE, MEDIA_CONNECTION_CONFIG)` to set connect config.
4. Set Contact type .
5. Call `KConnect(HANDLE).`
6. Call `KStartStreaming(HANDLE)` to get ready to receive.
7. Call `KPlay(HANDLE)` to start receive.
8. Call `KSetTextOut()` to diaply text.

```
// you should get HANDLE by KOpenInterface before Preview
     HANDLE hK = KOpenInterface();

// Set call back functions
     KSetRawDataCallback(hK, id, fnRawCallback);

// Prepare USER_INFO data structure by filling IP address, account, password.
     MEDIA_CONNECTION_CONFIG mcc;
// Set your connection information into struct mcc.
     mcc.ContactType = CONTACT_TYPE_UNICAST_PREVIEW;
     …
     KSetMediaConfig(hK, &mcc);
// Set render info
     MEDIA_RENDER_INFO mri;
     KSetRenderInfo(h, &mri);

     KConnect(hK);

// Start Streaming
     KStartStreaming(hK);

// Start Receive
     KPlay(hk);

// Display text
     KSetTextOut(h, 0, 0, 0, “123456789\0”, 9, true, false, false, “Arial”, 100,
     RGB(255, 255, 0), 2, RGB(0, 0, 255);
```

8 ACTi MPEG-4 Data Structure

Connection Type

Unicast Video and Control Connection

The section describes the mechanism on how to search ACTi's IP surveillance products on network. With this mechanism, you can locate the devices on the network, then use URL commands to operate or manage those devices.

The function sends out a broadcast message, ACTi's devices respond with detailed information, application then parse the replied information and parse the content with NET_SEARCHSERVER data structure.

Multicast Video + Control connection

The section describes the mechanism on how to search ACTi's IP surveillance products on network. With this mechanism, you can locate the devices on the network, then use URL commands to operate or manage those devices.

Multicast Video(Without Connection)

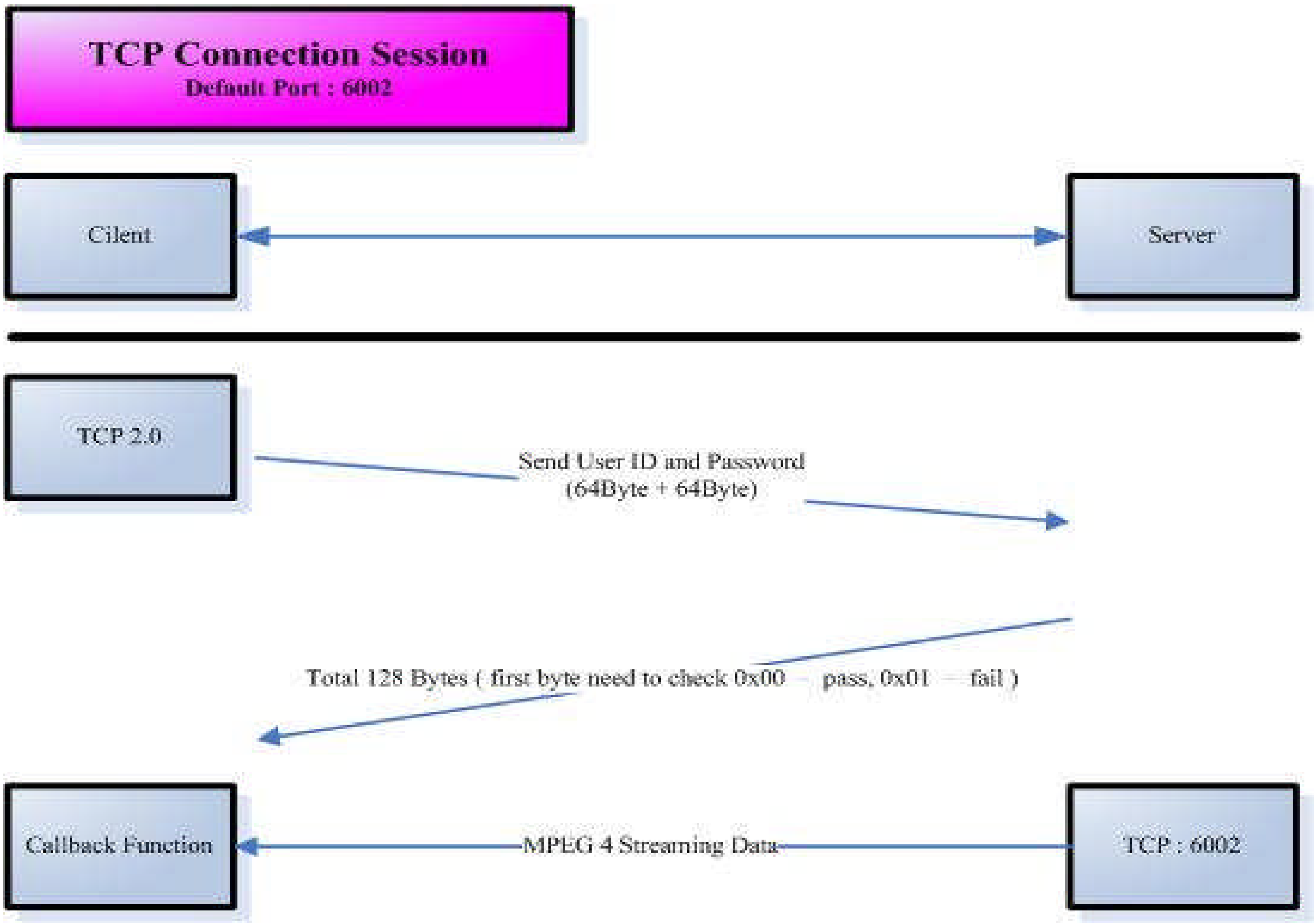
The section describes the mechanism on how to search ACTi's IP surveillance products on network. With this mechanism, you can locate the devices on the network, then use URL commands to operate or manage those devices.

Unicast Video and Control

Connect to Video Server

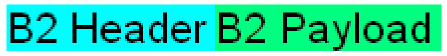
Here lists steps to build up the connection of getting audio/video streaming data.

1. Create a TCP socket connection that is needed to specific the IP and port. The default port is 6002.
2. Send a 128bytes command to video server. That includes User ID 64 bytes and Password 64 bytes.
3. Then we will get the response code. It are total 128 bytes code and includes a byte connect result.
4. Receive the data that will be audio/video streaming data.



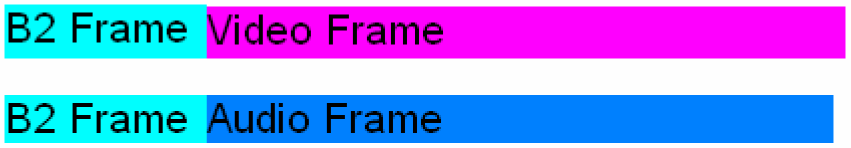
Definition of B2 Frame

The B2 Frame is composed of B2 Header and B2 Payload. The length of B2 Header is fixed to 12 bytes. The length of B2 payload is variable length depends on the B2 MsgType defined in the B2 Header.

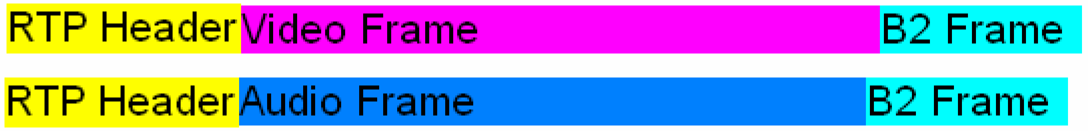


The location of the B2 frame in the streaming frame depends on the type of streaming protocols.

In ACTi TCP2.0 and Multicast over UDP streaming protocols, the B2 frame is located at the beginning of the audio/video frame.



In RTP streaming protocol, the B2 frame is located at the end of the audio/video frame.



Mpeg4 Video Data Format

Video and Audio Frame

After the connection is established, this section introduces how to get the streaming data from video Server.

We use the private data header(0x000001B2) to be the header tag. When we receive the data tag is the 0x000001B2 and the follow is the struct B2_HEADER. If the msg_type of the B2_HEADER is 1 and this frame is the video frame. Another 2 is the audio frame.

Video frame

Mpeg4 streaming data has two kind of video frame that is called I-Frame and P-Frame. There have some different. The I-Frame includes the sequence header that describe the information of decode(Bitstream data) like as below.

Header(0x000001B2) B2 FRAME	Bitstream Data (0x000001B0)	I-Frame data

The P-Frame is simple than I-Frame. It doesn’t include the sequence header.

Header(0x000001B2) B2 FRAME	P-Frame data

I-Frame Data Structure

B2 Header

```
typedef struct {
     B2_HEADER     header;
     PRIVATE_DATA   prdata;
} VIDEO_B2_FRAME;

#define ACTI_HEAD_MSG_B2_VIDEO_MPEG4              0x01
#define ACTI_HEAD_MSG_B2_AUDIO_8KPCM              0x02
#define ACTI_HEAD_MSG_B2_AUDIO_TIMESTAMP_8KPCM   0x03
#define ACTI_HEAD_MSG_B2_VIDEO_MJPEG              0x04
#define ACTI_HEAD_MSG_B2_VIDEO_H264               0x05

typedef struct {
     unsigned char acti[4];  /* 00 00 01 B2 */
     unsigned char msg_type;
unsigned char stream_id; /* video streaming id */
     unsigned char ext_b2_len; /* 1: length of the ext. b2 private data appended
to B2 frame */
     unsigned char rsvd;
     unsigned int  len;
} B2_HEADER

typedef struct {
     time_t       date_time;
     unsigned char  time_zone;  /* 0:-12, ..., 24:+13 */
     unsigned char  video_loss;  /* 0: video loss, 1 : video ok */
     unsigned char  motion;     /* 0x02: Motion 1 is active, 0x04: Motion 2 is
active, 0x08 Motion 3 is active */
     unsigned char  dio;        /* for DIs, 0: DI triggered. 1: no triggered */
     unsigned int   count;      /* frame counter */
     unsigned char  resolution;  /* 0:N720x480, ... */
     unsigned char  bitrate;    /* 0:28K, ... */
     unsigned char  fps_mode;  /* 0:MODE1(constant), 1:0:MODE2 */
     unsigned char  fps_number; /* In constant FPS mode, it indicates the video
server's constant FPS number.
                                     In variable FPS mode, in indicates the variable
FPS number which was requested by the TCP
host. If it is not in TCP, it indicates the variable
FPS number */
```

```
    struct timeval timestamp;
    unsigned short md_actives[3];   /* # of active microblocks in motion region
*/
    unsigned char reserved[2];
} PRIVATE_DATA_B2;
```

Name	Size
B2_HEADER	12 bytes (0x000001B2)
PRIVATE_DATA_B2	32 bytes

The user data segment total bytes : 44 bytes.

Bitstream Data

Name	Size
B0 Header	4 bytes (0x000001B0)
B0 Data	1 byte
B5 Header	4 bytes (0x000001B5)
B5 Data	1 byte
Sequence header	4 bytes (0x00000100)
Sequence data	17 bytes
	31 bytes

The Bitstream data segment total bytes : 31 bytes (B0 Header + B0 Data + B5 Header + B5 data + Sequence header + Sequence data).

I-Frame Data

Name	Size
B3 Header	4 bytes (0x000001B3)
B3 Data	3 bytes
B6 Header	4 bytes (0x000001B6)
Frame data	N bytes
	11 + N bytes

The I-Frame data segment total bytes : 11 bytes + N bytes(B3 Header + B3 Data + B6 Header + I-Frame data).

P-Frame Data Structure

B2 Header

```
typedef struct {
     B2_HEADER    header;
     PRIVATE_DATA_B2   prdata;
} VIDEO_B2_FRAME;

#define ACTI_HEAD_MSG_B2_VIDEO_MPEG4              0x01
#define ACTI_HEAD_MSG_B2_AUDIO_8KPCM              0x02
#define ACTI_HEAD_MSG_B2_AUDIO_TIMESTAMP_8KPCM   0x03
#define ACTI_HEAD_MSG_B2_VIDEO_MJPEG              0x04
#define ACTI_HEAD_MSG_B2_VIDEO_H264               0x05
typedef struct {
     unsigned char acti[4];  /* 00 00 01 B2 */
     unsigned char msg_type;
unsigned char stream_id; /* video streaming id */
     unsigned char ext_b2_len; /* 1: length of the ext. b2 private data appended
to B2 frame */
     unsigned char rsvd;
     unsigned int  len;
} B2_HEADER

typedef struct {
     time_t       date_time;
     unsigned char  time_zone;  /* 0:-12, ..., 24:+13 */
     unsigned char  video_loss;  /* 0: video loss, 1 : video ok */
     unsigned char  motion;     /* 0x02: Motion 1 is active, 0x04: Motion 2 is
active, 0x08 Motion 3 is active */
     unsigned char  dio;        /* for DIs, 0: DI triggered. 1: no triggered */
     unsigned int   count;      /* frame counter */
     unsigned char  resolution;  /* 0:N720x480, ... */
     unsigned char  bitrate;    /* 0:28K, ... */
     unsigned char  fps_mode;  /* 0:MODE1(constant), 1:0:MODE2 */
     unsigned char  fps_number; /* In constant FPS mode, it indicates the video
server's constant FPS number.
                                     In variable FPS mode, in indicates the variable
FPS number which was requested by the TCP
host. If it is not in TCP, it indicates the variable
FPS number */
     struct timeval timestamp;
     unsigned short md_actives[3];   /* # of active microblocks in motion region
*/
    unsigned char reserved[2];
} PRIVATE_DATA_B2;
```

Name	Size
B2_HEADER	12 bytes (0x000001B2)
PRIVATE_DATA_B2	32 bytes

The user data segment total bytes : 44 bytes.

P-Frame Data

Name	Size
B6 Header	4 bytes (0x000001B6)
Frame data	N bytes
	4 + N bytes

The P-Frame data segment total bytes : 4 bytes + N bytes(B6 Header data + P-Frame data).

Code Mapping in B2 Header

1.Time Zone

Time Zone	time_zone in PRIVATE_DATA_NEW
-12	0
-11	1
-10	2
-09	3
-08	4
-07	5
-06	6
-05	7
-04	8
-03	9
-02	10
-01	11
+00	12
+01	13
+02	14
+03	15
+04	16
+05	17
+06	18
+07	19
+08	20
+09	21
+10	22
+11	23
+12	24
+13	25
other time zone setting 1	26
another time zone setting 2	27
...	...

2.Resolution

Video Resolution	resolution in PRIVATE_DATA_NEW	
	Binary Value	Hex Value
NTSC		
N1920x1080	01000101b	0x45
N1600x1200	01000100b	0x44
N1280x1024	01000011b	0x43
N1280x960	01000010b	0x42
N1280x720	01000001b	0x41
N720x480	00000000b	0x00
N640x480	01000000b	0x40
N352x240	00000001b	0x01
N160x112	00000002b	0x02
N176x120	00000110b	0x06
PAL		
P720x576	00000011b	0x03
P640x480	11000000b	0xC0
P352x288	00000100b	0x04
P176x144	00000101b	0x05

3.Bitrate

Video Bitrate	bitrate in PRIVATE_DATA_NEW
28K	0
56K	1
128K	2
256K	3
384K	4
500K	5
750K	6
1M	7
1.2M	8
1.5M	9
2M	10
2.5M	11
3M	12
3.5M	13
4M	14
4.5M	15
5M	16
5.5M	17
6M	18

Note: In MJPEG mode and Variable Bitrate mode, this bitrate setting in B2 is not valid. It will be fixed at the current encoder bitrate setting which is for constant bit rate mode with MPEG4 or H.264 encoding.

Audio frame

The data structure of audio frame is simpler than video frame. We can see as below.

AUDIO_B2(0x000001B2)	Audio Frame data (audio 8K pcm payload data)

```
typedef struct {
     B2_HEADER header;
     struct timeval timestamp;
     unsigned char reserved[8];
} AUDIO_B2;
```

Name	**Size**
AUDIO_B2	28bytes (0x000001B2)
Audio Frame Data	N bytes
	28 + N bytes

The audio total bytes : AUDIO_B2 + FrameData (28 bytes + N)

Notice: The old version firmware send B2_HEADER(12 bytes) instead AUDIO_B2 (28 bytes)

Control Connect Session

Besides the video session we can get some of control from the control connection session.

Send a 128bytes command to the IP device.

Build a connection

When we want to connect to video server by control session. We can follow below step to connect with video server.

1. Create a TCP socket connection that is needed to specific the IP and port. The default port is 6001.
2. Send a 128bytes command to video server. That includes User ID 32 bytes ,Token command and reserved bytes.
3. Then we will get the response code. It are total 128 bytes code and inlcdes connect result, error code and reserve bytes
4. Receive the data that will be control data.

TCP Authentication Request and Response Frame

```
TCP Authentication Request and Response Frame
/* ##### definitions in msg_type in ACTI_HEADER ##### */
#define ACTI_HEAD_MSG_VARIABLE_FPS_REQ  0x20
#define ACTI_HEAD_MSG_PAUSE_ON_REQ     0x21
#define ACTI_HEAD_MSG_PAUSE_OFF_REQ 0x22

/* ##### definitions in server_reply in ACTI_HEADER ##### */
#define ACTI_HEAD_HOST_REQUEST 0
#define AVTI_HEAD_SERVER_REPLY 1

typedef struct {
     unsigned char acti[4];  /* A C T i */
unsigned short msg_type; /* not used. reset to 0 */
unsigned char server_reply; /* not used, reset to 0 */
unsigned char stream_id; /* same definition as B2_HEADER */
     unsigned int  len;
} ACTI_HEADER;

/* ##### definitions in encryption_type in TCP_AUTHEN_REQ #### */
#define NAME_ENCODED_NONE      0
#define NAME_ENCODED_BASE64         1
typedef struct {
     char user_name[32];
     char rsvd[28];
     unsigned short stream_id;   /* same definition as B2_HEADER */
     unsigned short encryption_type;
     int  user_pwd[64];
} TCP_AUTHEN_REQ;
```

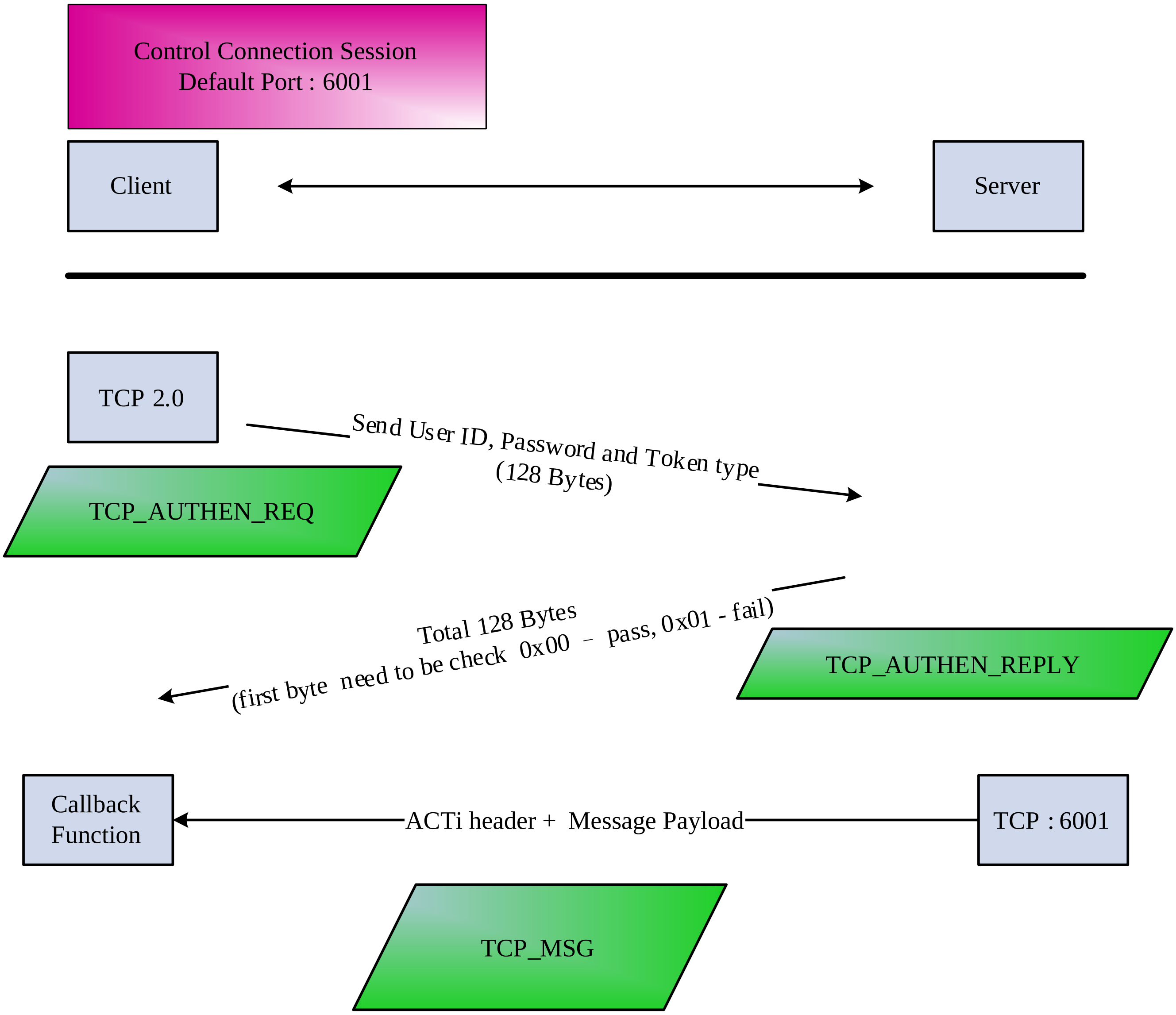
```
typedef struct {
    char status;
    char rsvd1;
    unsigned short stream_id;  /* same definition as B2_HEADER */
    int  sock;
    char camera_name[32];
    char rsvd2[88];
} TCP_AUTHEN_REPLY;

typedef struct {
    ACTI_HEADER header;
    unsigned char     msg[1];      /* variable length */
} TCP_MSG;
```

In msg_type = ACTI_HEAD_MSG_VARIABLE_FPS_REQ, the msg[0] in TCP_MSG is the variable FPS number

In msg_type = ACTI_HEAD_MSG_PAUSE_ON_REQ or ACTI_HEAD_MSG_PAUSE_OFF_REQ, there is no msg[].

In the reply packet, the msg[0] is the return code. The definition of the return code is listed below.

```
#define TCP_REPLY_CODE_OK      0x00
#define TCP_REPLY_CODE_ERR 0x01
```


Control Connection Session
Default Port : 6001
Client
Server
TCP 2.0
Send User ID, Password and Token type
(128 Bytes)
TCP_AUTHEN_REQ
Total 128 Bytes
(first byte need to be check 0x00 – pass, 0x01 - fail)
TCP_AUTHEN_REPLY
Callback Function
ACTi header + Message Payload
TCP : 6001
TCP_MSG

Control Authentication Request and Response Frame

```
/* ##### definitions in msg_type in ACTI_HEADER */
/* LIVE CHECK used in the control session */
#define ACTI_HEAD_MSG_LIVE     0x30
#define ACTI_HEAD_MSG_EXIT          0x31
/* DIOs used in the control session */
#define ACTI_HEAD_MSG_DIO_OUT  0x32
#define ACTI_HEAD_MSG_DIO_STATUS  0x33
#define ACTI_HEAD_MSG_DIO_INPUT   0x34  /* not used */
/* RS485 used in the control session */
#define ACTI_HEAD_MSG_SERIAL_RECV  0x35 /* not used */
#define ACTI_HEAD_MSG_SERIAL_SEND  0x36
/* AUDIO_IN used in the control session */
#define ACTI_HEAD_MSG_AUDIO_PLAY    0x37
/* VIDEO LOSS used in the control session */
#define ACTI_HEAD_MSG_VIDEO_LOSS   0x38  /* not used */
/* MOTION used in the control session */
#define ACTI_HEAD_MSG_MOTION_DETECT 0x39 /* not used */
/* CAMERA NAME in the control session */
#define ACTI_HEAD_MSG_CAMERA_NAME  0x40
/* ##### definitions in server_reply in ACTI_HEADER */
#define ACTI_HEAD_HOST_REQUEST 0
#define AVTI_HEAD_SERVER_REPLY 1

typedef struct {
    unsigned char acti[4];  /* A C T i */
unsigned short msg_type; /* not used. reset to 0 */
unsigned char server_reply; /* not used, reset to 0 */
unsigned char stream_id; /* same definition as B2_HEADER */
    unsigned int  len;
} ACTI_HEADER;
```

```
typedef struct {
    char user[32];
    int  token;
    char reserved[24];
    unsigned short stream_id; /* same definition as B2_HEADER */
    unsigned short encryption_type; /* same definition as CP_AUTHEN_REQ */
    char pwd[64];
} CTRL_REQ;

/* ##### definitions in the result in CTRL_RSP ##### */
#define RSP_OK                      0x00
#define RSP_ERR                     0x01
/* ##### definitions in the err_code in CTRL_RSP ##### */
#define ERR_NO_ERROR                0x00000000
#define ERR_ACCOUNT                 0x00010001
#define ERR_UNKNOWN_TOKEN           0x00010002
#define ERR_CTRL_TOKEN_BUSY         0x00010003
#define ERR_AUDIO_TOKEN_BUSY        0x00010004
#define ERR_AUDIO_NOT_SUPPORT       0x00010005

typedef struct {
    char result;
    char reserved1;
    unsigned short stream_id; /* same definition as B2_HEADER */
    char reserved1[3];
    int  err_code;
    int  ip_addr;
    char reserved2[116];
} CTRL_RSP;
```

```
typedef struct {
    ACTI_HEADER header;
    unsigned char   msg[1];    /* variable length */
} CTRL_MSG_FRAME;
```

DOs coding in the msg[0] (1byte):

Bit[4] : DO2, Bit[3] : DO1, Bit[1]=DI2, Bit[0]:DI1, where 1: high level of DO, 0: low level of DO.

RS485 coding in msg[] (variable length):

Data string of the RS485/RS422/RS232 data

Camera name coding in msg[] (max 31 bytes) :

Encoder's VIDEO_CAMERA_NAME setting

Audio data in msg[] (fixed to 4096 bytes):

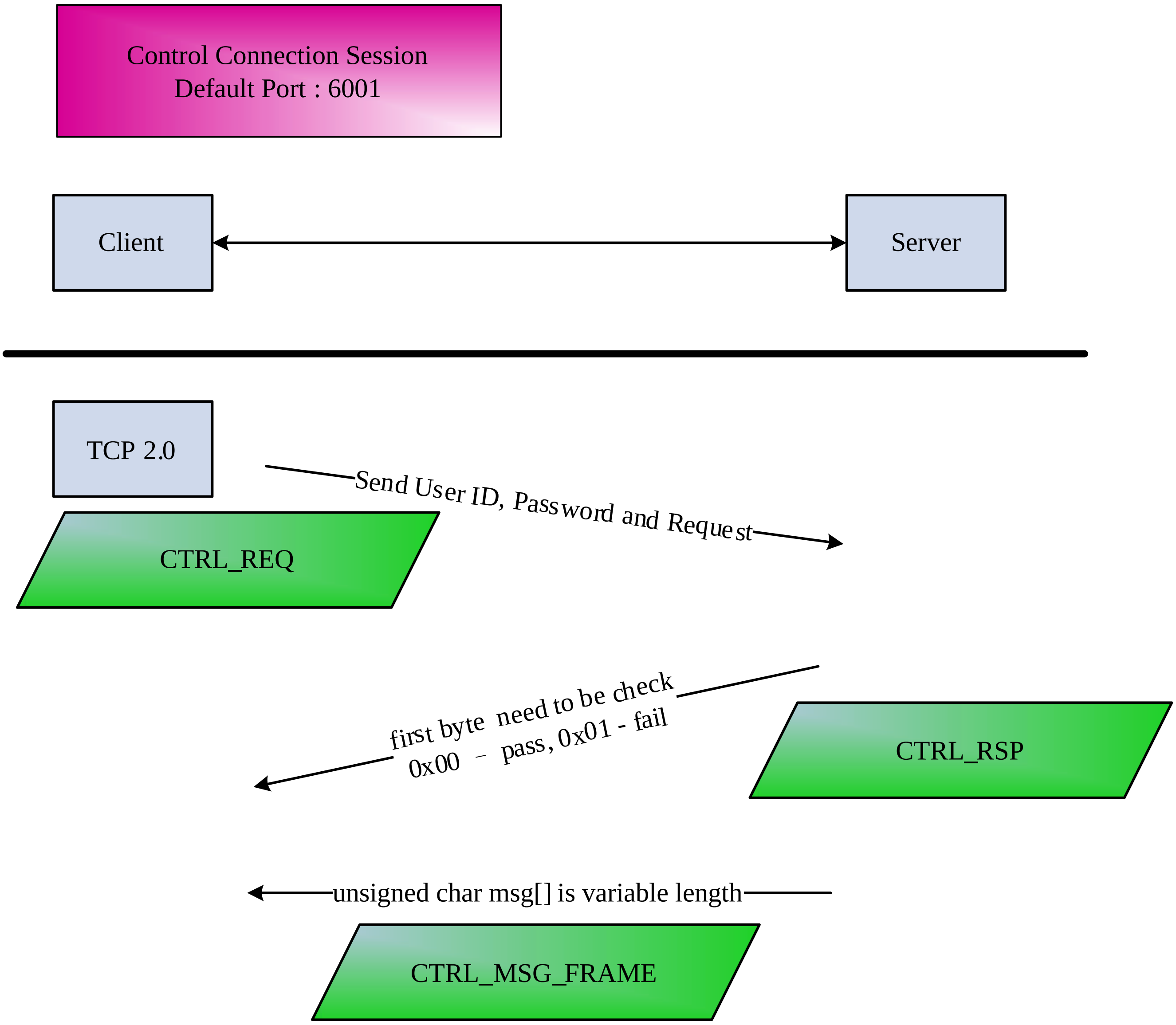
Audio data in host

Motion coding in the msg[0] (1byte):

Bit[1]: motion region 1, Bit[2]: motion region 2, Bit[3]: motion region 3, where 0: no motion, 1: detected motion

Video Loss coding in the msg[0] (1byte):

0: Video Loss, 1: Video Lock



Control Connection Session
Default Port : 6001

Client

Server

TCP 2.0

Send User ID, Password and Request

CTRL_REQ

first byte need to be check
0x00 – pass, 0x01 - fail

CTRL_RSP

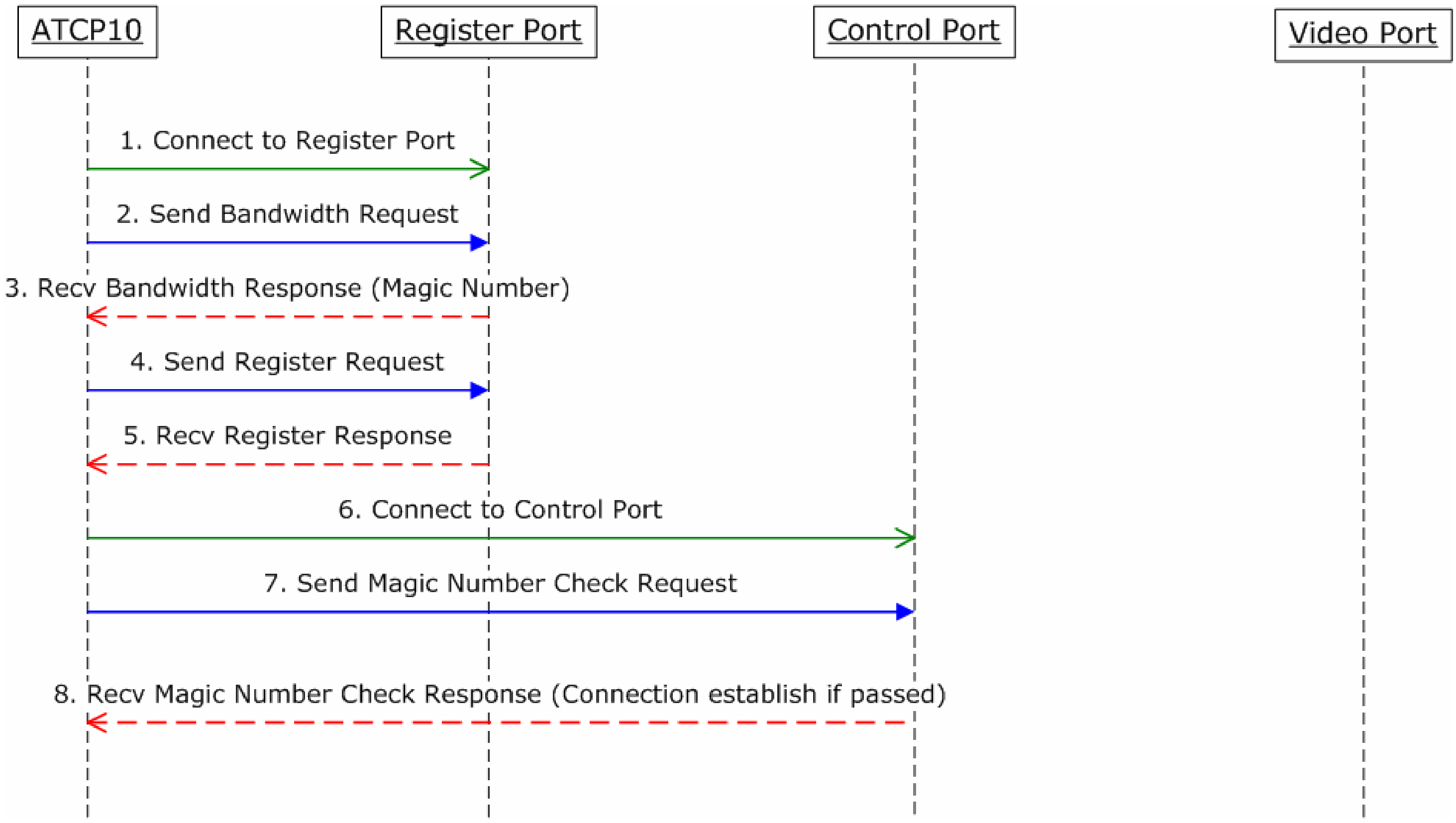
unsigned char msg[] is variable length

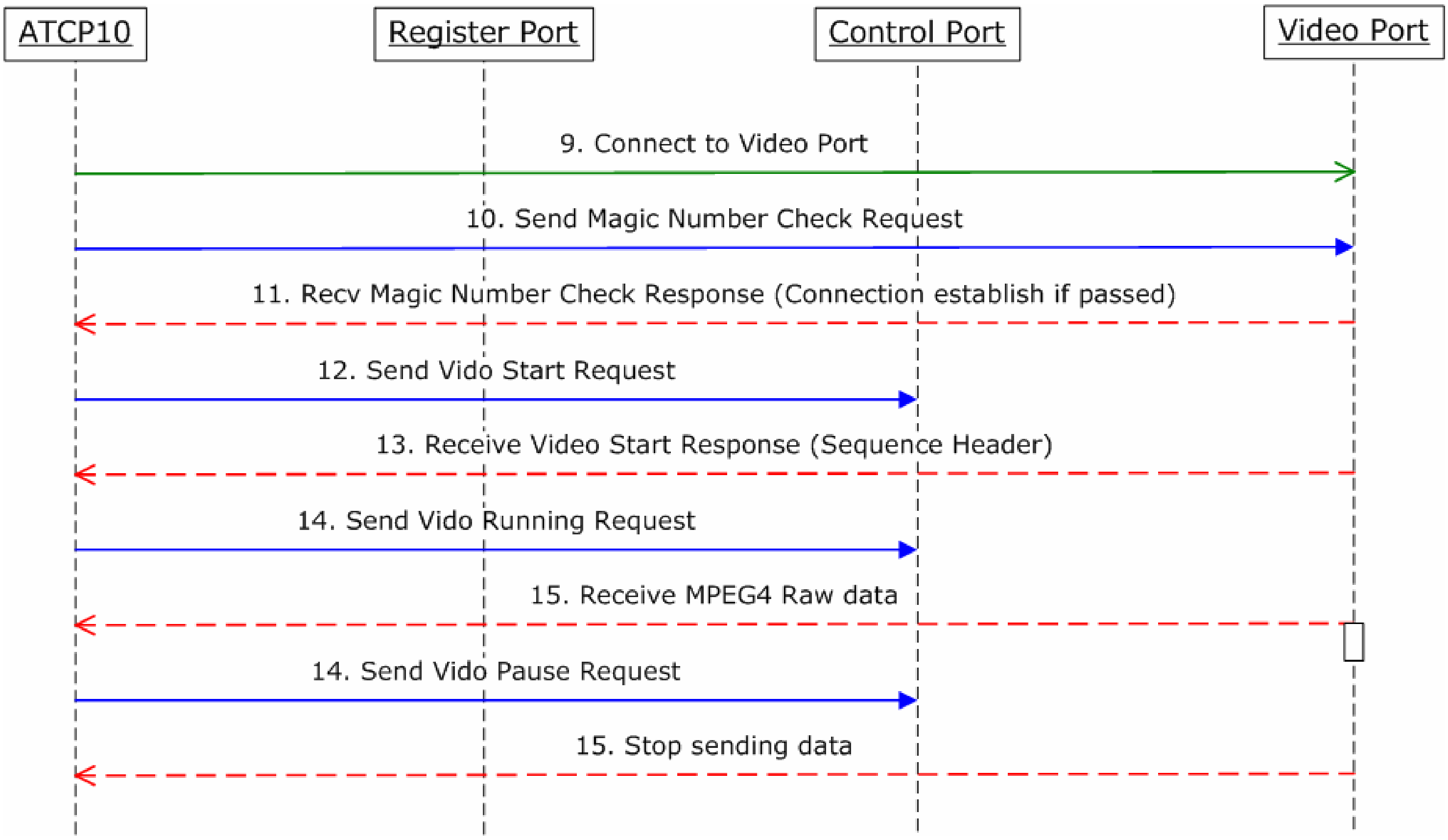
CTRL_MSG_FRAME

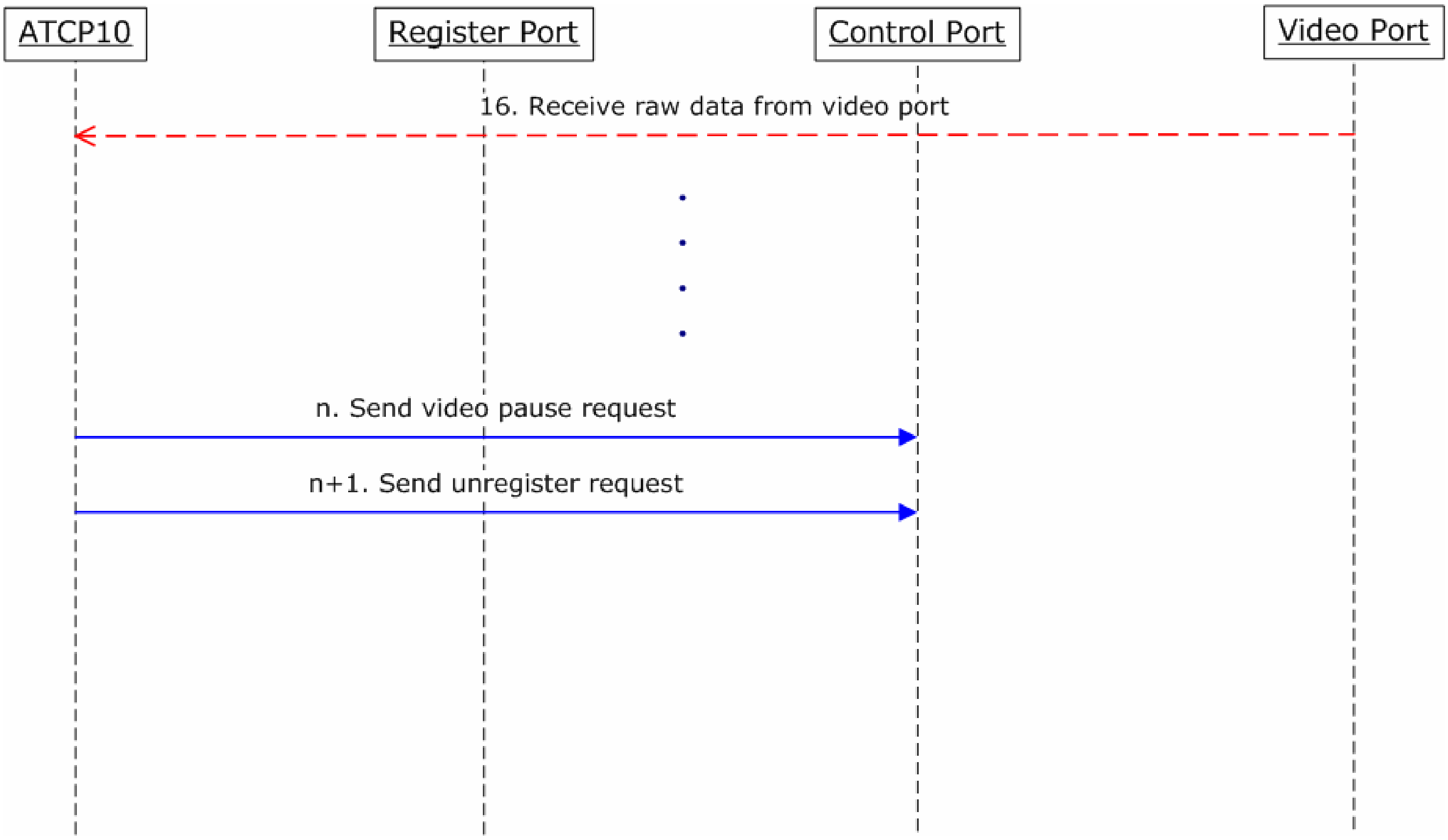
9 TCP and RTP/RTSP Packet Format

TCP v1.0 Packet

TCP v1.0 Video Connect Flow


ATCP10
Register Port
Control Port
Video Port
1. Connect to Register Port
2. Send Bandwidth Request
3. Recv Bandwidth Response (Magic Number)
4. Send Register Request
5. Recv Register Response
6. Connect to Control Port
7. Send Magic Number Check Request
8. Recv Magic Number Check Response (Connection establish if passed)


ATCP10
Register Port
Control Port
Video Port
9. Connect to Video Port
10. Send Magic Number Check Request
11. Recv Magic Number Check Response (Connection establish if passed)
12. Send Vido Start Request
13. Receive Video Start Response (Sequence Header)
14. Send Vido Running Request
15. Receive MPEG4 Raw data
14. Send Vido Pause Request
15. Stop sending data


ATCP10
Register Port
Control Port
Video Port
16. Receive raw data from video port
n. Send video pause request
n+1. Send unregister request

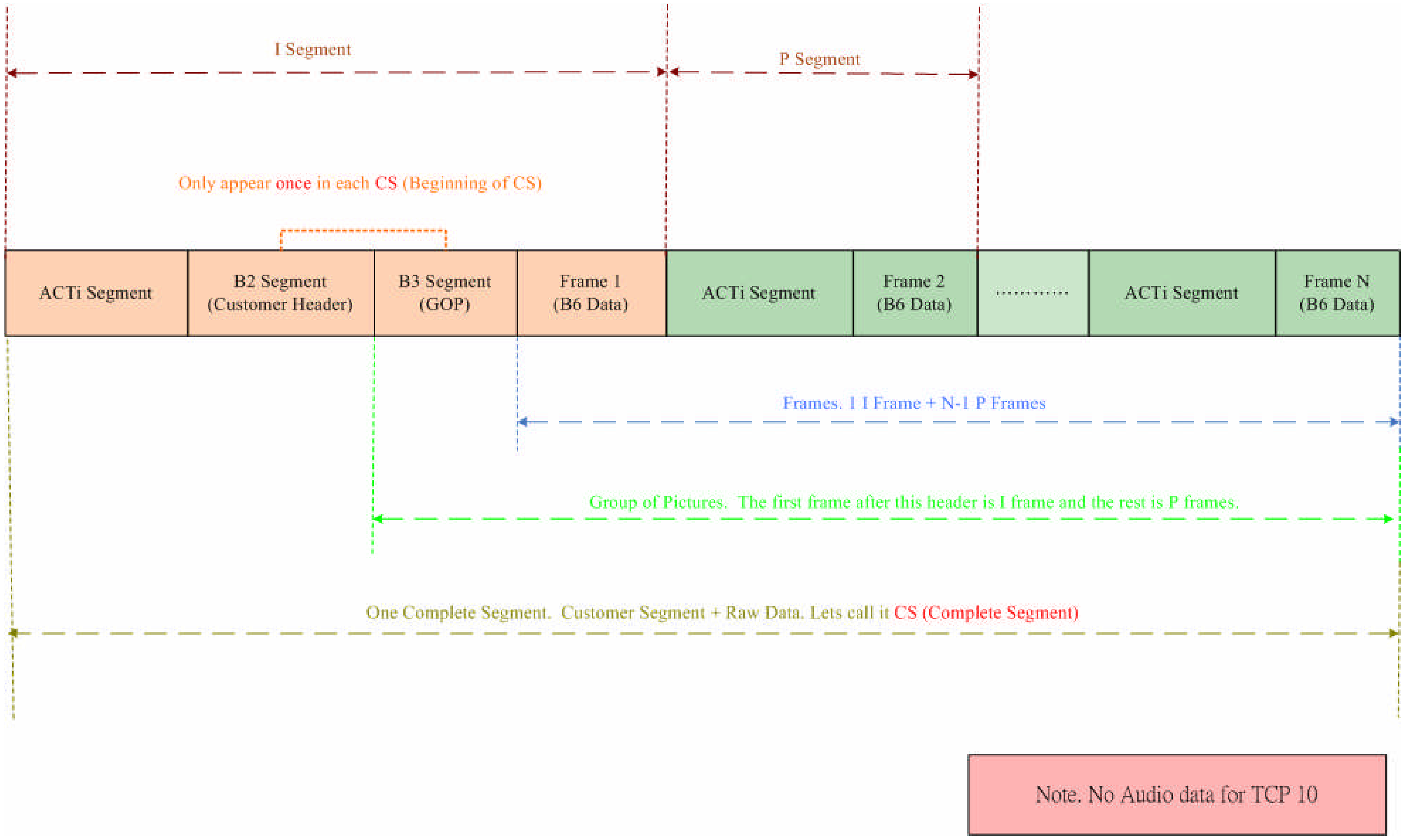
Note that:

1. Live check packet send between ATCP & Control port every constant time.

Disconnect steps

1. Disconnect register port
2. Do n and n+1 steps
3. Disconnect control port
4. Disconnect video port

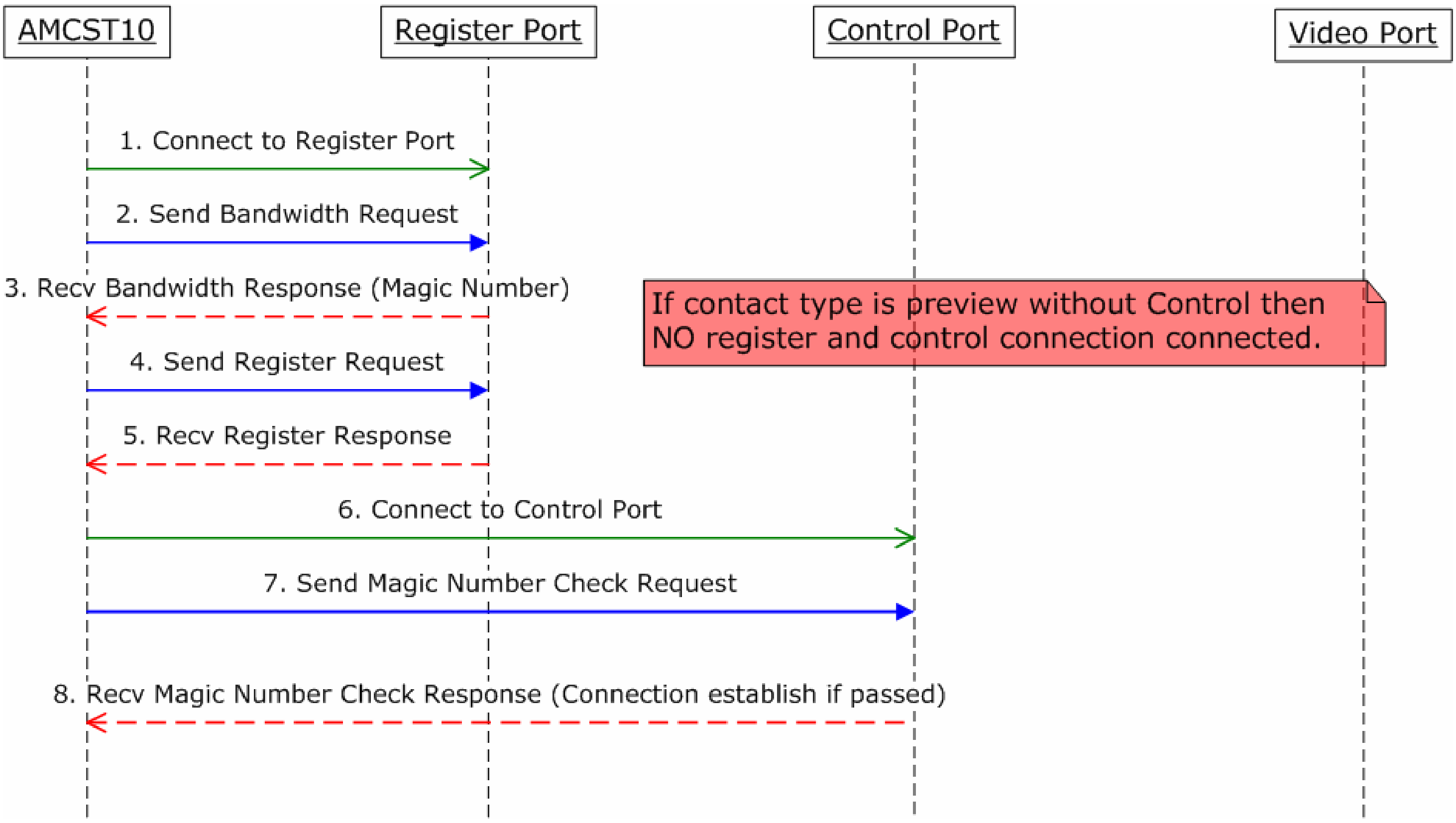
TCP v1.0 Video Packet Format

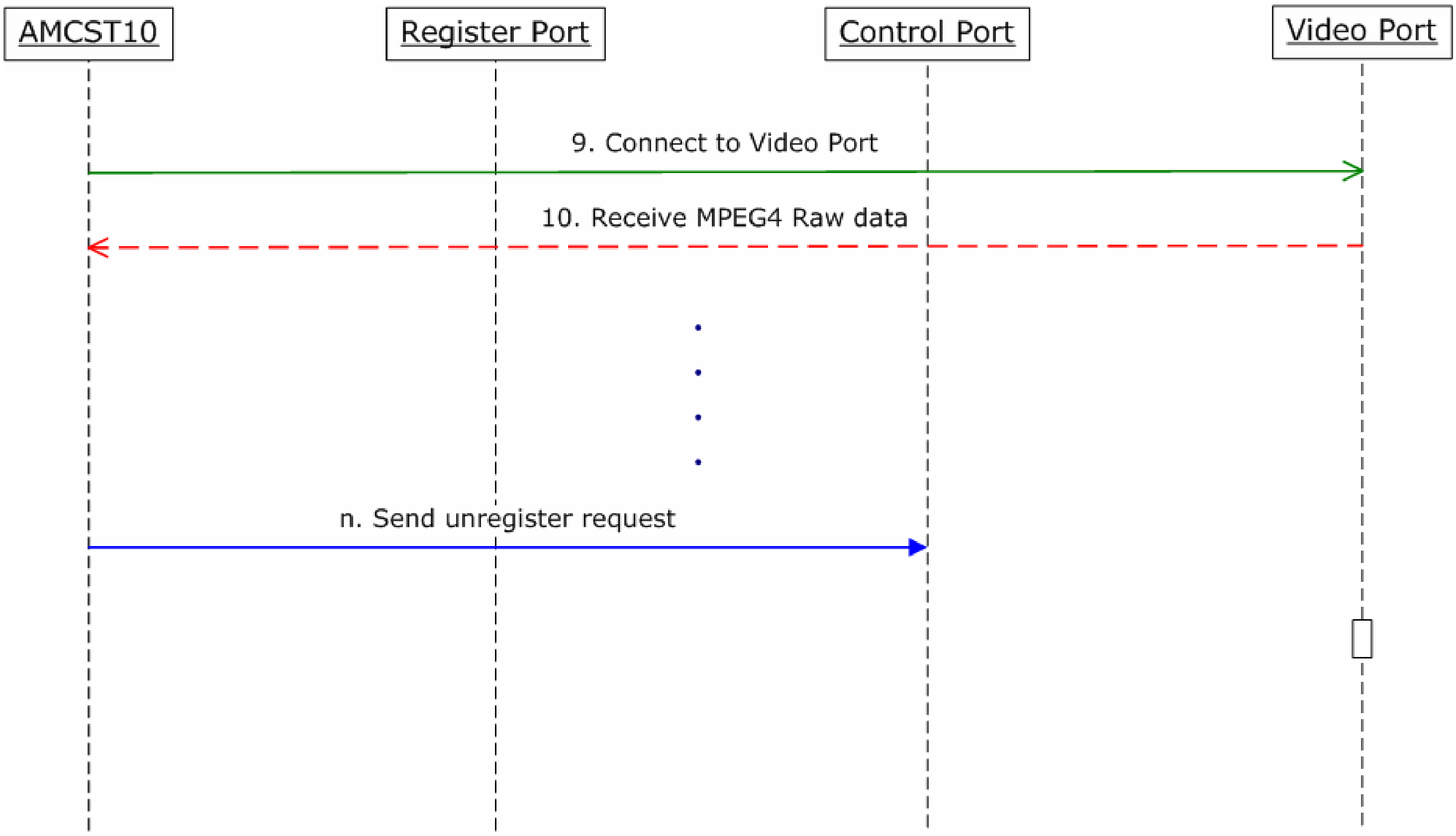

I Segment
P Segment
Only appear once in each CS (Beginning of CS)
ACTi Segment
B2 Segment (Customer Header)
B3 Segment (GOP)
Frame 1 (B6 Data)
ACTi Segment
Frame 2 (B6 Data)
…………
ACTi Segment
Frame N (B6 Data)
Frames. 1 I Frame + N-1 P Frames
Group of Pictures. The first frame after this header is I frame and the rest is P frames.
One Complete Segment. Customer Segment + Raw Data. Lets call it CS (Complete Segment)
Note. No Audio data for TCP 10

```
ACTi Segment
{
char ACTi[4];   // String "ACTi"
DWORD dwVersion;  // 0x00010022
DWORD dwLength;  // Data length
}
```

Multicast v1.0 Packet

Multicast v1.0 Video Connect Flow


AMCST10
Register Port
Control Port
Video Port
1. Connect to Register Port
2. Send Bandwidth Request
3. Recv Bandwidth Response (Magic Number)
4. Send Register Request
5. Recv Register Response
6. Connect to Control Port
7. Send Magic Number Check Request
8. Recv Magic Number Check Response (Connection establish if passed)
If contact type is preview without Control then
NO register and control connection connected.



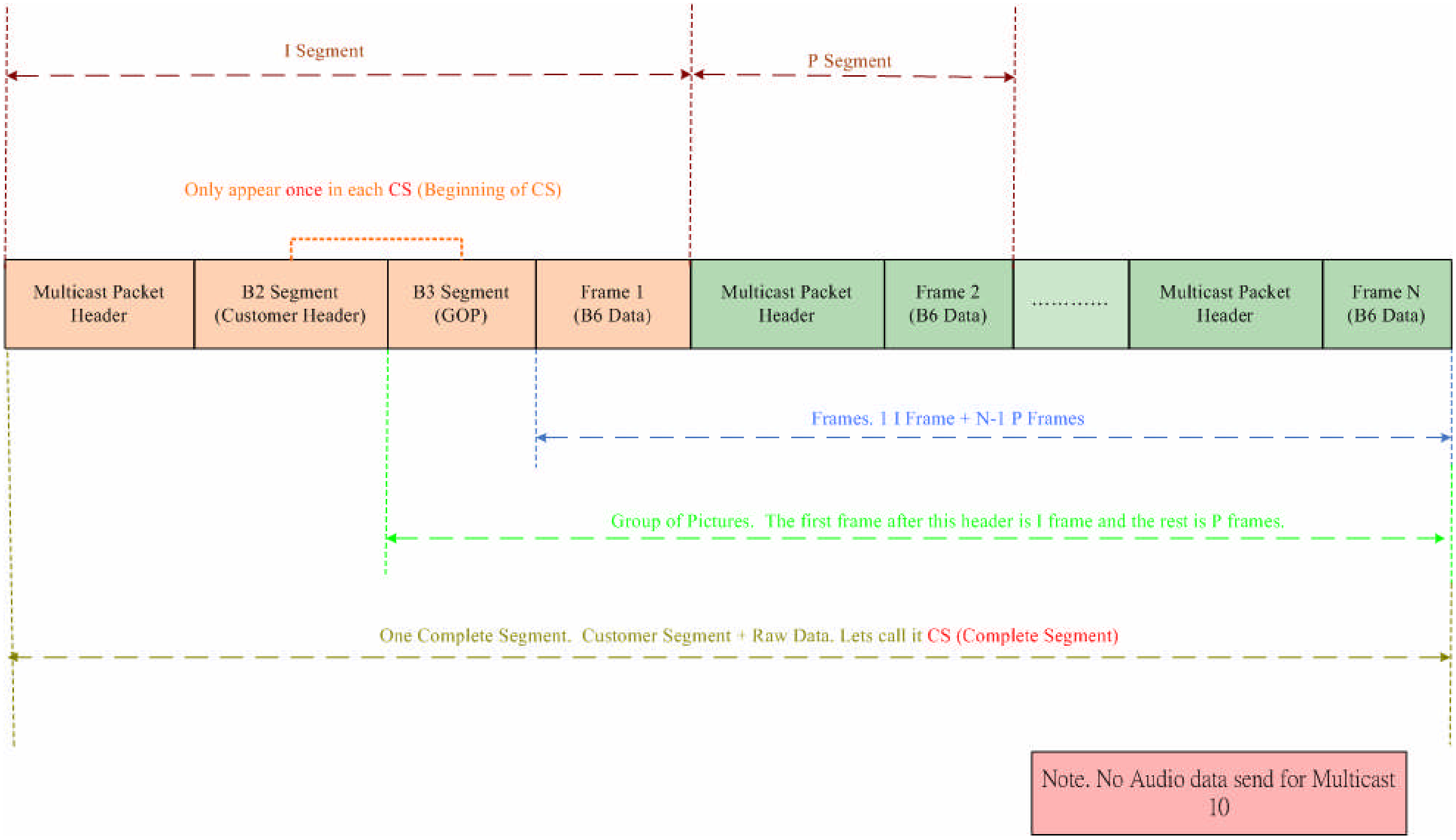
Note that:

Live check packet send between ATCP & Control port every constant time.

Disconnect steps

1. Disconnect register port
2. Do n.
3. Disconnect control port
4. Disconnect video port

Multicast v1.0 Video Packet Format


I Segment
P Segment
Only appear once in each CS (Beginning of CS)
Multicast Packet Header
B2 Segment (Customer Header)
B3 Segment (GOP)
Frame 1 (B6 Data)
Multicast Packet Header
Frame 2 (B6 Data)
....
Multicast Packet Header
Frame N (B6 Data)
Frames. 1 I Frame + N-1 P Frames
Group of Pictures. The first frame after this header is I frame and the rest is P frames.
One Complete Segment. Customer Segment + Raw Data. Lets call it CS (Complete Segment)
Note. No Audio data send for Multicast 10

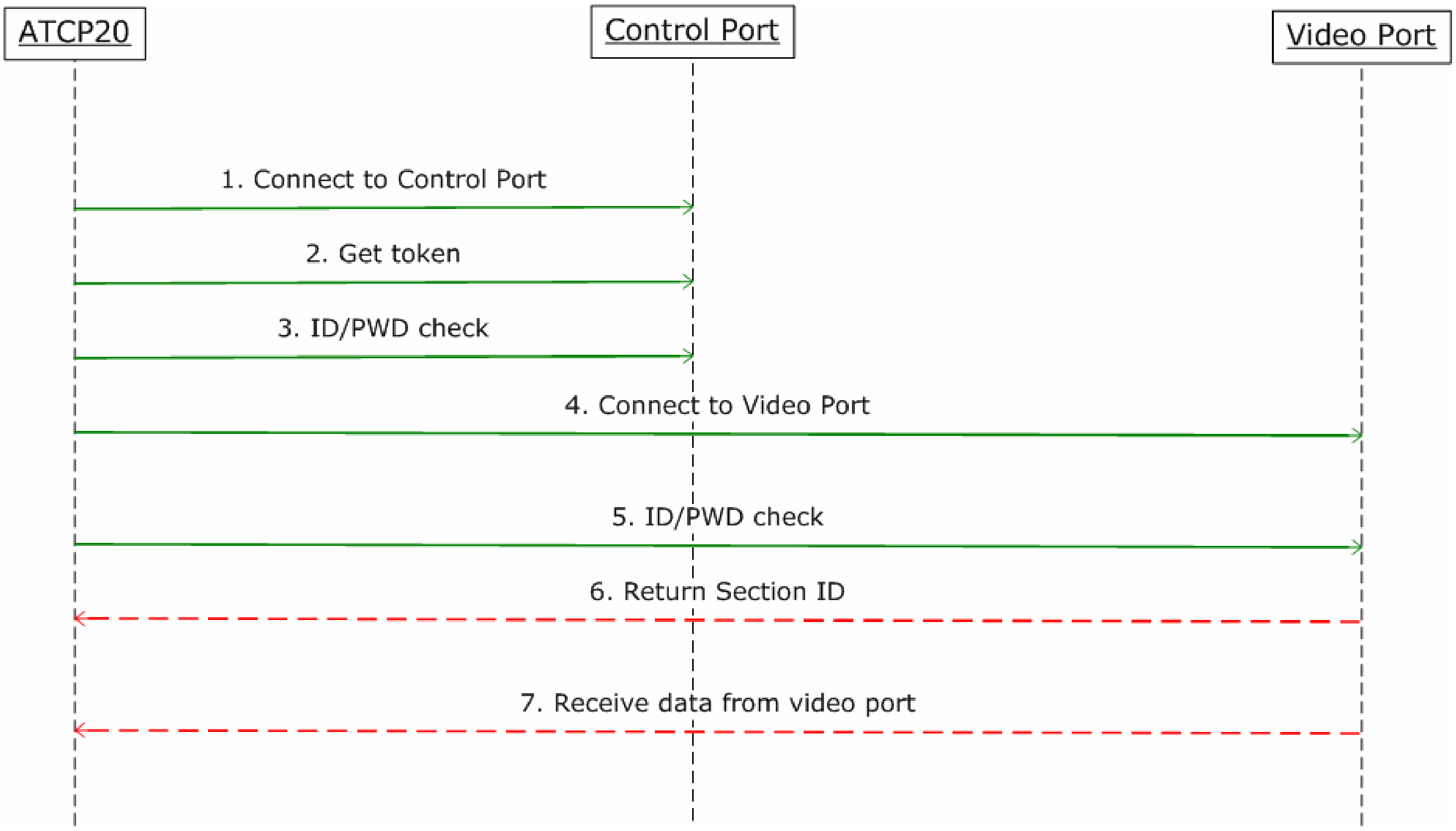
```
typedef struct tagMCPacketHead
{
unsigned char StreamId;
unsigned char StreamSubId;
unsigned char KeyPacket;
unsigned char TotalPacket;
unsigned char PacketNum;
unsigned char FrameCheckSum;
unsigned char Resolution;
unsigned char Fps;
unsigned int  FrameNum;
unsigned int  FrameLen;
} MCPacketHead;
```

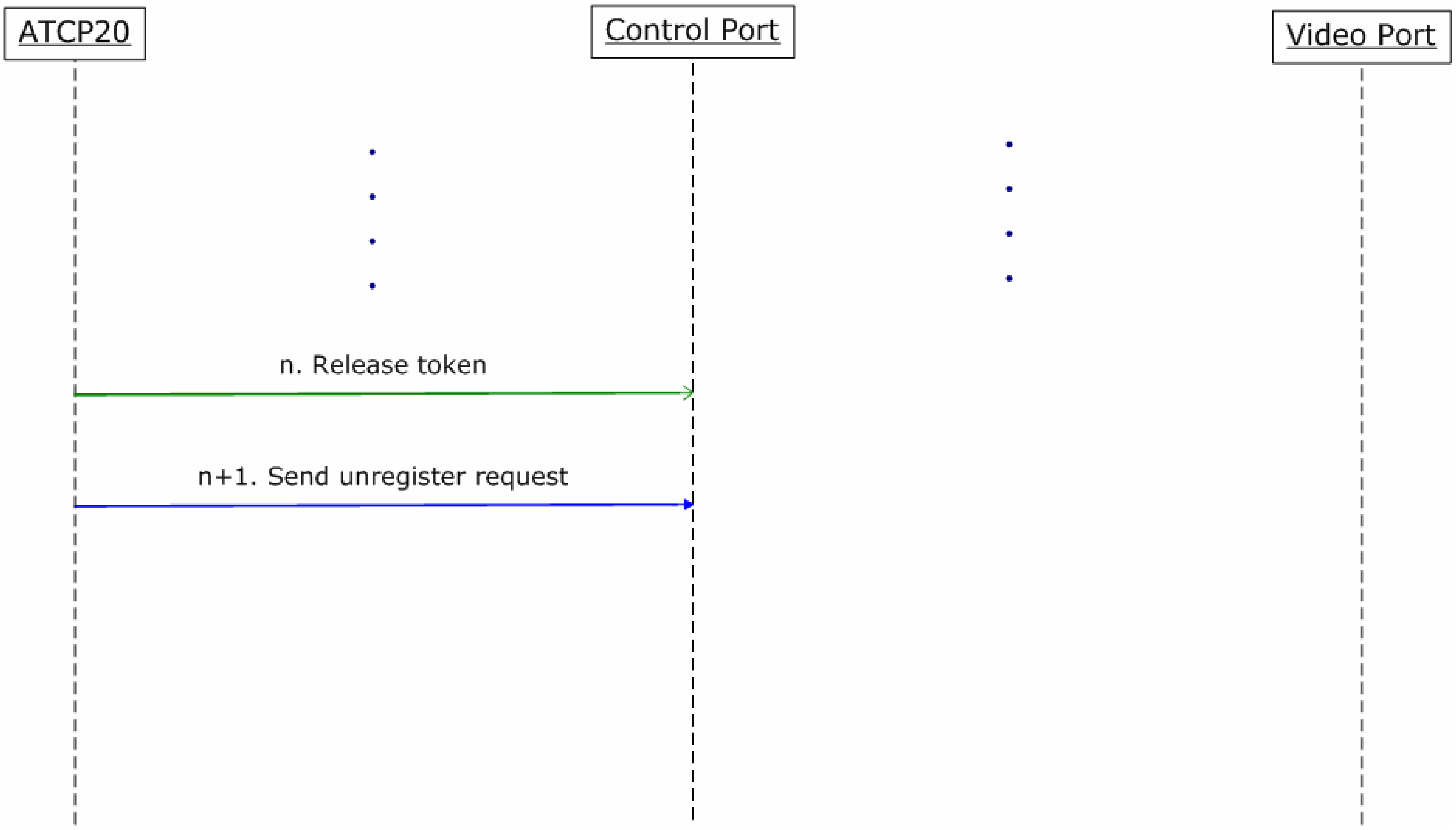
Important Note:

1. Key packet attribute is very important to determine the last packet of the frame.
2. Only key packet has both FPS and Resolution information.

TCP v2.0 Packet

TCP 2.0 Video Connect Flow





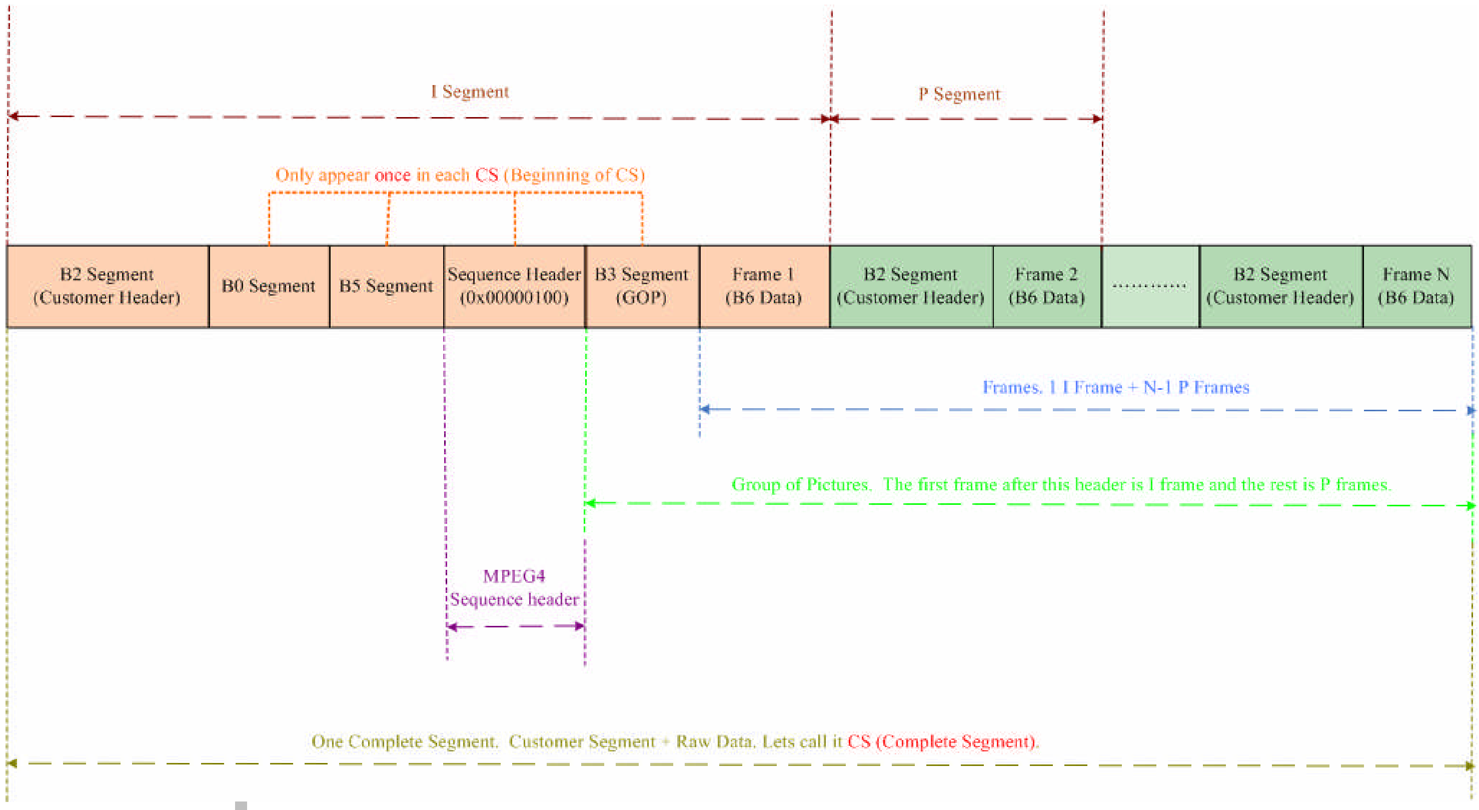
Disconnect steps

1. Do n and n+1 steps

2. Disconnect control port

3. Disconnect video port

TCP 2.0 Video Packet Format


I Segment
P Segment
Only appear once in each CS (Beginning of CS)
B2 Segment (Customer Header)
B0 Segment
B5 Segment
Sequence Header (0x00000100)
B3 Segment (GOP)
Frame 1 (B6 Data)
B2 Segment (Customer Header)
Frame 2 (B6 Data)
…………
B2 Segment (Customer Header)
Frame N (B6 Data)
Frames. 1 I Frame + N-1 P Frames
Group of Pictures. The first frame after this header is I frame and the rest is P frames.
MPEG4 Sequence header
One Complete Segment. Customer Segment + Raw Data. Lets call it CS (Complete Segment).

```
typedef struct {
     B2_HEADER     header;
     PRIVATE_DATA   prdata;
} VIDEO_B2_FRAME; //44 bytes (details in chapter 8)
```

TCP of ACTi :

(a) I Frame

0x000001B2	User Data	0x000001B0	0x000001B5	0x00000100
0x00000120	0x000001B3	0x000001B6	I Frame	

(b) P Frame

0x000001B2	User Data	0x000001B6	P Frame

(c) Audio Frame

0x000001B2	Audio Frame

Multicast of ACTi :

(a) I Frame

Multicast Header	0x000001B2	User Data	0x000001B0	0x000001B5
0x00000100	0x00000120	0x000001B3	0x000001B6	I Frame

(b) P Frame

Multicast Header	0x000001B2	User Data	0x000001B6	P Frame

(c) Audio Frame

Multicast Header	0x000001B2	Audio Frame

RTP over UDP :

```
Video :
    (a) [RTP Header][000001B0][000001B5][00000100][00000120][000001B3]
                    [000001B6][I-Frame][000001B2][User Data]
    (b) [RTP Header][000001B6][P-Frame][000001B2][User Data]
```

```
Audio :
    (a) [RTP Header][Audio Frame]
```

RTP over Multicast :

```
Video :
    (a) [RTP Header][000001B0][000001B5][00000100][00000120][000001B3]
                    [000001B6][I-Frame][000001B2][User Data]
    (b) [RTP Header][000001B6][P-Frame][000001B2][User Data]
Audio :
    (a) [RTP Header][Audio Frame]
```

Exported Struct

Media Connection Configuration :

```
typedef struct structural_MEDIA_CONNECTION_CONFIG
{
     int                ContactType;               // Contact Type
     int                ChannelNumber;             // Channel number
     char               UniCastIP[16];             // Unicasat IP address
     char               MultiCastIP[16];           // Multicast IP address
     char               PlayFileName[256];         // Playback file name
     char               UserID[64];                // User login ID
     char               Password[64];              // User login password
     unsigned long      RegisterPort;              // Register port number
     unsigned long      StreamingPort;             // Streaming port number
     unsigned long      ControlPort;               // Control port number
     unsigned long      MultiCastPort;             // Multicast port number
     unsigned long      SearchPortC2S;             // Search port number for client
                                                   // to server
     unsigned long      SearchPortS2C;             // Search port number for server
                                                   // to client.
     unsigned long      HTTPPort;                  // HTTP port number
     unsigned long      RTSPPort;                  // RTSP port number
     unsigned long      VideoRTPOverMCastPort;     // RTP over Multicast port number
                                                   // for video
     unsigned long      AudioRTPOverMCastPort;     // Rtp over Multicast port number
                                                   // for audio.
     int                ConnectTimeOut;            // Time out value for connection
}MEDIA_CONNECTION_CONFIG;
```

Media Video Configuration :

```
typedef struct structural_MEDIA_VIDEO_CONFIG
{
     DWORD dwTvStander;          // 0:NTSC 1:PAL
     DWORD dwVideoResolution;    // See resolution definition
     DWORD dwBitsRate;           // See bit rate definition
     DWORD dwVideoBrightness;    // 0 ~ 100 : Low ~ High
     DWORD dwVideoContrast;      // 0 ~ 100 : Low ~ High
     DWORD dwVideoSaturation;    // 0 ~ 100 : Low ~ High
     DWORD dwVideoHue;           // 0 ~ 100 : Low ~ High
     DWORD dwFps;                // 0 ~ 30 frame pre second
} MEDIA_VIDEO_CONFIG;
```

Media Port Information :

```
typedef struct structural_MEDIA_PORT_INFO   // Device port info.
{
    unsigned long PORT_HTTP;                // HTTP Port
    unsigned long PORT_SearchPortC2S;       // Search Port 1
    unsigned long PORT_SearchPortS2C;       // Search Port 2
    unsigned long PORT_Register;            // Register Port
    unsigned long PORT_Control;             // Control Port
    unsigned long PORT_Streaming;           // Streaming Port
    unsigned long PORT_Multicast;           // Multicast Port
    unsigned long PORT_RTSP;                // RTSP Port
} MEDIA_PORT_INFO;
```

Media Render Information

```
typedef struct structural_MEDIA_RENDER_INFO
{
    int   RenderInterface;                  // Reserve, in the future this
                                            // parameter meaning DGDI or DDRAW
    HWND  hWnd;                             // The handle of drawing window
    RECT  rect;                             // rect. info. of drawing window.
} MEDIA_RENDER_INFO;
```

Media Motion Information

```
typedef struct structural_MEDIA_MOTION_INFO
{
    DWORD dwEnable;                         // Enable flag
    DWORD dwRangeCount;                     // Number of range count
    DWORD dwRange[3][4];                    // Range information
    DWORD dwSensitive[3];                   // 0 - 100
} MEDIA_MOTION_INFO;
```

MPEG4 File Record Information

```
typedef struct structural_MP4FILE_RECORD_INFO
{
    time_t          tBeginTime;             // Begin time of record file
    time_t          tEndTime;               // End time of record file.
    BYTE            btTimeZone;             // Time zone
    DWORD           dwGOP;                  // GOP
    DWORD           dwFrameCount;           // Number of frames
    ULONGLONG       FileSize;               // Size of record file
} MP4FILE_RECORD_INFO;
```

Time Zone

0 : GMT-12	1 : GMT-11	2 : GMT-10	3 : GMT-09	4 : GMT-08
5 : GMT-07	6 : GMT-06	7 : GMT-05	8 : GMT-04	9 : GMT-03
10 : GMT-02	11 : GMT-01	12 : GMT+00	13 : GMT+01	14 : GMT+02
15 : GMT+03	16 : GMT+04	17 : GMT+05	18 : GMT+06	19 : GMT+07
20 : GMT+08	21 : GMT+09	22 : GMT+10	23 : GMT+11	24 : GMT+12
25 : GMT+13				

DI Notify

```
typedef struct structural_NOTIFY_DI
{
    HANDLE   DIEvent;                    // [IN] Event handle
    BYTE DI;                             // [OUT] Digital input
}NOTIFY_DI;
```

Time Code Notify

```
typedef struct structural_NOTIFY_TIMECODE
{
    HANDLE   TimeCodeEvent;              // [IN] Event handle
    DWORD    dwTimeCode;                 // [OUT] Time code
}NOTIFY_TIMECODE;
```

Raw Data Refresh Notify

```
typedef struct structural_NOTIFY_RAWDATAREFRESH
{
    HANDLE   RawDataRefreshEvent;        // [IN] Event handle
    void*    pBuffer;                    // [OUT] Buffer pointer
    int      nFillLength;                // [IN/OUT] Buffer length
}NOTIFY_RAWDATA_REFRESH;
```

Video Status Notify

```
typedef struct structural_NOTIFY_VIDEOSTATUS
{
    HANDLE   VideoLossEvent;             // [IN] Event handle
    HANDLE   VideoRecoveryEvent;         // [IN] Event handle
}NOTIFY_VIDEO_STATUS;
```

Network Loss Notify

```
typedef struct structural_NOTIFY_NETWORKLOSS
{
    HANDLE   NetworkLossEvent;              // [IN] Event handle
}NOTIFY_NETWORK_LOSS;
```

Motion Detection Notify

```
typedef struct structural_NOTIFY_MOTIONDETECTION
{
    HANDLE   MotionDetectionEvent;          // [IN] Event handle
    BYTE     MotionDetection;               // [OUT] Motion detection info.
}NOTIFY_MOTION_DETECTION;
```

Image Refresh Notify

```
typedef struct structural_NOTIFY_IMAGE_REFRESH
{
    HANDLE   ImageRefreshEvent;             // [IN] Event handle
    void*    pImage;                        // [OUT] Buffer pointer
    int      nFillLength;                   // [IN/OUT] Buffer length
}NOTIFY_IMAGE_REFRESH;
```

After Render Notify

```
typedef struct structural_NOTIFY_AFTER_RENDER
{
    HANDLE   AfterRenderEvent;              // [IN] Event handle
}NOTIFY_AFTER_RENDER;
```

Resolution Change Notify

```
typedef struct structural_NOTIFY_RESOLUTION_CHANGE
{
    HANDLE   ResolutionChangeEvent;         // [IN] Event handle
    int      nResolution;                   // [OUT] Resolution
}NOTIFY_RESOLUTION_CHANGE;
```

Resolution Map

In this chapter, new megapixel resolution has been added.

```
#define NTSC_720x480  0             ///< #0# - NTSC - 720 x 480
#define NTSC_352x240  1             ///< #1# - NTSC - 352 x 240
#define NTSC_160x112  2             ///< #2# - NTSC - 160 x 112
#define PAL_720x576        3            ///< #3# - PAL  - 720 x 576
#define PAL_352x288        4            ///< #4# - PAL  - 352 x 288
#define PAL_176x144        5            ///< #5# - PAL  - 176 x 144
#define PAL_176x120        6            ///< #6# - PAL  - 176 x 144
#define PAL_640x480        192          ///< #7# - NTSC - 160 x 112

#define NTSC_640x480  64            ///< #8# - NTSC - 160 x 112
#define NTSC_1280x720 65            ///< #9# - NTSC - 1280 x 720
#define NTSC_1280x900 66            ///< #10# - NTSC - 1280 x 960
#define NTSC_1280x102467            ///< #11# - NTSC - 1280 x 1024
#define NTSC_1920x108068            ///< #12# - NTSC - 1920 x 1080
```

RS232 Data Refresh Notify

```
typedef struct structural_NOTIFY_RS232DATA_REFRESH
{
    HANDLE   RS232DataRefreshEvent;          // Event handle
    void*    pBuffer;                        // [OUT] Buffer pointer
    int      nFillLength;                    // [IN/OUT] Buffer length
}NOTIFY_RS232DATA_REFRESH;
```

Digital Input Default Value

```
#define DI_DEFAULT_IS_LOW      0x00          // Digital Input Default is Low
#define DI_DEFAULT_IS_HIGH     0x03          // Digital Input Default is High
```

Digital Output Value

```
#define DO_OUTPUT_1            0x01          // Digital Output 1st
#define DO_OUTPUT_2            0x02          // Digital Output 2nd
#define DO_OUTPUT_BOTH         0x03          // Digital Output Both 1st & 2nd
#define DO_OUTPUT_CLEAN        0x00          // Clen up Digital Output
```

RS232 Setting

```
#define RS232_SET_N81          0x00          // RS232 Setting, N,    8, 1
#define RS232_SET_O81          0x08          // RS232 Setting, Odd,  8, 1
```

```
#define RS232_SET_E81          0x18         // RS232 Setting, Even, 8, 1
```

Play Rate

```
enum PLAY_RATES                             // Play rate
{
    RATE_0_5,                               // 1/2 Speed
    RATE_1_0,                               // 1.0 Speed
    RATE_2_0,                               // 2.0 Speed
    RATE_4_0,                               // 4.0 Speed
    RATE_8_0                                // 8.0 Speed
};
```

Contact Type

```
enum CONTACT_TYPE                           // Contact Type
{
    CONTACT_TYPE_UNUSE,                     // not used
    CONTACT_TYPE_UNICAST_WOC_PREVIEW,       // Preview - Uni-cast without control
                                            // port, using ATCP10 and ATCP20
    CONTACT_TYPE_MULTICAST_WOC_PREVIEW,     // Preview - Multicast without control
                                            // port, using AMCST10 and AMCST20
    CONTACT_TYPE_RTSP_PREVIEW,              // Preview - RTSP , using ARTSP( not
                                            // release yet )
    CONTACT_TYPE_CONTROL_ONLY,              // Control only - using ATCP10 and
                                            // ATCP20
    CONTACT_TYPE_UNICAST_PREVIEW,           // Preview - Uni-cast , using ATCP10
                                            // and ATCP20
    CONTACT_TYPE_MULTICAST_PREVIEW,         // Preview - Multicast, using AMCST10
                                            // and AMCST20
    CONTACT_TYPE_PLAYBACK,                  // Playback - Playback, using ARAW
    CONTACT_TYPE_CARD_PREVIEW               // Preview - 4100 preview, using A4100
};
```

RS232 Baud Rate

```
enum RS232_BAUD_RATE                        // RS232 BaudRate
{
    BAUD_RATE_1200BPS,                      // 1200 BPS
    BAUD_RATE_2400BPS,                      // 2400 BPS
    BAUD_RATE_4800BPS,                      // 4800 BPS
    BAUD_RATE_9600BPS,                      // 9600 BPS
    BAUD_RATE_19200BPS,                     // 19200 BPS
    BAUD_RATE_38400BPS,                     // 38400 BPS
    BAUD_RATE_57600BPS,                     // 57600 BPS
    BAUD_RATE_115200BPS,                    // 115200 BPS
    BAUD_RATE_230400BPS                     // 230400 BPS
};
```

Bit Rate

```
enum BITRATE_TYPES                          // Bitrate Types
{
    BITRATE_28K,                            // 28K Bits per second
    BITRATE_56K,                            // 56K Bits per second
    BITRATE_128K,                           // 128K Bits per second
    BITRATE_256K,                           // 256K Bits per second
    BITRATE_384K,                           // 384K Bits per second
    BITRATE_500K,                           // 500K Bits per second
    BITRATE_750K,                           // 750K Bits per second
    BITRATE_1000K,                          // 1M Bits per second
    BITRATE_1200K,                          // 1.2M Bits per second
    BITRATE_1500K,                          // 1.5M Bits per second
    BITRATE_2000K,                          // 2M Bits per second
    BITRATE_2500K,                          // 2.5M Bits per second
    BITRATE_3000K                           // 3M Bits per second
};
```

Codec Type

```
enum CODEC_TYPES                            // CODEC Types
{
    XVIDCODEC,                              // XVID - using XVIDCODEC
    FFMCODEC,                               // FFMpeg - using FFMCODEC
    P51CODEC                                // PCI5100 - using P51CODEC
    IPPCODEC                        ///< #3# - IPPCODEC - using IPPCODEC
};
```

File Write Type

```
enum FILE_WRITER_TYPES                      // File writer types
{
    FRAW,                                   // Record by *.Raw File - using FRAW
    FAVI                                    // Record by *.Avi File - using FAVI
};
```

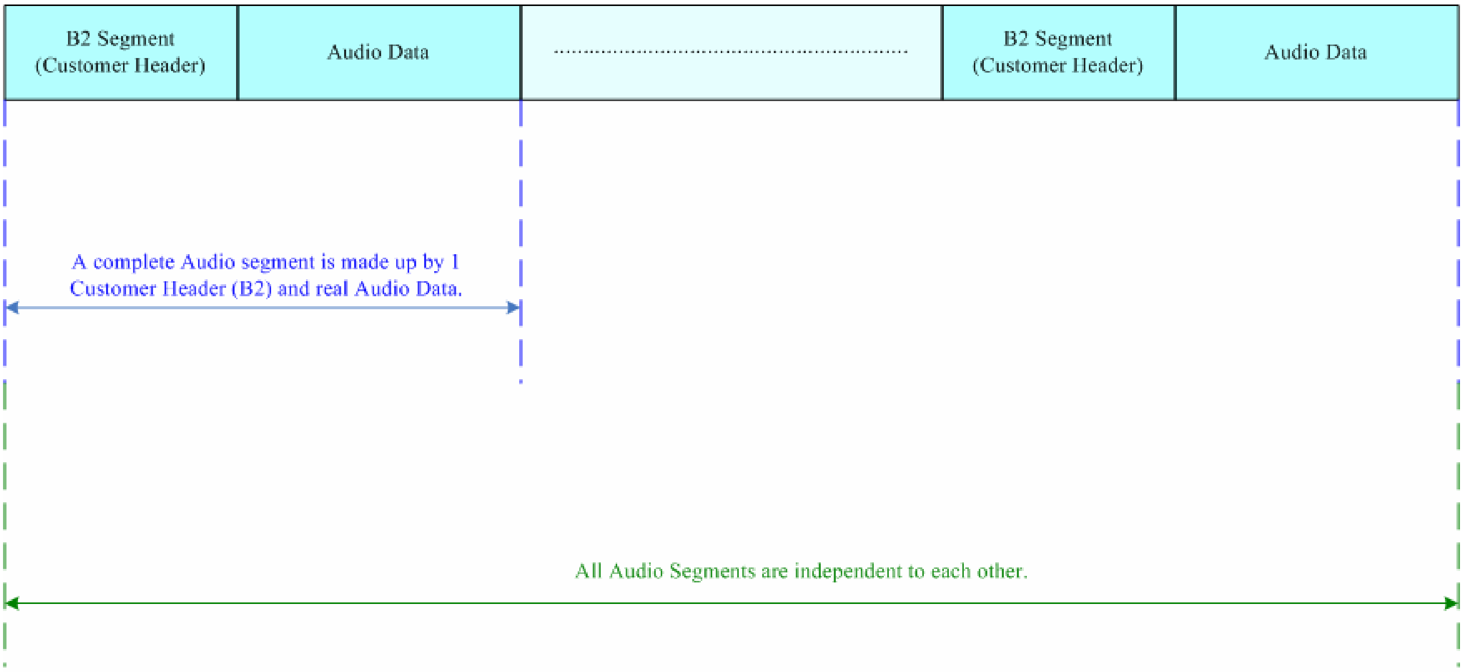
Render Type

```
enum RENDER_TYPES                           // Render interface types
{
    DGDI,                                   // Windows GDI for render
    DDRAW                                   // Direct Draw for render
};
```

Device Type

```
enum DEVICE_TYPE                                       // Device Type
{
    Type_None,                                         // None type
    Type_StandAlong,                                   // Stand along
    Type_RackMount,                                    // Rack Mount
    Type_Blade                                         // Blade
};
```

TCP v2.0 Audio Packet Format

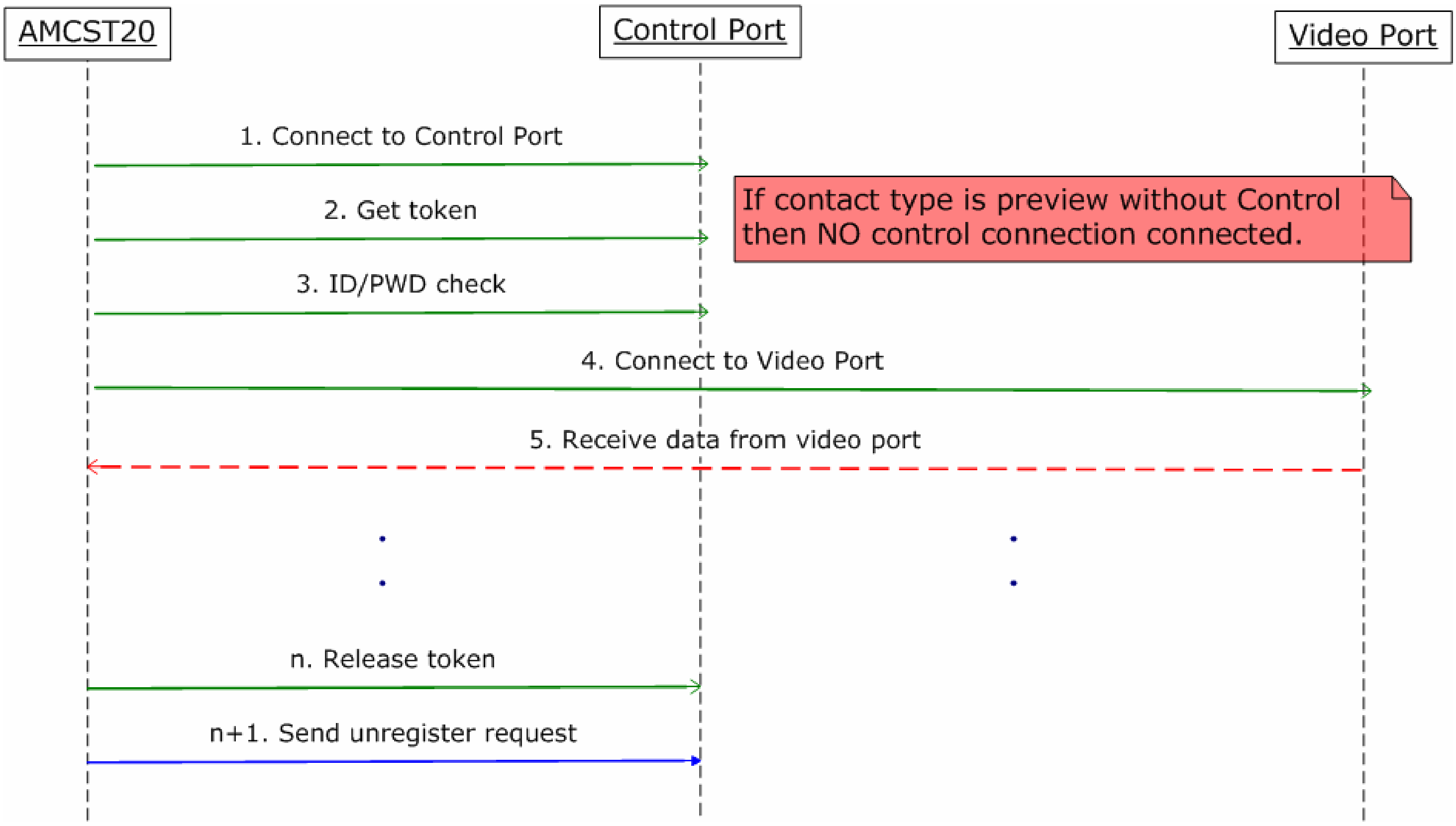


Note. Video server will send Audio & Video data in random order.

```
typedef struct {
     B2_HEADER header;
     struct timeval timestamp;
     unsigned char reserved[8];
} AUDIO_B2;    // 28 bytes (details in chapter 8)
```

Multicast v2.0 Packet

Multicast v2.0 Video Connect Flow


AMCST20
Control Port
Video Port
1. Connect to Control Port
2. Get token
If contact type is preview without Control then NO control connection connected.
3. ID/PWD check
4. Connect to Video Port
5. Receive data from video port
n. Release token
n+1. Send unregister request

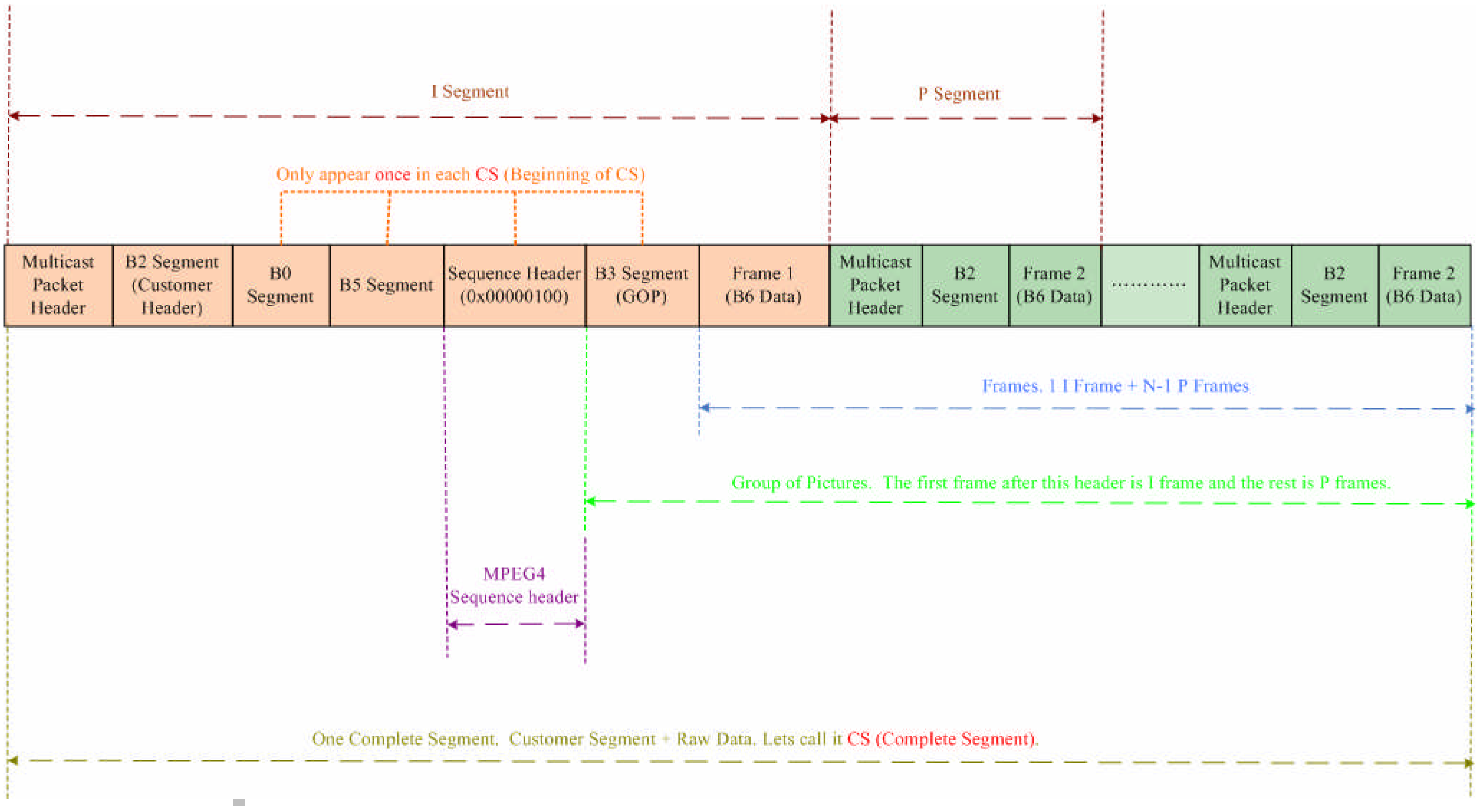
Disconnect steps

1. Do n.

2. Disconnect control port

3. Disconnect video port

Multicast v2.0 Video Packet Format


I Segment
P Segment
Only appear once in each CS (Beginning of CS)
Multicast Packet Header
B2 Segment (Customer Header)
B0 Segment
B5 Segment
Sequence Header (0x00000100)
B3 Segment (GOP)
Frame 1 (B6 Data)
Multicast Packet Header
B2 Segment
Frame 2 (B6 Data)
…………
Multicast Packet Header
B2 Segment
Frame 2 (B6 Data)
Frames. 1 I Frame + N-1 P Frames
Group of Pictures. The first frame after this header is I frame and the rest is P frames.
MPEG4 Sequence header
One Complete Segment. Customer Segment + Raw Data. Lets call it CS (Complete Segment).

```
typedef struct tagMCPacketHead
{
unsigned char StreamId;
unsigned char StreamSubId;
unsigned char KeyPacket;
unsigned char TotalPacket;
unsigned char PacketNum;
unsigned char FrameCheckSum;
unsigned char Resolution;
unsigned char Fps;
unsigned int  FrameNum;
unsigned int  FrameLen;
} MCPacketHead;
```

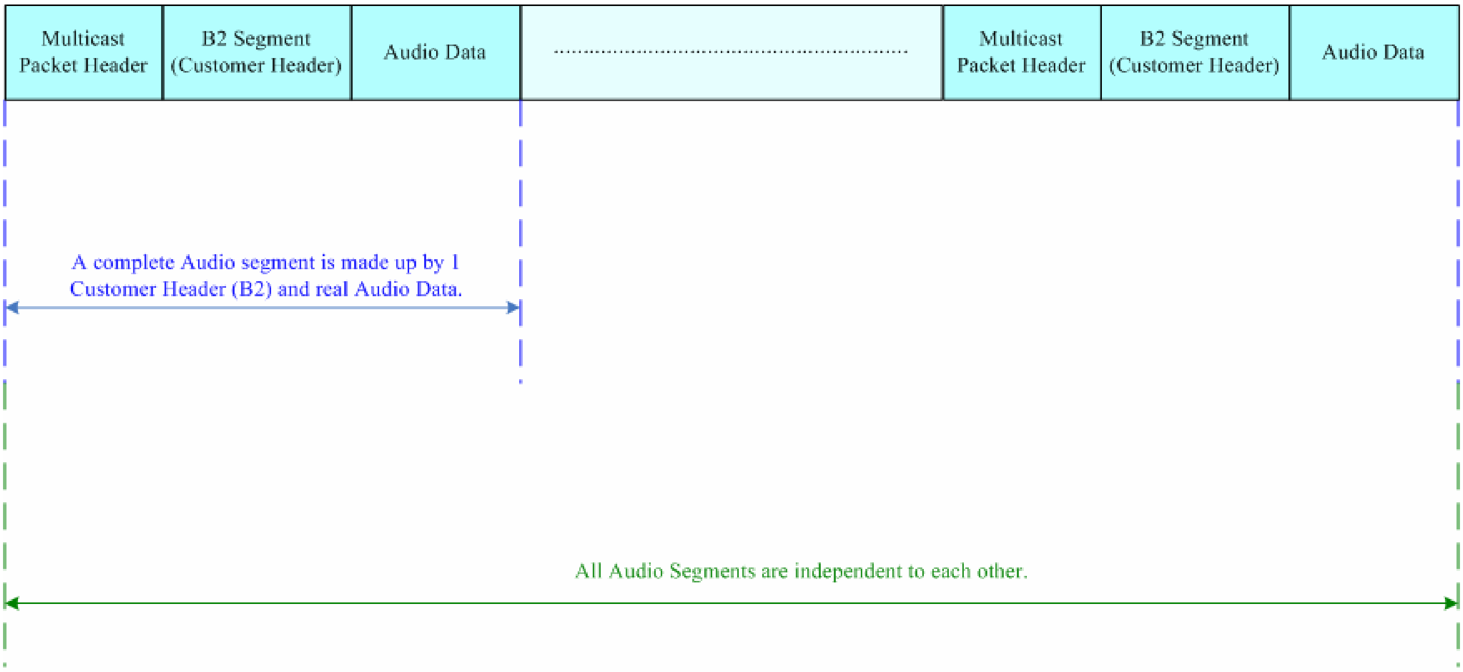
Important Note.

1. Key packet attribute is very important to determine the last packet of the frame.

2. Need to find out Resolution and FPS from Sequence Header

3. 1(I or P frame) frame may divide into several multicast packets, each with a multicast packet header in front of it.

Multicast v2.0 Audio Packet Format


Multicast Packet Header
B2 Segment (Customer Header)
Audio Data
....
Multicast Packet Header
B2 Segment (Customer Header)
Audio Data
A complete Audio segment is made up by 1 Customer Header (B2) and real Audio Data.
All Audio Segments are independent to each other.

Note. Video server will send Audio & Video data in random order.

```
typedef struct tagMCPacketHead
{
unsigned char StreamId;
unsigned char StreamSubId;
unsigned char KeyPacket;
unsigned char TotalPacket;
unsigned char PacketNum;
unsigned char FrameCheckSum;
unsigned char Resolution;
unsigned char Fps;
unsigned int  FrameNum;
unsigned int  FrameLen;
} MCPacketHead;
```

RTP Packet Format

Note that RTP/RTSP protocol is implemented in TCP v2.0 compliant devices

RTP Interface

SDP description :

```
v=0
o=- 1072886400760000 1 IN IP4 192.168.1.100
s=LIVE.COM Session streamed by a GO7007SB WISchip
i=LIVE.COM Streaming Media v
t=0 0
a=tool:LIVE.COM Streaming Media v2004.12.28
a=type:broadcast
a=control:*
a=range:npt=0-
a=x-qt-text-nam:LIVE.COM Session streamed by a GO7007SB WISchip
a=x-qt-text-inf:LIVE.COM Streaming Media v
m=video 0 RTP/AVP 96
c=IN IP4 0.0.0.0
a=rtpmap:96 MP4V-ES/90000
a=fmtp:96 
profile-level-id=245;config=000001B0F5000001B509000001000000012000C888
BAA760FA62D087828307
a=control:track1
m=audio 0 RTP/AVP 111
c=IN IP4 0.0.0.0
a=rtpmap:111 L16/8000
a=control:track2
```

RTSP request command :

[OPTIONS request]

```
rtsp://192.168.1.254:7070/ RTSP/1.0
CSeq: 1
User-Agent: VLC Media Player (LIVE.COM Streaming Media v2004.11.11)
```

[OPTIONS response]

```
sending response: RTSP/1.0 200 OK
CSeq: 1
Public: OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE
```

[DESCRIBE request]

```
DESCRIBE rtsp://192.168.1:100:7070 RTSP/1.0
CSeq: 1
Accept: application/sdp
Bandwidth: 384000
Accept-Language: en-GB
User-Agent: QuickTime/7.0.3 (qtver=7.0.3;os=Windows NT 5.1Service Pack 1)
```

[DESCRIBE response]

```
sending response: RTSP/1.0 200 OK
CSeq: 1
Date: Fri, Dec 02 2005 06:38:53 GMT
Content-Base: rtsp://192.168.1.100:7070//
Content-Type: application/sdp
Content-Length: 608

v=0
o=- 1133505497174429 1 IN IP4 192.168.1.100
s=LIVE.COM Session streamed by a GO7007SB WISchip
i=LIVE.COM Streaming Media v
t=0 0
a=tool:LIVE.COM Streaming Media v2004.12.28
a=type:broadcast
a=control:*
```

```
a=range:npt=0-
a=x-qt-text-nam:LIVE.COM Session streamed by a GO7007SB WISchip
a=x-qt-text-inf:LIVE.COM Streaming Media v
m=video 0 RTP/AVP 96
c=IN IP4 0.0.0.0
a=rtpmap:96 MP4V-ES/90000
a=fmtp:96 
profile-level-id=245;config=000001B0F5000001B509000001000000012000C888BAA760FA
62D087828307
a=control:track1
m=audio 0 RTP/AVP 111
c=IN IP4 0.0.0.0
a=rtpmap:111 L16/8000
a=control:track2
```

[SETUP request]

```
SETUP rtsp://192.168.1.100:7070//track1 RTSP/1.0
CSeq: 2
Transport: RTP/AVP;unicast;client_port=6970-6971
x-retransmit: our-retransmit
x-dynamic-rate: 1
x-transport-options: late-tolerance=2.900000
User-Agent: QuickTime/7.0.3 (qtver=7.0.3;os=Windows NT 5.1Service Pack 1)
Accept-Language: en-GB
```

[SETUP response]

```
sending response: RTSP/1.0 200 OK
CSeq: 2
Date: Fri, Dec 02 2005 06:38:54 GMT
Transport: 
RTP/AVP;unicast;destination=192.168.1.3;client_port=6970-6971;server_port=1024
-1025
Session: 1
```

[SETUP request]

```
rtsp://192.168.1.100:7070//track2 RTSP/1.0
CSeq: 3
Transport: RTP/AVP;unicast;client_port=6972-6973
x-retransmit: our-retransmit
x-dynamic-rate: 1
```

```
x-transport-options: late-tolerance=2.900000
Session: 1
User-Agent: QuickTime/7.0.3 (qtver=7.0.3;os=Windows NT 5.1Service Pack 1)
Accept-Language: en-GB
```

[SETUP response]

```
sending response: RTSP/1.0 200 OK
CSeq: 3
Date: Fri, Dec 02 2005 06:38:54 GMT
Transport: 
RTP/AVP;unicast;destination=192.168.1.3;client_port=6972-6973;server_port=1026
-1027
Session: 1
```

[PLAY request]

```
rtsp://192.168.1.100:7070 RTSP/1.0
CSeq: 4
Range: npt=0.000000-
x-prebuffer: maxtime=2.000000
Session: 1
User-Agent: QuickTime/7.0.3 (qtver=7.0.3;os=Windows NT 5.1Service Pack 1)
```

[PLAY response]

```
sending response: RTSP/1.0 200 OK
CSeq: 4
Date: Fri, Dec 02 2005 06:38:54 GMT
Range: npt=0.000-
Session: 1
RTP-Info: 
url=rtsp://192.168.1.100:7070//track1;seq=64955,url=rtsp://192.168.1.100:7070/
/track2;seq=39531
```

[PAUSE request]

```
PAUSE rtsp://192.168.1.100:7070 RTSP/1.0
CSeq: 5
Session: 1
```

```
User-Agent: QuickTime/7.0.3 (qtver=7.0.3;os=Windows NT 5.1Service Pack 1)
```

[PAUSE response]

```
sending response: RTSP/1.0 200 OK
CSeq: 5
Date: Fri, Dec 02 2005 06:39:36 GMT
Session: 1
```

[TEARDOWN request]

```
rtsp://192.168.1.100:7070 RTSP/1.0
CSeq: 6
Session: 1
User-Agent: QuickTime/7.0.3 (qtver=7.0.3;os=Windows NT 5.1Service Pack 1)
```

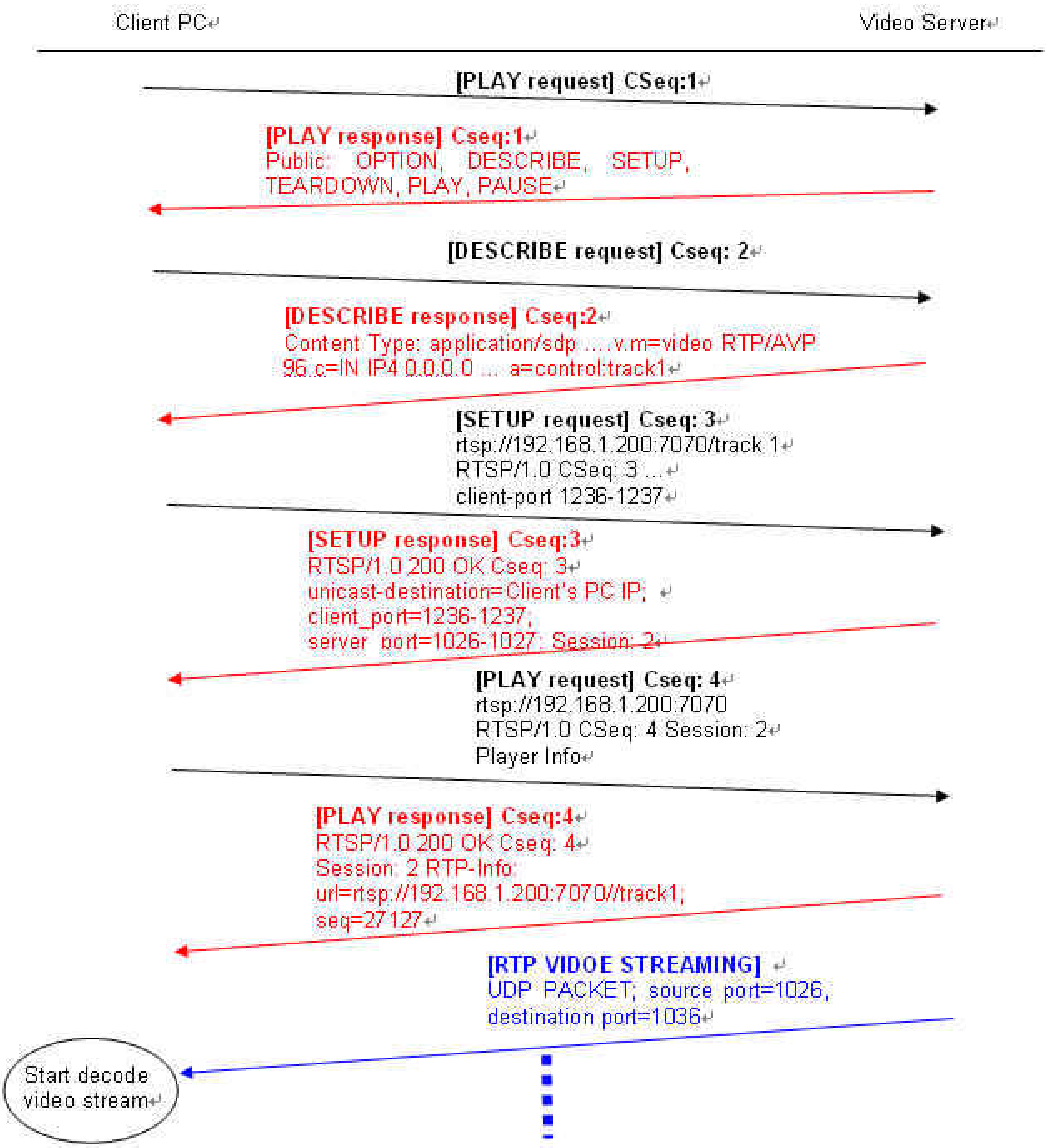
[TEARDOWN response]

```
sending response: RTSP/1.0 200 OK
CSeq: 6
Date: Fri, Dec 02 2005 06:39:36 GMT
```

Play an unicast RTP video stream (TRACK 1), while play an unicast audio stream (TRACK 2)

RTP Protocol Flow

Establishment



Client PC

Video Server

[PLAY request] CSeq:1

[PLAY response] Cseq:1
Public: OPTION, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE

[DESCRIBE request] Cseq: 2

[DESCRIBE response] Cseq:2
Content Type: application/sdp ....v.m=video RTP/AVP 96 c=IN IP4 0.0.0.0 ... a=control:track1

[SETUP request] Cseq: 3
rtsp://192.168.1.200:7070/track 1
RTSP/1.0 CSeq: 3 ...
client-port 1236-1237

[SETUP response] Cseq:3
RTSP/1.0 200 OK Cseq: 3
unicast-destination=Client's PC IP;
client_port=1236-1237;
server_port=1026-1027; Session: 2

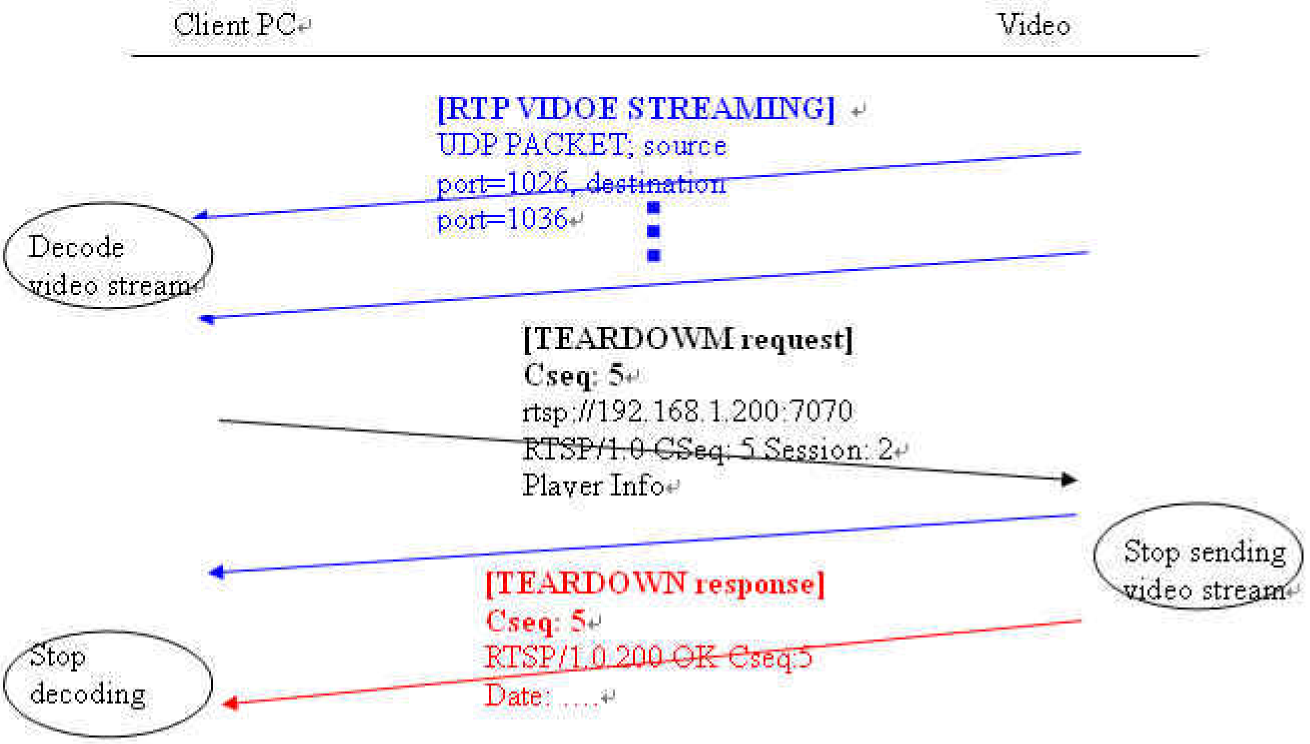
[PLAY request] Cseq: 4
rtsp://192.168.1.200:7070
RTSP/1.0 CSeq: 4 Session: 2
Player Info

[PLAY response] Cseq:4
RTSP/1.0 200 OK Cseq: 4
Session: 2 RTP-Info:
url=rtsp://192.168.1.200:7070//track1;
seq=27127

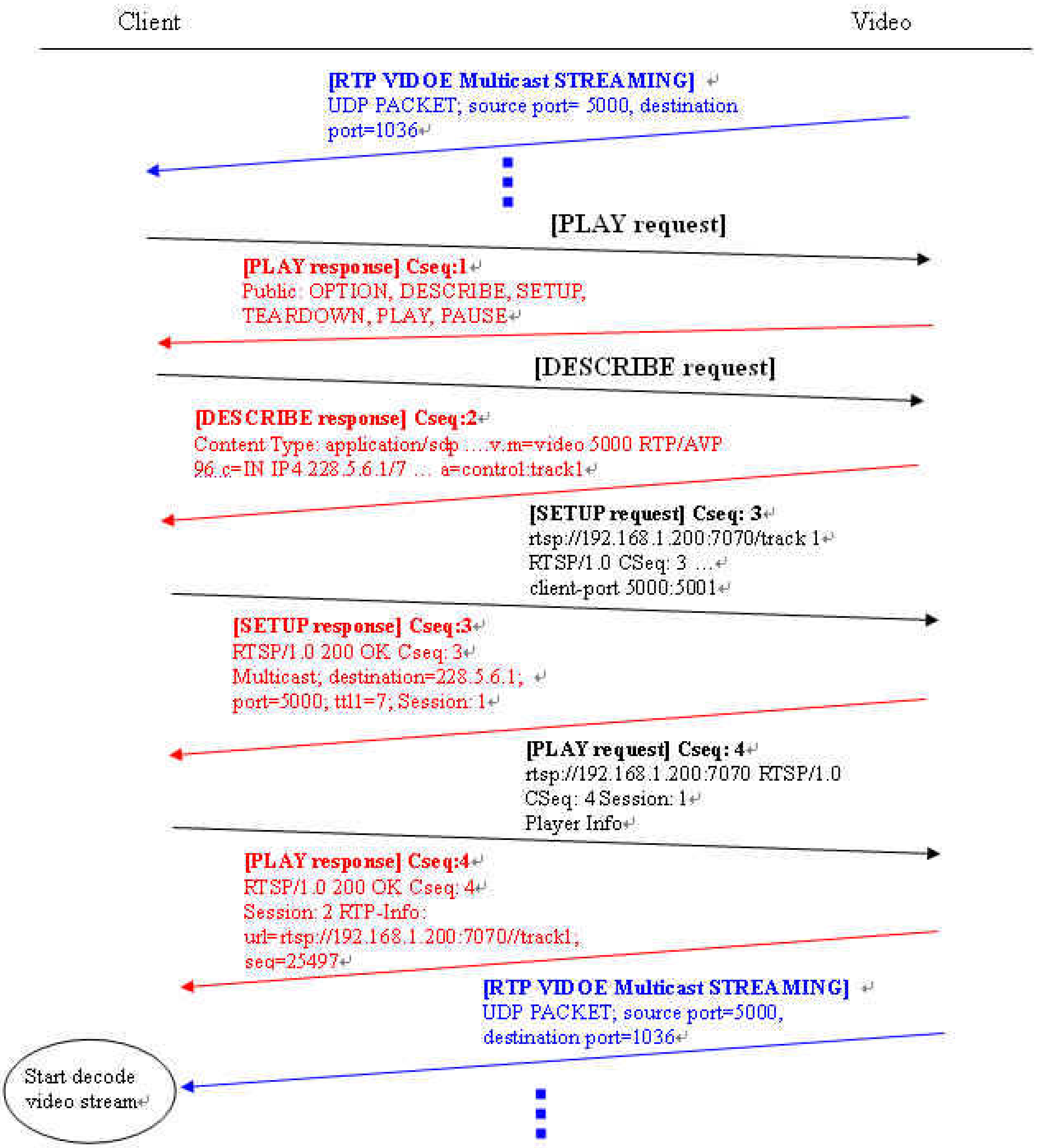
[RTP VIDOE STREAMING]
UDP PACKET; source port=1026,
destination port=1036

Start decode video stream

TEARDOWN An Unicast RTP VIDEO STREAM



PLAY A Multicast RTP VIDEO STREAM (TRACK 1):



[RTP VIDOE Multicast STREAMING]
UDP PACKET; source port= 5000, destination port=1036

[PLAY request]

[PLAY response] Cseq:1
Public: OPTION, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE

[DESCRIBE request]

[DESCRIBE response] Cseq:2
Content Type: application/sdp ….v.m=video 5000 RTP/AVP 96.c=IN IP4 228.5.6.1/7 … a=control:track1

[SETUP request] Cseq: 3
rtsp://192.168.1.200:7070/track 1
RTSP/1.0 CSeq: 3 …
client-port 5000:5001

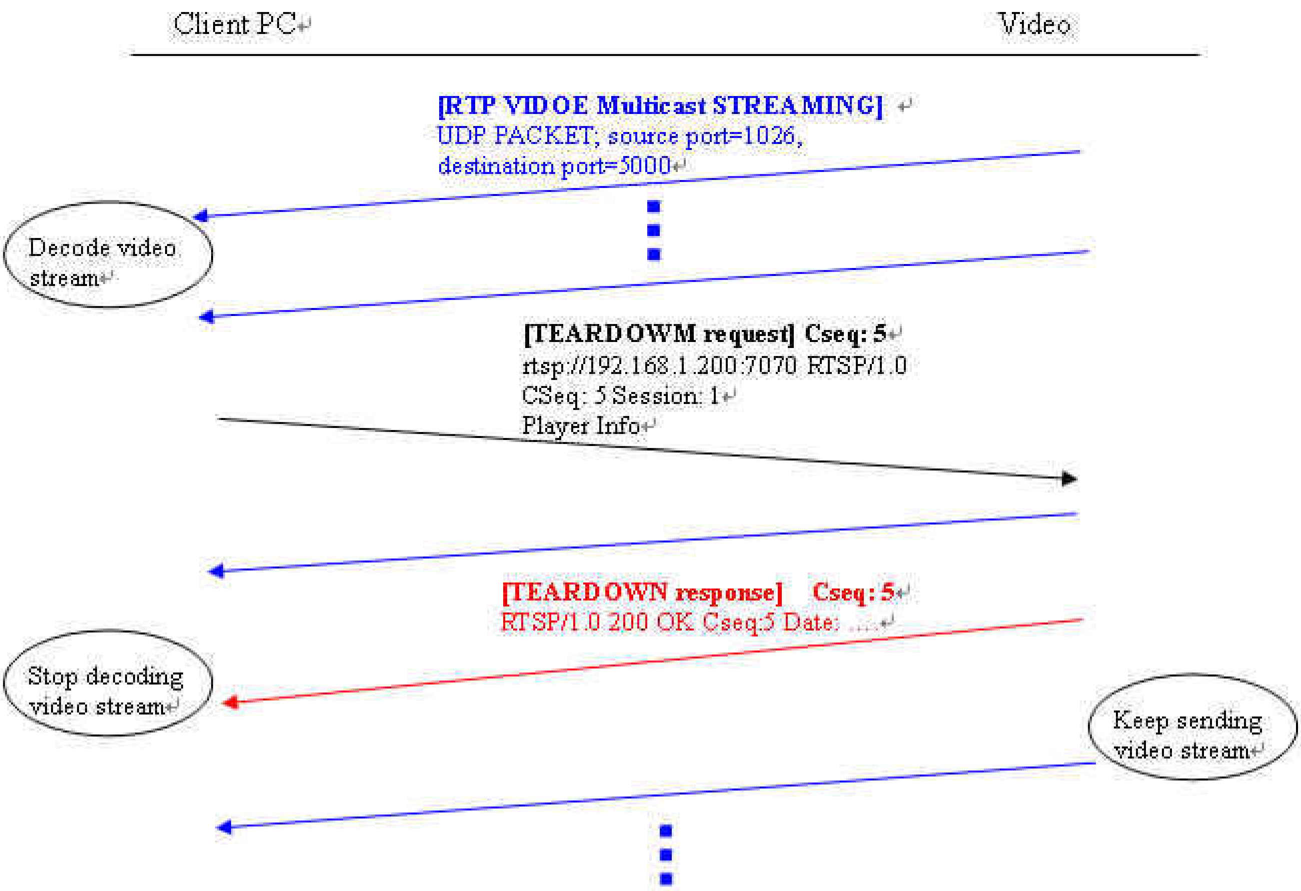
[SETUP response] Cseq:3
RTSP/1.0 200 OK Cseq: 3
Multicast; destination=228.5.6.1;
port=5000; ttl1=7; Session: 1

[PLAY request] Cseq: 4
rtsp://192.168.1.200:7070 RTSP/1.0
CSeq: 4 Session: 1
Player Info

[PLAY response] Cseq:4
RTSP/1.0 200 OK Cseq: 4
Session: 2 RTP-Info:
url=rtsp://192.168.1.200:7070//track1,
seq=25497

[RTP VIDOE Multicast STREAMING]
UDP PACKET; source port=5000, destination port=1036

TEARDOWN A Multicast RTP VIDEO STREAM


Client PC
Video
[RTP VIDOE Multicast STREAMING]
UDP PACKET; source port=1026,
destination port=5000
Decode video
stream
[TEARDOWM request] Cseq: 5
rtsp://192.168.1.200:7070 RTSP/1.0
CSeq: 5 Session: 1
Player Info
[TEARDOWN response] Cseq: 5
RTSP/1.0 200 OK Cseq:5 Date: ....
Stop decoding
video stream
Keep sending
video stream

10 Migration Plan from SDK-2000 to SDK-10000

SDK-10000 New Features

SDK-10000 v1.0 series contains new design architecture with following features:

- Unified SDK for IP devices (IP Camera, Video Server, IP Speed Dome, Quad Video Server), Capture Cards, Decoder Cards, Streaming Engine and File Playback. One programming can fit all above devices.
- Superset of SDK-2000, SDK-4000 and SDK-5000
- Scalable architecture: new adaptor can be added without chaning codes
- Better performance: SDK-10000 has better performance and memory management over previous SDK. It also provide shorter video latency than previous SDK
- New adaptors: Direct Draw and FAVI (record to AVI file) adaptors provided
- Multi-channel Support: Supports multiple channel devices, 2/4/8-channel video server, 4-channel capture card

SDK-2000 vs SDK-10000 Function Calls

SDK – 2000	SDK – 10000	Remark
netGetTCPMode	KGetTCPTypeByHTTP	
netOpenInterface	KOpenInterface	
netRegisterServer	KSetMediaConfig KConnect	
netInitStream	KSetRenderInfo	
netStartStream	KStartStreaming KPlay	
netSetStatusCallBack	KSetVideoLossCallback KSetVideoRecoveryCallback KSetNetworkLossCallback	
netSetMDCallBack	KSetMotionDetectionCallback	
netSetDIDefault	KSetDIDefaultValue	
netSetDIOCallBack	KSetDICallback	
netSetTimeCodeCallBack	KSetTimeCodeCallback	
netSetAfterFlushCallBack		Not support in SDK-10000
netSetAfterRenderCallBack	KSetAfterRenderCallback	
netSetImageCallBack	KSetImageCallback	
netSetRS232CallBack	KSetRS232DataCallback	
netSetServerSerialDataCallBack	KSetRS232DataCallback	
netUnRegisterServer	KDisconnect	
netGetServerConfig	KGetVideoConfig	
netSetServerConfig	KSetVideoConfig	
netStopStream	KStopStreaming	
netSetAutoFrameRate		Not support in SDK-10000
netSetAlarmPreRecordingT	KSetPrerecordTime	

ime		
netStartAlarmRecord	KStartRecord	
netStopAlarmRecord	KStopRecord	
netStopAlarmRecord2	KStopRecord	
netStartRecord	KStartRecord	
netStopRecord	KStopRecord	
netStopRecord2	KStopRecord	
netSend2ServerSerialPort	KSendRS232Command	
netSendKeyPadCommand	KSendPTZCommand	
netSendDIO	KSendDO	
netSetMotionRange	KSetMotionInfo	
netSetMotionSensitive	KGetMotionInfo KSetMotionInfo	
netGetLastError	KGetLastError	
netGetFrameReceived	KGetTotalReceiveVideoFrameCount	
netGetDataReceived	KGetTotalReceiveSize	
netGetDispWindowPos		Not support in SDK-10000
netSetDispWindowPos	KSetRenderInfo	
netSetRS232	KSendRS232Setting	
netSetServerSerialPort	KSendRS232Setting	
netSearchServer	KSearchServer	
netGetDioStatus	KGetDIDefaultValueByHTTP	
netGetMotionSetting	KGetMotionInfo	
netSetMpeg4RawCallBack	KSetRawDataCallback	
netGetOnLineUser	KGetOnLineUser	
netGetSDKVersion	KGetVersion	
netGetServerVersion	KGetServerVersion	
netRegisterServerEx	KSetMediaConfig KConnect	
netSetCommunicationPort		Not support in SDK-10000

netDecodeI	KSetDecodeIFrameOnly	
netSendURL	KSendURLCommand	
netSendCMD		Not support in SDK-10000
netCloseInterface	KCloseInterface	
netGetCameraName	KGetCameraName	
netSaveReBoot	KSaveReboot	
netGetControlToken		Not support in SDK-10000
netGetAudioToken	KGetAudioToken	
netFreeControlToken		Not support in SDK-10000
netGiveOffSound		Not support in SDK-10000
netCloseSound	KStopAudioTransfer	
netFreeAudioToken	KFreeAudioToken	
netIsMute		Not support in SDK-10000
netSetVolume	KSetVolume	
netGetVolume	KGetVolume	
netSetPreviewBuffer		Not support in SDK-10000
netSendAudio	KStartAudioTransfer	
netSetMpeg4RawCallBack2	KSetRawDataCallback	
netSetAudioRawCallBack	KSetRawDataCallback	
netSetStreamRawCallBack	KSetRawDataCallback	
netSend2StreamEngine	KSendCommandToStreamingEngine	
netMute	KSetMute	
netSetSocketSize		Not support in SDK-10000
netRegisterServerControlOnly	KSetMediaConfig KConnect	Set Contact type to CONTACT_TYPE_CONTROL_ONLY
netStartWriteInfo		Not support in SDK-10000
netStopWriteInfo		Not support in SDK-10000
netGetDeviceType	KGetDeviceTypeByHTTP	
netSetHTTPPort		Not support in SDK-10000

netSetResolutionChangeCallBack	KSetResolutionChangeCallback	
netSetChannelNumber		Not support in SDK-10000
netSetConnectTimeOut		Not support in SDK-10000

Application Migration Guide

This section describes the steps for customers to port their application from SDK-2000 to SDK-10000.

We provides 2 different step-by-step guides for following applications:

- Application that uses MPEG-4 raw data only
- Application that uses most function calls

Application that uses MPEG-4 raw data only

Steps to migrate from SDK-2000 to SDK-10000:

1. Re-compile the source codes with SDK-10000
2. Use `KGetTCPTypeByHTTP()` to detect if the device is compatible to TCP 1.0 format or TCP 2.0 (supports audio) format
3. Use `KSetRawDataCallback()` to receive both MPEG4-Video and MPEG4-Audio data.
4. Use `KSetImageCallback()` to get RGB buffer at the same time
5. Call `KSendPTZCommand()` to send PTZ commands.
6. Note that in SDK-10000, every I-Frame contains sequence header in the frame
7. Refer to Audio API for 1-way or 2-way audio functions
8. Refer to MPEG-4 data structure section for detailed MPEG-4 audio + video format

Application that uses most function calls

Steps to migrate from SDK-2000 to SDK-10000:

1. Re-compile the source codes with SDK-10000
2. Use `KGetTCPTypeByHTTP()` to detect if the device is compatible to TCP 1.0 format or TCP 2.0 (supports audio) format
3. Use `KSetDecodeIFrameOnly()` function to decode I-Frame only to save CPU utilization; this will only affect on the decoding part, recording can still record with specified frame rate
4. Call `KSendPTZCommand()` to send PTZ commands.
5. Refer to Audio API for 1-way or 2-way audio functions.

11 URL Command Index

MPEG4 Category

This category lists the commands that is related to MPEG-4 settings.

The syntax of the command is listed as follow:

```
http://<IP Address>/cgi-bin/mpeg4?USER=<Account Name>&PWD=<Password>&<Parameters>
```

or

```
http://<Account Name>:<Password>@<IP Address>/cgi-bin/cmd/mpeg4?<Parameters>
```

The notation of the value inside is listed as follow:

R: Read

W: Write

*: On the fly change. Does not need to execute save and reboot to the firmware; all other parameters without * mark need to run save and reboot to the firmware to take effect.

G: Global setting, meaning that when user sets the value for Global setting, then all channels in the sub-unit are applied automatically

--: Not supported

C: Individual channels under a multi-channel device

<RED Color>: Indicates that the setting of 2-channel device is different from that of 8-channel devices

MPEG4					
Parameter	**Value**	**Format**	**1-CH**	**2-CH**	**8-CH**
VIDEO_BRIGHTNESS	0~100	<value>	RW*	G:--,C:RW*	G:--,C:RW*
VIDEO_CONTRAST	0~100	<value>	RW*	G:--,C:RW*	G:--,C:RW*
VIDEO_SATURATION	0~100	<value>	RW*	G:--,C:RW*	G:--,C:RW*
VIDEO_HUE	0~100	<value>	RW*	G:--,C:RW*	G:--,C:RW*
VIDEO_RESOLUTION	NTSC: N720x480/N352x240/N160x112 PAL : P720x576/P352x288/P176x144	<string>	RW*	G:--,C:RW*	G:--,C:RW*
VIDEO_BITRATE	28K/56K/128K/256K/384K/500K/750K/ 1M/1.2M/1.5M/2M/2.5M/3M	<string>	RW*	G:--,C:RW*	G:--,C:RW*
VIDEO_FPS_NUM	1/2/3/4/5/6/7/10/15/30 for NTSC 1/2/3/4/5/6/8/12/25 for PAL	<value>	RW*	G:--,C:RW*	G:--,C:RW*
VIDEO_CAMERA_NAME	Max sizes: 20 bytes	<string>	RW	G:--,C:RW	G:--,C:RW
VIDEO_STATUS	VIDEO_BRIGHTNESS='43' VIDEO_CONTRAST='50' VIDEO_SATURATION='58' VIDEO_HUE='50' VIDEO_RESOLUTION='N720x480' VIDEO_BITRATE='1.2M' VIDEO_FPS='MODE1' VIDEO_FPS_NUM='30' VIDEO_CAMERA_NAME='Camera-1'	text/plain	RO*	G:--,C:RO*	G:--,C:RO*
DIO_OUTPUT	0x00: DO1 LOW, DO2 LOW 0x01: DO1 HI, DO2 LOW 0x02: DO1 LOW, DO2 HI 0x03: DO1 HI, DO2 HI	<hex>	WO*	G:WO*,C:--	G:--,C:RW*
DIO_STATUS	BIT0: DI1 status BIT1: DI2 status BIT2: Reserved BIT3: Reserved BIT4: DO1 status BIT5: DO2 status BIT6: Reserved BIT7: Reserved	text/plain	RO*	G:RO*,C:--	G:--,C:RW*
MOTION_ENABLED	0x00: Disabled 0x01: Enabled	<hex>	RW*	G:--,C:RW*	G:--,C:RW*
MOTION_SETTING	window: 1~3	<window><x_u	WO*	G:--,C:WO*	G:--,C:WO*

	x_upper: NTSC:0~720/PAL:0~720 y_upper: NTSC:0~720/PAL:0~720 x_bottom: NTSC:0~480/PAL:0~576 y_bottom: NTSC:0~480/PAL:0~576 sensitive: 0~100	pper>,<y_upper>,<x_bottom>,<y_bottom><sensitive>			
MOTION_SENSITIVE	window: 1~3 sensitive: 0~100	<window>,<sensitive>	WO*	G:--,C:WO*	G:--,C:WO*
MOTION_STATUS	MOTION_STATUS=1,0,0,0,0,0 MOTION_STATUS=2,0,0,0,0,0 MOTION_STATUS=3,0,0,0,0,0	text/plain	RO*	G:--,C:RO*	G:--,C:RO*
SERIAL_SETTING	line: 8N1/8O1/8E1 baudrate: 2400/4800/9600/19200 38400/57600/115200	<line>,<baudrate>	RW*	G:RW*,C:--	G:--,C:RW*
SERIAL_ASCII	ascii string	<ascii string>	WO*	G:WO*,C:--	G:--,C:WO*
SERIAL_HEX	hex string	<hex string>	WO*	G:WO*,C:--	G:--,C:WO*
VIDEO_VARIABLE_FPS	id: get session id form sdk fps: NTSC: 1/3/6/30 PAL : 1/3/5/25	<id>,<fps>	WO*	G:--,C:WO*	G:--,C:WO*
RTP_MULTICAST_STREAMING	PLAY/PAUSE	<string>	RW*	G:--,C:RW*	G:--,C:RW*

SYSTEM Category

This category lists the commands that is related to system settings.

The syntax of the command is listed as follow:

```
http://<IP Address>/cgi-bin/system?USER=<Account Name>&PWD=<Password>&<Parameters>
```

or

```
http://<Account Name>:<Password>@<IP Address>/cgi-bin/cmd/system?<Parameters>
```

The notation of the value inside is listed as follow:

R: Read

W: Write

*****: On the fly change. Does not need to execute save and reboot to the firmware; all other parameters without * mark need to run save and reboot to the firmware to take effect.

G: Global setting, meaning that when user sets the value for Global setting, then all channels in the sub-unit are applied automatically

--: Not supported

C: Individual channels under a multi-channel device

<RED Color>: Indicates that the setting of 2-channel device is different from that of 8-channel devices

SYSTEM					
Parameter	**Value**	**Format**	**1-CH**	**2-CH**	**8-CH**
SAVE_REBOOT	SAVE_REBOOT OK	text/plain	WO*	G:WO*,C--	G:WO*,C:WO*
SAVE	Save Finish	Test/plain	WO	G:WO,C--	G:WO,C:WO
REBOOT	REBOOT OK	text/plain	WO*	G:WO*,C--	G:WO*,C:WO*
FACTORY_DEFAULT	Factory Default Finish	text/plain	WO	G:WO,C--	G:WO,C:WO
SYSTEM_INFO	Firmware Version = A1D-P2N-V2.00.07-AC MAC Address = 00:0F:7C:00:00:67 Production ID = SED2400-04I-8-00027 Factory Default Type = NTSC, Composite, Two Ways Audio (0x71)	text/plain	RO*	G:RO*,C--	G:RO*,C:RO*
SYSTEM_PROPERTY	SYSTEM='E' TYPE='A' NO_OF_CHANNEL='01' MULTIPLEXING='X' NO_OF_AUDIO_WAYS='2' AUDIO_TYPE='PCM' MOTION_TYPE='0' PROTOCOL_TYPE='2'	text/plain	RO*	G:RO*,C--	G:RO*,C:RO*
LAN_HOSTNAME	Max size: 30bytes	<string>	RW	G:RW,C--	G:RW,C:--
WAN	WAN_TYPE='1' WAN_IP='10.0.0.1' WAN_NETMASK='255.255.255.0' WAN_GATEWAY='10.0.0.254' WAN_PPPOE_USERNAME='' WAN_PPPOE_PASSWORD=	text/plain	RO*	G:RO*,C--	G:RO*,C:--
WAN_TYPE	1: Dynamic IP 2: Static IP 3: PPPoE	<value>	RW	G:RW,C--	G:RW,C:--
WAN_IP	Static ip address	<ip address>	RW	G:RW,C--	G:RW,C:--
WAN_NETMASK	Static netmask ip	<ip address>	RW	G:RW,C--	G:RW,C:--
WAN_GATEWAY	Static gateway ip	<ip address>	RW	G:RW,C--	G:RW,C:--
WAN_PPPOE_USERNAME	Max sizes: 60bytes	<string>	RW	G:RW,C--	G:RW,C:--
WNA_PPPOE_PASSWORD	Max sizes: 60bytes	<string>	RW	G:RW,C--	G:RW,C:--
WAN_STATUS	IP Address : 172.16.3.15 Netmask : 255.255.255.0	text/plain	RO*	G:RO*,C--	G:RO*,C:--

	Gateway : 172.16.3.253 DNS Server : 172.16.5.19 172.16.5.22 DNS Host : WAN Connect Status : Connect DNS Connect Status : Connect DDNS Connect Status : Disconnect				
V2_WAN_STATUS	WAN_TYPE=`1` WAN_IP='172.16.3.27' WAN_NETMASK='255.255.255.0' WAN_GATEWAY='172.16.3.253'	text/plain	RO*	G:RO*,C:--	G:RO*,C:--
DNS_PRIMARY	Primary domain name server	<ip address>	RW	G:RW,C:--	G:RW,C:--
DNS_SECONDARY	Secondary domain name server	<ip address>	RW	G:RW,C:--	G:RW,C:--
DNS	DNS_PRIMARY='' DNS_SECONDARY=''	text/plain	RO*	G:RO*,C--	G:RO*,C:--
DDNS_TYPE	1: Disabled 2: Enabled	<value>	RW	G:RW,C--	G:RW,C:--
DDNS_HOSTNAME		<string>	RW	G:RW,C--	G:RW,C:--
DDNS_SERVICE	dyndns: members.dyndns.org qdns: members.3322.org ezip: www.EZ-IP.Net pgpow: www.penguinpowered.com dhs: members.fhs.org ods: update.ods.com tzo: cgi.tzo.com easydns: members.easydns.com justlinux: www.justlinux.com dyns: www.dyns.cx hn: www.hn.org zoneedit: www.zoneedit.com	<string>	RW	G:RW,C--	G:RW,C:--
DDNS_USERNAME	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:--
DDNS_PASSWORD	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:--
DDNS	DDNS_TYPE='1' DDNS_HOSTNAME='' DDNS_SERVICE='dyndns' DDNS_USERNAME='' DDNS_PASSWORD=''	text/plain	RO*	G:RO*,C--	G:RO*,C:--
DATE_TYPE	1: Manual setting 2: NTP/SNTP	<value>	RW	G:RW,C--	G:RW,C:--
DATE_SNTP_IP		<ip address>	RW	G:RW,C--	G:RW,C:--

DATE_SNTP_UPDATE	30: 5 min 3600: 1 hour 21600: 6 hour 43200: 12 hour 86400: 1 day	<value>	RW	G:RW,C--	G:RW,C:--
DATE_MANUAL_DATE	MM: Month DD: Day hh: Hour mm: Minute YYYY: Year	<MMDDhhmm YYYY>	RW	G:RW,C--	G:RW,C:--
DATE_MANUAL_TIME	hh: Hour mm: Minute ss: Second	< hh:mm:ss>	RW	G:RW,C--	G:RW,C:--
DATE_MANUAL_ZONE	-12 ~ +00 ~ +13	<string>	RW	G:RW,C--	G:RW,C:--
DATE	DATE_TYPE='2' DATE_SNTP_IP='192.168.0.2' DATE_SNTP_UPDATE='86400' DATE_MANUAL_DATE='010100002004' DATE_MANUAL_TIME='00:00:00' DATE_MANUAL_ZONE='+00'	text/plain	RO*	G:RO*,C--	G:RO*,C:--
VIDEO_TOS_TYPE	1: Disabled 2: Enabled	<value>	RW	G:RW,C--	G:RO*,C:RW G=CH 1
VIDEO_TOS_PRIORITY	Minimize-Delay Maximize-Throughput Maximize-Reliability Minimize-Cost Normal-Service	<string>	RW	G:RW,C--	G:RO*,C:RW G=CH 1
TOS	VIDEO_TOS_TYPE='1' VIDEO_TOS_PRIORITY='Normal-Service'	text/plain	RO*	G:RO*,C--	G:RO*,C:RO* G=CH 1
ACCOUNT_ROOT_NAME	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_ROOT_PASSWORD	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_NAME_1	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_PASSWORD_1	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_NAME_2	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_PASSWORD_2	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*

ACCOUNT_USER_NAME_3	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_PASSWORD_3	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_NAME_4	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_PASSWORD_4	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_NAME_5	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_PASSWORD_5	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_NAME_6	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_PASSWORD_6	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_NAME_7	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_PASSWORD_7	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_NAME_8	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_PASSWORD_8	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_NAME_9	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_PASSWORD_9	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_NAME_10	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT_USER_PASSWORD_10	Max sizes: 30bytes	<string>	RW	G:RW,C--	G:RW,C:RO*
ACCOUNT	ACCOUNT_ROOT_NAME='Admin' ACCOUNT_ROOT_PASSWORD='123456' ACCOUNT_USER_NAME_1='' ACCOUNT_USER_PASSWORD_1='' ACCOUNT_USER_NAME_2='' ACCOUNT_USER_PASSWORD_2='' ACCOUNT_USER_NAME_3='' ACCOUNT_USER_PASSWORD_3='' ACCOUNT_USER_NAME_4='' ACCOUNT_USER_PASSWORD_4='' ACCOUNT_USER_NAME_5='' ACCOUNT_USER_PASSWORD_5=''	text/plain	RO*	G:RO*,C--	G:RW,C:RO*

	ACCOUNT_USER_NAME_6='' ACCOUNT_USER_PASSWORD_6='' ACCOUNT_USER_NAME_7='' ACCOUNT_USER_PASSWORD_7='' ACCOUNT_USER_NAME_8='' ACCOUNT_USER_PASSWORD_8='' ACCOUNT_USER_NAME_9='' ACCOUNT_USER_PASSWORD_9='' ACCOUNT_USER_NAME_10='' ACCOUNT_USER_PASSWORD_10=''				
PORT_HTTP	Http port number	<value>	RW	G:RW,C:--	G:RW,C:RO*
PORT_SEARCH_1	Client send search command to this port	<value>	RW	G:RW,C:--	G:RW,C:RO*
PORT_SEARCH_2	Server return search result to this port	<value>	RW	G:RW,C:--	G:RW,C:RO*
PORT_REGISTER	Register port for TCP1.0 streaming	<value>	RW	G:--,C:--	G:--,C:--
PORT_CONTROL	IO Control port for TCP1.0/2.0 streaming	<value>	RW	G:--,C:RW	G:RW,C:RO*
PORT_VIDEO	Video/Audio streaming port for TCP1.0/2.0	<value>	RW	G:--,C:RW	G:RW,C:RO*
PORT_MULTICAST	Multicast streaming port for TCP1.0/2.0	<value>	RW	G:RW,C:--	G:RW,C:RO*
V2_PORT_RTSP	RTSP port number	<value>	RW	G:--,C:RW	G:RW,C:RO*
PORT	PORT_HTTP='80' PORT_SEARCH_1='6005' PORT_SEARCH_2='6006' PORT_REGISTER='6000' PORT_CONTROL='6001' PORT_VIDEO='6002' PORT_MULTICAST='5000' V2_PORT_RTSP='7070'	text/plain	RO*	G:--,C:RO*	G:RO*,C:RO*
LANGUAGE	0: English 1: Traditional Chinese 2: Simplified Chinese 3: Japanese 4: Spanish 5: Italian 6: German 7: Portuguese 8: Czech 9: French	<value>	RW	G:RW,C:--	G:RW,C:RO*

V2_STREAMING_TYPE	1: TCP Version 1.0 2: TCP Version 2.0	<value>	RW	G:RW,C--	G:RO,C:RO*
V2_STREMAING_METHOD	0: TCP Only for TCP2.0 1: Multicast Only for TCP2.0 2: TCP & Multicast for TCP2.0 3: RTP over UDP for TCP2.0 4: RTP over Multicast for TCP2.0 5: RTP over UDP & Multicast for TCP2.0	<value>	RW	G:RW,C--	G:--,C:RW
V2_MULTICAST_IF	0: LAN Port 1: WAN Port	<value>	RW	G:RW,C--	G:RO, C:--
V2_MULTICAST_IP	224.3.1.0 ~ 239.255.255.255 for TCP2.0	<ip address>	RW	G:--,C:RW	G:RW,C:RO*
V2_PORT_RTP_MULTI_VIDEO	Video port for rtp over multicast	<value>	RW	G:--,C:RW	G:RW,C:RO*
V2_PORT_RTP_MULTI_AUDIO	Audio port for rtp over multicast	<value>	RW	G:--,C:RW	G:RW,C:RO*
V2_FAILOVER	0: Disabled 1: Enabled	<value>	RW	G:RW,C:--	G:--, C:--
IGMP_ENABLED	0: Disabled 1: Enabled	<value>	RW	G:RW,C:--	G:RW,C:RO*
SPEED_LAN	0: Auto detect speed 1: 100Mbps/Full Duplex 2: 100Mbps/Half Duplex 3: 10Mbps/Full Duplex 4: 10Mbps/Half Duplex	<value>	RW	G:RW,C:--	G:--, C:--
SPEED_WAN	0: Auto detect speed 1: 100Mbps/Full Duplex 2: 100Mbps/Half Duplex 3: 10Mbps/Full Duplex 4: 10Mbps/Half Duplex	<value>	RW	G:RW,C--	G:RW,C--
VIDEO_MULTICAST_TTL	1~255	<value>	RW	G:--,C:RW	G:RO*, C:RW
VIDEO_MULTICAST_IP	1~255 multicast ip for TCP1.0	<value>	RW	G:--,C:--	G:--,C--
VIDEO_LAN	DISABLE/TCP MULTICAST Streaming method for TCP1.0	<value>	RW	G:--,C:--	G:--,C--
VIDEO_WAN	DISABLE/TCP MULTICAST Streaming method for TCP1.0	<value>	RW	G:--,C:--	G:--,C--

HTTP Code Status

HTTP Code	HTTP Text	Description
200	OK	The request has succeeded, but an application error can still occur, which will be returned as an application error code.
204	No Content	The server has fulfilled the request, but there is no new information to send back.
400	Bad Request	The request had bad syntax or was inherently impossible to be satisfied.
401	Unauthorized	The request requires user authentication or the authorization has been refused.
404	Not Found	The server has not found anything matching the request.
409	Conflict	The request could not be completed due to a conflict with the current state of the resource.
500	Internal Error	The server encountered an unexpected condition which prevented it from fulfilling the request.
503	Service Unavailable	The server is unable to handle the request due to temporary overload.

Example :

Return success http context

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
 \n
```

Return failed http context

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
 \n
ERROR: error description
```

12 Sample Code Listing

URL Command for Mpeg4

CHANNEL=n might be added in the URL command where n is in the range of 1 to maximum channels. For example, the n is in the range of 1 and 8 for the 8-channel video server device. The CHANNEL=n should be followed by PWD parameter in an URL command.
If the CHANNEL=n is missed in this method (mpeg4), the CHANNEL=1 is used when read.

How to get video status

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_STATUS

How to get brightness

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_BRIGHTNESS

How to set brightness

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_BRIGHTNESS=a

<parameter>	**<values>**	**Description**
VIDEO_BRIGHTNESS	a: 0 ~ 100	0: -25IRE . .. 50: 0IRE . .. 100: 25IRE

How to get contrast

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_CONTRAST

How to set contrast

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_CONTRAST=a

<parameter>	**<values>**	**Description**
VIDEO_CONTRAST	a: 0 ~ 100	0: 0% . ..

		50: 100% . .. 100: 200%

How to get saturation

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_SATURATION

How to set saturation

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_SATURATION=a

<parameter>	**<values>**	**Description**
VIDEO_SATURATION	A: 0 ~ 100	0: 0% . .. 50: 100% . .. 100: 200%

How to get hue

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_HUE

How to set hue

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_HUE=a

<parameter>	**<values>**	**Description**
VIDEO_BRIGHTNESS	a: 0 ~ 100	0: -180 degree . .. 50: 0 degree . .. 100: 180 degree

How to get resolution

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_RESOLUTION

How to set resolution

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_RESOLUTION=xxxx

<parameter>	<values>	Description
VIDEO_RESOLUTION	Xxxx: string	N720x480: NTSC 720x480 N320x240: NTSC 320x240 N160x112: NTSC 160x112 P720x576: PAL 720x576 P352x288: PAL 352x288 P176x144: PAL 176x144

How to get bitrate

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_BITRATE

How to set bitrate

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_BITRATE=xxxx

<parameter>	<values>	Description
VIDEO_BITRATE	xxxx: string	28K: 28K bps 56K: 56K bps 128K: 128K bps 256K: 256K bps 384K: 384K bps 500K: 500K bps 750K: 750K bps 1M: 1M bps 1.5M: 1.5M bps 2M: 2M bps 2.5M: 2.5M bps 3M: 3M bps

How to get fps

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_FPS_NUM

How to set fps

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_FPS_NUM=a

<parameter>	<values>	Description
VIDEO_FPS_NUM	a: NTSC - 1,2,3,4,5,6,7,10,15,30 PAL - 1,2,3,4,5,6,8,12,25	

How to get camera name

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_CAMERA_NAME

How to set camera name

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_CAMERA_NAME=xxxx

<parameter>	**<values>**	**Description**
VIDEO_CAMERA_NAME	Xxxx: string	String max length : 15 bytes

How to get DIO status

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&DIO_STATUS	
<parameter>	**<values>**	**Description**
CHANNEL		Rackmount : Support Multi-Channel : Not Support
DIO_STATUS	0xnn: hexadecimal	BIT0: DI1 status BIT1: DI2 status BIT2: Reserved BIT3: Reserved BIT4: DO1 status BIT5: DO2 status BIT6: Reserved BIT7: Reserved

How to set DO

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&DIO_OUTPUT=0xnn

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support
DIO_OUTPUT	0xnn: hexadecimal	0x00 : DO1 LOW, DO2 LOW 0x01 : DO1 HI, DO2 LOW 0x02 : DO1 LOW, DO2 HI 0x03 : DO1 HI, DO2 HI

How to get motion enabled

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&MOTION_ENABLED

How to set motion enabled

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&MOTION_ENABLED=0xnn

<parameter>	**<values>**	**Description**
MOTION_ENABLED	0xnn: 0x00,0x01	0x00 : Motion disabled 0x01 : Motion enabled

How to get motion config

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&MOTION_STATUS

How to set motion config

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&MOTION_SETTING=a,b,c,d,e

<parameter>	**<values>**	**Description**
MOTION_SETTING	a: 1 ~ 3 b: 0 ~ 720 c: 0 ~ 480/576 d: 0 ~ 720 f: 0 ~ 480/576 g: 0 ~ 100	a: region number b: x upper c: y upper d: x lower f: y lower g: sensitive

How to set motion sensitive

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&MOTION_SENSITIVE=a,b

<parameter>	**<values>**	**Description**
MOTION_SETTING	a: 1 ~ 3 b: 0 ~ 100	a: region number b: 0: less sensitive . .. 50: middle sensitive . .. 100: more sensitive

How to get serial config

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&SERIAL_SETTING

How to set serial config

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&SERIAL_SETTING=xxxx,a

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support
SERIAL_SETTING	xxxx: 8N1,8O1,8E1 a: 2400,4800,9600,19200 38400,57600,115200	xxxx: Line Control a: Bitrate

How to send ASCII to serial

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&SERIAL_ASCII=xxxxxxxx

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support

How to send HEX to serial

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&SERIAL_HEX=xxxxxxxx

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support

How to on-fly-change variable fps

Syntax	http://192.168.1.1/cgi-bin/mpeg4?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_VARIABLE_FPS=a,b

<parameter>	**<values>**	**Description**
VIDEO_VARIABLE_FPS	a: client session id b: variable frame rate	A: get client session id from SDK b: NTSC: 1,3,6,30 PAL: 1,3,5,25

URL Command for System

CHANNEL=n might be added in the URL command where n is in the range of 1 to maximum channels. For example, the n is in the range of 1 and 8 for the 8-channel video server device. The CHANNEL=n should be followed by PWD parameter in an URL command.

How to save parameter to flash

Syntax	`http://192.168.1.1/cgi-bin/update?USER=Admin&PWD=123456&SAVE`

Save the configuration file of system to the flash

Syntax	`http://192.168.1.1/cgi-bin/update?USER=Admin&PWD=123456&CHANNEL=n&SAVE`

Save the configuration file of the nth video server to the flash

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support

Http return context

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
Save Finish
```

How to reboot system

Syntax	`http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&REBOOT`

Reboot the whole system included all of video servers

Syntax	`http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&REBOOT`

Reboot the nth video server only

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support

How to save parameter to flash and reboot system

Syntax	`http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&SAVE_REBOOT`

Save the configuration file of system and reboot included all of video servers

Syntax	`http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&SAVE_REBOOT`

Save the configuration file of the nth video server and reboot the nth video server

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support

		Multi-Channel: Not Support

How to set factory default

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&FACTORY_DEFAULT

Restore the factory default setting in the system

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&FACTORY_DEFAULT

Restore the factory default setting in the nth video server.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support

Http return context

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
Factory Default Finish
```

How to get system information

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&SYSTEM_INFO

Read the SYSTEM_INFO of the system.

Http return context

```
Firmware Version = A8D-R2N-V2.00.01-AC
MAC Address = 00:0F:7C:00:00:80
Production ID = SED2610
Factory Default Type = NTSC, Composite, Two Ways Audio (0x71).
NOTE: The return value of Factory Default Type is the CHANNEL=1 video server's value.
```

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&SYSTEM_INFO

Read the SYSTEM_INFO of the nth video server.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support

Http return context

```
Firmware Version = A1D-P2N-V2.00.07-AC
MAC Address = 00:0F:7C:00:00:67
Production ID = SED2400-04I-8-00027
Factory Default Type = NTSC, Composite, Two Ways Audio (0x71)
```

How to get system property

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&SYSTEM_PROPERTY

Read the SYSTEM_PROPERTY of the system. All of the return values are the same as the values of the CHANNEL=1 video server.

Http return context

```
SYSTEM='E'
```

```
TYPE='A'
NO_OF_CHANNEL='01'
MULTIPLEXING='X'
NO_OF_AUDIO_WAYS='2'
AUDIO_TYPE='PCM'
MOTION_TYPE='0'
PROTOCOL_TYPE='2'
```

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&SYSTEM_PROPERTY

Read the SYSTEM_PROPERTY of the nth video server.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support

Http return context

```
SYSTEM='E'
TYPE='A'
NO_OF_CHANNEL='01'
MULTIPLEXING='X'
NO_OF_AUDIO_WAYS='2'
AUDIO_TYPE='PCM'
MOTION_TYPE='0'
PROTOCOL_TYPE='2'
```

How to get protocol type

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&PROTOCOL_TYPE

Read the PROTOCOL_TYPE of the system. The 2 is always returned.

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&PROTOCOL_TYPE

Read the PROTOCOL_TYPE of the nth video server which is the same as the setting in the system

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support

<parameter>	**<values>**	**Description**
PROTOCOL_TYPE	a: 1 ~ 2	1: Run version 1 protocol 2: Run Version 2 protocol

How to get LAN

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&LAN

Read the LAN HOSTNAME settings in the system

How to get LAN hostname

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&LAN_HOSTNAME

Get the LAN HOSTNAME setting in the system

How to set LAN hostname

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&LAN_HOSTNAME=xxxxxxxx

Set the LAN HOSTNAME to the system and all of video servers.

How to get WAN

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&WAN

Read the WAN port settings in the system

How to set dynamic ip for WAN

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&WAN_TYPE=1

Set the dynamic type of WAN in the system.

How to set static ip for WAN

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&WAN_TYPE=2& WAN_IP=x.x.x.x&WAN_NETMASK=x.x.x.x&WAN_GATEWAY=x.x.x.x

Set the static wan ip in the system.

<parameter>	**<values>**	**Description**
WAN_TYPE	n : 1 ~ 2	1: Dynamic IP 2: Static IP
WAN_IP	x.x.x.x : IP address	Static IP address
WAN_NETMASK	x.x.x.x : NetMask	Netmask, ex: 255.255.255.0
WAN_GATEWAY	x.x.x.x : gateway IP	Default gateway ip

How to get DNS

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&DNS

Get the DNS settings in the system

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&DNS

Get the DNS settings in the nth video server which should be the same as the settings in the system

How to set DNS

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456& DNS_PRIMARY=x.x.x.x&DNS_SECONDARY=x.x.x.x

Set the DNS in the system and all of video servers.

<parameter>	**<values>**	**Description**
DNS_PRIMARY	x.x.x.x : IP address	Primary DNS server ip address
DNS_SECONDARY	x.x.x.x : IP address	Secondary DNS server ip address

How to get DDNS

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&DDNS

Get the DDNS in the system

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&DDNS

Get the DDNS in the nth video server which is the same as the setting in the system

How to disable DDNS

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&DDNS_TYPE=1

Disable the DDNS in the system and all of video servers.

How to enable DDNS

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&DDNS_TYPE=2& DDNS_SERVICE=xxxx&DDNS_HOSTNAME=x.x.x.x&DDNS_USERNAME=xxxx&DDNS_PASSWORD=xxxx

Enable the DDNS in the system and all of video servers.

<parameter>	**<values>**	**Description**
DDNS_TYPE	n: 1 ~2	1: Disabled 2: Enabled
DDNS_SERVICE	xxxx: string	dyndns: members.dyndns.org qdns: members.3322.org ezip: www.EZ-IP.Net pgpow: www.penguinpowered.com dhs: members.fhs.org ods: update.ods.com tzo: cgi.tzo.com easydns: members.easydns.com justlinux: www.justlinux.com dyns: www.dyns.cx hn: www.hn.org zoneedit: www.zoneedit.com
DDNS_HOSTNAME	x.x.x.x: string	Host domain name
DDNS_USERNAME	xxxx: string	User name
DDNS_PASSWORD	Xxxx: string	Password

How to get date

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&DATE

Get the DATE settings in the system

How to set manual config

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&DATE_TYPE=1 DATE_MANUAL_DATE=MMDDhhmmYYYY&DATE_MANUAL_TIME=hh:mm:ss&DATE_MANUAL_ZONE=nn

Set the MANUAL DATE settings in the system and all video servers.

How to set NTP/SNTP

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&DATE_TYPE=2 DATE_SNTP_IP=x.x.x.x&DATE_SNTP_UPDATE=m&DATE_MANUAL_ZONE=nn

Set the NTP/SNTP in the system and all of video servers.

<parameter>	**<values>**	**Description**
DATE_TYPE	n: 1 ~2	1: Manual setting 2: NTP/SNTP
DATE_MANUAL_DATE	MMDDhhmmYYYY: number	MM: Month DD: Day hh: Hour mm: Minute YYYY: Year
DATE_MANUAL_TIME	hh:mm:ss : number	Hh: Hour mm: Minute ss: Second
DATE_MANUAL_ZONE	nn: -12 ~ +00 ~ +13	Time zone
DATE_SNTP_IP	x.x.x.x: IP address	NTP/SNTP Server
DATE_SNTP_UPDATE	n: 30,3600,21600, 43200,86400	30: 5 min 3600: 1 hour 21600: 6 hour 43200: 12 hour 86400: 1 day

How to get TOS

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&TOS

Read the TOS in the system which is the same as the value in the CHANNEL=1 video server

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&TOS

Read the TOS in the nth video server

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support

How to disabled TOS

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_TOS_TYPE=1

Set the TOS of the nth video server.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support If the CHANNEL=n is missed, error message is returned. HTTP/1.0 200 OK\r\n Content-Type: text/plain\n \n ERROR!! The CHANNEL is not assigned!! Multi-Channel: Not Support

How to enabled TOS

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&VIDEO_TOS_TYPE=2& VIDEO_TOS_PRIORITY=xxxx

Enable TOS of the nth video server.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support If the CHANNEL=n is missed, error message is returned. HTTP/1.0 200 OK\r\n Content-Type: text/plain\n \n ERROR!! The CHANNEL is not assigned!! Multi-Channel: Not Support

<parameter>	**<values>**	**Description**
VIDEO_TOS_TYPE	n: 1 ~2	1: Disabled 2: Enabled
DATE_MANUAL_DATE	xxxx: string	Minimize-Delay Maximize-Throughput Maximize-Reliability Minimize-Cost Normal-Service

How to get account

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&ACCOUNT

Get the account information in the system

How to set root account

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456& ACCOUNT_ROOT_PASSWORD=xxxx&ACCOUNT_ROOT_PASSWORD=xxxx

Set the root account and password in the system and all of video servers.

How to set user account

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456& ACCOUNT_USER_NAME_1=xxxx&ACCOUNT_USER_PASSWORD_1=xxxx

Set the root account and password in the system and all of video servers.

<parameter>	**<values>**	**Description**
ACCOUNT_ROOT_NAME	xxxx: string	User name for root
ACCOUNT_ROOT_PASSWORD	xxxx: string	Password for root
ACCOUNT_USER_NAME_?	? : 1 ~ 10 xxxx: string	User name
ACCOUNT_USER_PASSWORD_?	? : 1 ~ 10 xxxx: string	Password

How to get port number

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&PORT

Get all port information of system. The value in the PORT_CONTROL, PORT_VIDEO and V2_PORT_RTSP ports is the same as the value in the CHANNEL=0 video server.

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&PORT

Get all port information of the nth video server. The value in the PORT_REGISTER, PORT_HTTP, PORT_MULTICAST, PORT_SEARCH_1 and PORT_SEARCH_2 are the same as the value in the system.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Support

How to set http port

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&PORT_HTTP=n

Set HTTP PORT of the system.

How to set search port

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456& PORT_SEARCH_1=n&PORT_SEARCH_2=m

Set SEARCH PORTS of the system.

How to set streaming port

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n& PORT_CONTROL=m&PORT_VIDEO=p

Set streaming ports for the nth video server.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support If the CHANNEL=n is not assigned in the URL command, the error message is returned. HTTP/1.0 200 OK\r\n Content-Type: text/plain\n \n ERROR!! The CHANNEL is not assigned! Multi-Channel: Support

How to set multicast port

Syntax `http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&PORT_MULTICAST=q`

Set multicast port in the system and all of video servers. This setting will be same as the URL command with V2_PORT_RTP_MULTI_VIDEO.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Not Support If the CHANNEL=n is assigned in the URL command, the error message is returned. HTTP/1.0 200 OK\r\n Content-Type: text/plain\n \n ERROR!! CHANNEL=n should not be set. Multi-Channel: Not Support

How to set RTSP port

Syntax `http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&V2_PORT_RTSP=p`

Set the RTSP port in the nth video server.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support If the CHANNEL=n is not assigned in the URL command, the error message is returned. HTTP/1.0 200 OK\r\n Content-Type: text/plain\n \n ERROR!! The CHANNEL is not assigned! Multi-Channel: Support

<parameter>	**<values>**	**Description**
PORT_HTTP	n: number	Web server port number

PORT_SEARCH_1	n: number	For ip search tool used
PORT_SEARCH_2	m: number	For ip search tool used
PORT_REGISTER	n: number	Verson 1 protocol used
PORT_CONTROL	m: number	Version 1/Version 2 protocol used
PORT_VIDEO	p: number	Version 1/Version 2 protocol used
PORT_MULTICAST	q: numba=er	Version 1/Version 2 protocol used
V2_PORT_RTSP	n: number	Version 2 protocol used

How to get language

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&LANGUAGE

Get the language setting in the system

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&LANGUAGE

Get the language setting in the nth video server which is the same as the value in the system

How to set language

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&LANGUAGE=n

Set the language in the system and all of video servers.

<parameter>	**<values>**	**Description**
LANGUAGE	n: number	0: English 1: Traditional Chinese 2: Simplified 3: Japanese 4: Spanish

How to get streaming type

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&V2_STREAMING_TYPE

Get the streaming type of the system. The 2 is always returned.

<parameter>	**<values>**	**Description**
V2_STREAMING_TYPE	n: number	1: TCP Version 1.0 2: TCP Version 2.0

How to set version 2.0 streaming method

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&V2_STREAMING_METHOD=n

Set the streaming method in the nth video server.

<parameter>	**<values>**	**Description**

CHANNEL		Rackmount: Support If the CHANNEL=n is not assigned in the URL command, the error message is returned. HTTP/1.0 200 OK\r\n Content-Type: text/plain\n \n ERROR!! The CHANNEL is not assigned! Multi-Channel: Not Support
V2_STREAMING_METHOD	n: number	0: TCP Only 1: Multicast Only 2: TCP & Multicast 3: RTP over UDP 4: RTP over Multicast

How to get multicast interface

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&V2_MULTICAST_IF

Get the multicast interface setting of the system

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&V2_MULTICAST_IF

Get the multicast interface setting of the nth video server which is the same as the setting of the system

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support Multi-Channel: Not Support

<parameter>	**<values>**	**Description**
V2_MULTICAST_IF	n: number	0: LAN Port 1: WAN Port

How to set version 2.0 multicast ip

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&V2_MULTICAST_IP=x.x.x.x

Set the Multicast IP address of the nth video server.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Support If the CHANNEL=n is not assigned in the URL command, the error message is returned. HTTP/1.0 200 OK\r\n Content-Type: text/plain\n \n ERROR!! The CHANNEL is not assigned!!

		Multi-Channel: Support

<parameter>	**<values>**	**Description**
V2_MULTICAST_IP	x.x.x.x: IP addres	Multicast ip address 224.3.1.0 ~ 239.255.255.255

How to set version 2.0 RTP over Multicast port number

Syntax	http://192.168.1.1/cgi-bin/system?USER=Admin&PWD=123456&CHANNEL=n&V2_PORT_RTP_MULTI_VIDEO=n& V2_PORT_RTP_MULTI_AUDIO=m

Set the multicast video and audio port in the system and all of the video servers.

<parameter>	**<values>**	**Description**
CHANNEL		Rackmount: Not Support If the CHANNEL=n is assigned in the URL command, the error message is returned. HTTP/1.0 200 OK\r\n Content-Type: text/plain\n \n ERROR!! CHANNEL=n should not be set. Multi-Channel: Not Support
V2_MULTICAST_IP	n: number m: number	n: Video port number for RTP over Multicast m: Audio port number for RTP over Multicast

13 URL Command for IP Quad

URL Command for IP Quad

How to set display mode

Syntax	http://192.168.1.1/cgi-bin/quad?DISPLAY=n

How to get display mode

Syntax	http://192.168.1.1/cgi-bin/quad?DISPLAY

<parameter>	**<values>**	**Description**
DISPAY	n: 0~4	0: quad display 1: display channel 1 2: display channel 2 3: display channel 3 4: display channel 4

How to set osd enabled

Syntax	http://192.168.1.1/cgi-bin/quad?OSD_ENABLED=0xnn

How to get osd enabled status

Syntax	http://192.168.1.1/cgi-bin/quad?OSD_ENABLED

<parameter>	**<values>**	**Description**
OSD_ENABLED	0xnn : hexadecimal	BIT0: 1:title name enabled BIT1: 1:video loss enabled BIT2: 1:motion detect enabled BIT3: 1:date time enabled BIT4: 1:DIO status enabled BIT5: Reserved BIT6: Reserved BIT7: Reserved

How to set motion detect enabled

Syntax	http://192.168.1.1/cgi-bin/quad?MOTION_ENABLED=0xnn

How to get motion enabled status

Syntax	http://192.168.1.1/cgi-bin/quad?MOTION_ENABLED

<parameter>	**<values>**	**Description**
MOTION_ENABLED	0xnn : hexadecimal	BIT0: 1:channel 1 motion detect enabled BIT1: 1:channel 2 motion detect enabled BIT2: 1:channel 3 motion detect enabled BIT3: 1:channel 4 motion detect enabled BIT4: Reserved BIT5: Reserved BIT6: Reserved BIT7: Reserved

How to set sensitive for motion detect

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&SENSITIVE=m

How to get sensitive setting

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&SENSITIVE

<parameter>	**<values>**	**Description**
CHANNEL	n: 1~4	channel number
SENSITIVE	m: 0~100	0: less sensitive . .. 50: middle sensitive . .. 100: more sensitive

How to set title name

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&TITLE_NAME=xxxxxxxx

How to get title name setting

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&TITLE_NAME

<parameter>	**<values>**	**Description**
CHANNEL	n: 1~4	channel number
TITLE_NAME	xxxxxxxx: title name	max length: 8bytes ASCII: A~Z & 0~9 & space

How to set brightness

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&BRIGHTNESS=m

How to get brightness setting

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&BRIGHTNESS

<parameter>	**<values>**	**Description**
CHANNEL	n: 1~4	channel number
BRIGHTNESS	m: 0~255	0: -25IRE . .. 128: 0IRE . .. 255: 25IRE

How to set contrast

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&CONTRAST=m

How to get contrast setting

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&CONTRAST

<parameter>	**<values>**	**Description**
CHANNEL	n: 1~4	channel number

CONTRAST	m: 0~255	0: 0% . .. 128: 100% . .. 255: 200%

How to set saturation

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&SATURATION=m

How to get saturation setting

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&SATURATION

<parameter>	**<values>**	**Description**
CHANNEL	n: 1~4	channel number
SATURATION	m: 0~255	0: 0% . .. 128: 100% . .. 255: 200%

How to set hue

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&HUE=m

How to get contrast setting

Syntax	http://192.168.1.1/cgi-bin/quad?CHANNEL=n&HUE

<parameter>	**<values>**	**Description**
CHANNEL	n: 1~4	channel number
HUE	m: 0~255	0: -180degree . .. 128: 0degree . .. 255: 180degree

How to get system information

Syntax	`http://192.168.1.1/cgi-bin/system?INFO`

Http return context

```
Firmware Version = SED2300Q-20050404.02-AC-D1
MAC Address = 00:0F:7C:00:00:67
Factory Default Type = NTSC (0x51)
Serial ID = SED2300-04I-8-00027
Model Number = SED-2300Q (11)
```

How to set factory default

Syntax	`http://192.168.1.1/cgi-bin/system?FACTORY_DEFAULT`

How to save all setting to flash and reboot system

Syntax	`http://192.168.1.1/cgi-bin/system?SAVE_REBOOT`

14 URL Command for Transcoder

URL Command for Transcoder

This category lists the commands that is related to MPEG-4 settings.

The syntax of the command is listed as follow:

```
http://<IP Address>/cgi-bin/url.cgi?USER=<Account Name>&PWD=<Password>&<Parameters>
```

Host Setting

Parameter	Value	Description
LAN_HOSTNAME		String (32 Bytes with terminal 0)
LAN_IP	x.x.x.x : IP address	LAN IP Address
LAN_NETMASK	x.x.x.x : IP address Mask	Netmask Address Format
LANGUAGE	0 ~ MAX LANGUAGE SUPPORT	
BAUDRATE	1 ~ 7	2400, 4800, 9600, 19200, 38400, 57600, 115200 BPS
UARTSETTING	0~2	0x00, 0x08, 0x18
OSD_X	8 ~ 96	OSD Upper Left X
OSD_Y	8 ~ 96	OSD Upper Left Y
VIDEO_TV	0, 1	0 : PAL, 1 : NTSC
VIDEO_OSD	1, 2, 4, 8, 16, 32	0x01:Time Code 0x02:Server IP 0x04:Camera Name 0x08:Local IP 0x10:Motion Detect 0x20:DI
VOL_AUDIOOUT	0, 1, 2, 3	0: mute 1: small 2: middle 3: large
VOL_AUDIOIN	0, 1, 2, 3	0: mute 1: small 2: middle 3: large

WAN Setting

Parameter	Value	Description
WAN_TYPE	1 ~ 3	1 : Dynamic IP 2 : Static IP 3 : PPPoE
WAN_IP	x.x.x.x : IP address	WAN IP Addrss
WAN_NETMASK	x.x.x.x : IP address Mask	Netmask Address Format
WAN_GATEWAY	x.x.x.x : IP address	Gateway Address
WAN_PPPOE_USERNAME		String (32 Bytes with terminal 0)
WAN_PPPOE_PASSWORD		String (32 Bytes with terminal 0)
DNS_PRIMARY		IP Address Format
DNS_SECONDARY		IP Address Format
DDNS_TYPE	1, 2	1 : Disabled 2 : Enabled
DDNS_HOSTNAME		String (32 Bytes with terminal 0)
DDNS_SERVICE	As right ➔	members.dyndns.org => dyndns members.3322.org => qdns www,EZ-IP.Net => ezip www.penguinpowered.com => pgpow members.dhs.org => dhs update.ods.org => ods cgi.tzo.com => tzo members.easydns.com => easydns www.justlinux.com => justlinux

		www.dyns.cx => dyns www.hn.org => hn www.zoneedit.com => zoneedit
DDNS_USERNAME		String
DDNS_PASSWORD		String

Video Server Connection Setting

URL Command Name	Value	Mark
PROTOCOLVERSION	1, 2	Version 1 or Version 2
CHANNEL	1 ~ 16	Channel Number
CHxx_VIDEO_CONNECT	0, 1, 2	0: MultiCast 1: UniCast 2: RTP
CHxx_VIDEO_VARFPS	0, 1, 2, 3, 4	0: Using Server's Setting 1: 30 (NTSC) or 25 (PAL) 2: 6 (NTSC) or 5 (PAL) 3: 3 (NTSC) or 3 (PAL) 4: 1 (NTSC) or 1 (PAL)
CHxx_VIDEO_STREAM	0, 1	0 : Disable, 1 : Enale
CHxx_VIDEO_AUDIO	0, 1	0 : Disable, 1 : Enale
CHxx_VIDEO_CONTROL	0, 1	0 : Disable, 1 : Enale
CHxx_VIDEO_TCP_IP	x.x.x.x : IP address	Server IP Address
CHxx_VIDEO_MULTICAST_IP	x.x.x.x : IP address	Server Multicast Address
CHxx_VIDEO_USERNAME		String (32 Bytes with terminal 0)
CHxx_VIDEO_PASSWORD		String (32 Bytes with terminal 0)
CHxx_PORT_STREAMIN		Stream Port Number
CHxx_PORT_CONTROL		Control & AudioOut Port Number
CHxx_PORT_MULTICAST		Server's Multicast Port Number
CHxx_PORT_RTP		Server's RTP Port Number
CHxx_PORT_HTTP		Server's Http Port Number
CHxx_TIME		Server's Dwell Time (seconds) 0 means infinite

User Account Setting

URL Command Name	Value	Mark
ACCOUNT_ROOT_NAME		String (32 Bytes with terminal 0)
ACCOUNT_ROOT_PASSWORD		String (32 Bytes with terminal 0)
ACCOUNT_USER_NAME_1		String (32 Bytes with terminal 0)
ACCOUNT_USER_PASSWORD_1		String (32 Bytes with terminal 0)
ACCOUNT_USER_NAME_2		String (32 Bytes with terminal 0)
ACCOUNT_USER_PASSWORD_2		String (32 Bytes with terminal 0)
ACCOUNT_USER_NAME_3		String (32 Bytes with terminal 0)
ACCOUNT_USER_PASSWORD_3		String (32 Bytes with terminal 0)
ACCOUNT_USER_NAME_4		String (32 Bytes with terminal 0)
ACCOUNT_USER_PASSWORD_4		String (32 Bytes with terminal 0)
ACCOUNT_USER_NAME_5		String (32 Bytes with terminal 0)
ACCOUNT_USER_PASSWORD_5		String (32 Bytes with terminal 0)
ACCOUNT_USER_NAME_6		String (32 Bytes with terminal 0)
ACCOUNT_USER_PASSWORD_6		String (32 Bytes with terminal 0)
ACCOUNT_USER_NAME_7		String (32 Bytes with terminal 0)
ACCOUNT_USER_PASSWORD_7		String (32 Bytes with terminal 0)
ACCOUNT_USER_NAME_8		String (32 Bytes with terminal 0)
ACCOUNT_USER_PASSWORD_8		String

		(32 Bytes with terminal 0)
ACCOUNT_USER_NAME_9		String (32 Bytes with terminal 0)
ACCOUNT_USER_PASSWORD_9		String (32 Bytes with terminal 0)
ACCOUNT_USER_NAME_10		String (32 Bytes with terminal 0)
ACCOUNT_USER_PASSWORD_10		String (32 Bytes with terminal 0)

Port Setting

URL Command Name	Value	Mark
PORT_HTTP'		Host HTTP PORT
PORT_HOST_SEARCH_1		HOST SEARCH PORT (Client to Server)
PORT_HOST_SEARCH_2		HOST SEARCH PORT (Server to Client)

URL Command for Transcoder Return Value

This category lists the commands that is related to Transcoder return value.

The syntax of the command is listed as follow:

```
http://<IP Address>/cgi-bin/url.cgi?USER=<Account Name>&PWD=<Password>&<Parameters>
```

Parameter	Description
HOST	Return all the items of host setting page
WAN	Return all the items of wan setting page
VIDEOSERVER	Return all the items of video setting page
ACCOUNT	Return all the items of user account page
PORT	Return all the items of port setting page
SYSTEM_LOG	Return all the items of system info page

URL Command for Transcoder System Setting

This category lists the commands that is related to Transcoder system settings.

The syntax of the command is listed as follow:

```
http://<IP Address>/cgi-bin/url.cgi?USER=<Account Name>&PWD=<Password>&<Parameters>
```

Parameter	Description
FACTORY_DEFAULT	Load factory default
SAVE_REBOOT	Save and Reboot
SWITCH	Switch to another video server or IP camera

URL Command for Transcoder Variable Frame Rate Setting

This category lists the commands that is related to Transcoder variable frame rate settings.

The syntax of the command is listed as follow:

```
http://<IP Address>/cgi-bin/url.cgi?USER=<Account Name>&PWD=<Password>&<Parameters>
```

Parameter	Description
VIDEO_VARFPS	Change the Variable Frame Rate on the fly in the current channel

Note that the defition is the same as CHxx_VIDEO_VARFPS

URL Command for Transcoder Connecting NVR

This category lists the commands that is related to Transcoder connecting to NVR setting

The syntax of the command is listed as follow:

```
http://<IP Address>/cgi-bin/url.cgi?netSendVideoCmd&
```

Parameter	Description
netSendVideoCmd	The URL Command for supporting NVR Ex: 0006ANET http://192.168.0.200/A2100?USER=Admin&PWD=123456&Cid=6

URL Command for Transcoder Samples

This category lists the sample URL Commands for Transcoder

How to get Transcoder host setting

Syntax	http://192.168.0.200/cgi-bin/url.cgi?USER=Admin&PWD=123456&HOST

How to set Transcoder host setting

Syntax	http://192.168.0.200/cgi-bin/url.cgi?USER=Admin&PWD=123456&LAN_HOSTNAME=ACTi&LAN_IP=192.168.1.20&LAN_NETMASK=255.255.255.0&LANGUAGE=1&BAUDRATE=4

How to save and reboot Transcoder

Syntax	http://192.168.0.200/cgi-bin/url.cgi?USER=Admin&PWD=123456&SAVE_REBOOT

How to get Transcoder system log

Syntax	http://192.168.0.200/cgi-bin/url.cgi?USER=Admin&PWD=123456&SYSTEM_LOG

How to set Transcoder to factory default

Syntax	http://192.168.0.200/cgi-bin/url.cgi?USER=Admin&PWD=123456&FACTORY_DEFAULT

How to request Transcoder to switch to another video server

Syntax	http://192.168.0.200/cgi-bin/url.cgi?USER=Admin&PWD=123456&VIDEO_TCP_IP=192.168.0.100&VIDEO_USERNAME=Admin&VIDEO_PASSWORD=123456&SWITCH