

SAMSUNG IP CAMERA SDK

VNP API Development Guide

Date 2009-06-22
Version 1.20

1. SAMSUNG IP CAMERA SDK	4
1.1 HTTP API.....	4
1.2 Standard RTP/RTSP API	4
1.3 VNP (Samsung Video Security Network Protocol) API	5
1.4 ActiveX Control API	5
1.5 XNS API.....	5
1.6 Supported IP Camera List	5
2. VNP (Samsung Video Security Network Protocol) API	6
2.1 Overview	6
2.1.1 Byte Ordering	6
2.1.2 Description for Connection.....	6
2.1.3 Sequence for connection and acquiring stream data	8
2.1.4 About “Object List”	10
2.1.5 How to make the “Object_ID”	12
2.2 Basic Communication	13
2.2.1 VNP Header.....	13
2.2.2 MEDIA Header.....	18
2.3 Command Protocol Definition	24
2.3.1 Command ID Value Table	24
2.3.2 CMD_LOGIN.....	25
2.3.3 CMD_LOGOUT.....	30
2.3.4 CMD_MEDIA_INFO.....	31
2.3.5 CMD_GET_OBJECT_LIST.....	34
2.3.6 CMD_MEDIA_REQUEST	38
2.3.7 CMD_DATA_SEARCH	43
2.3.8 CMD_SEARCH_CONTROL.....	48
2.3.9 CMD_TALK.....	52
2.3.10 CMD_ALARM_RESET	55
2.3.11 CMD_ALARM_OUT	57
2.3.12 CMD_VERSION	61
2.3.13 CMD_CAMERA OSDMENU	63
2.3.14 CMD_PTZ.....	66
2.3.15 CMD_PTZ_MOVE	69
2.3.16 CMD_PRESET.....	73
2.3.17 CMD_AUTOPAN.....	77

2.3.18 CMD_SCAN.....	80
2.3.19 CMD_PATTERN	83
2.3.20 CMD_AUTHORITY	86
2.3.21 CMD_GET_OBJECT_INFO	88
2.3.22 CMD_HEARTBEAT	92
2.3.23 CMD_DEVICE_INFO.....	94
2.3.24 CMD_GET_OBJECTLIST_ADV	98
2.4 Notification Protocol Definition	104
2.4.1 Event.....	104

1. SAMSUNG IP CAMERA SDK

Samsung Electronics IP Camera SDK has variable Application Programming Interface (API).

This SDK enables you to obtain images, audio stream, control IP camera functions (PTZ, Alarm I/O, etc.), set/get internal parameter values and much more. The purpose of the SDK is to make it easier for developers to build applications that support Samsung Electronics IP Cameras.

SAMSUNG IP CAMERA SDK consists of:

- HTTP API
- Standard RTP/RTSP API
- VNP (Samsung Video Security Network Protocol) API
- ActiveX Control API (For Windows Application Development)
- XNS API (For Windows Application Development)

As above, there are different ways whereby an application can interface with Samsung IP Cameras:

- Using low level protocol directly (HTTP API, RTP/RTSP API, VNP API)
- Using Windows development tools (XNS API, ActiveX API)

1.1 HTTP API

This API specifies the HTTP-based application programming interface (API) to integrate Samsung IP Cameras with 3rd Party Applications.

The HTTP API provides the functionality for requesting single and multi-part JPEG images and for getting and setting internal parameter values.

MPEG-4 and audio stream can not be obtained by HTTP API.

More information can be found in the “HTTP API” document.

1.2 Standard RTP/RTSP API

This API describes the standard RTSP-based application programming interface (API) to integrate Samsung IP Cameras with 3rd Party Applications.

Using this API, application can receive MJPEG, MPEG-4 video and one-way AUDIO stream from IP Camera.

More information can be found in the “RTP/RTSP API” document.

1.3 VNP (Samsung Video Security Network Protocol) API

This API describes the Samsung own VNP protocol-based application programming interface (API) to integrate Samsung IP Cameras with 3rd Party Applications.

Using this API, application can receive MJPEG, MPEG-4 video from IP Camera.

Also, VNP API can support bi-directional audio communication.

1.4 ActiveX Control API

The ActiveX Control API enables easy integration of viewing MPEG-4 and MJPEG streams directly in Microsoft Internet Explorer, Visual Basic, Delphi and other Windows applications.

Also, ActiveX API can support bi-directional audio communication.

ActiveX Control is worked by Samsung VNP protocol for network communication.

More information can be found in the “ActiveX Control API” document.

1.5 XNS API

The XNS API which is based on MFC style enables Windows Based Application development for viewing MPEG-4 and MJPEG streams from Samsung IP Cameras.

Also, XNS API can support bi-directional audio communication.

XNS API is worked by Samsung VNP protocol for network communication.

More information can be found in the “XNS API” document.

1.6 Supported IP Camera List

Model Name	Feature	Firmware
SNC-B2315	D1 Real-time Dual codec IP Camera	v2.01 and above
SNC-B5395	D1 Real-time Dual codec IP Anti-Vandal Dome Camera	v2.01 and above
SNC-M300	3 Mega Pixel IP Camera	v2.01 and above
SNC-C7225	10x Zoom PTZ Mini Speed Dome IP Camera (Outdoor)	V1.01 and above
SNC-C6225	10x Zoom PTZ Mini Speed Dome IP Camera	V1.01 and above
SNC-C7478	36x Zoom PTZ Speed Dome IP Camera	V1.01 and above

2. VNP (Samsung Video Security Network Protocol) API

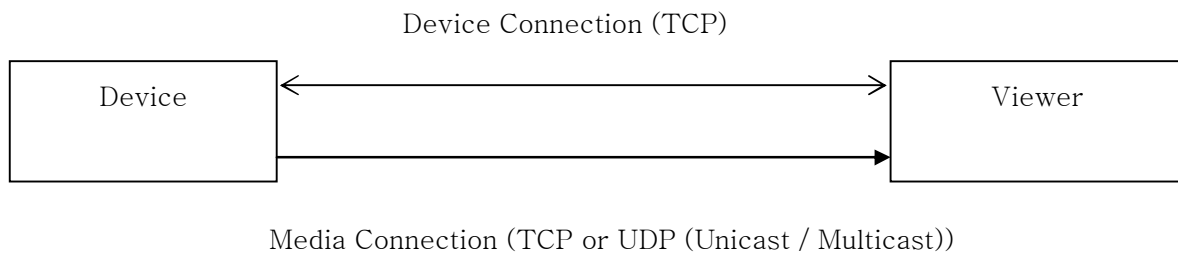
2.1 Overview

2.1.1 Byte Ordering

1. All data structures in this document are organized as Little Endian
2. All data structures in this document are using 'pack' pragma to consider byte align

2.1.2 Description for Connection

VNP uses at least two connections. One is the device connection, the other is the media connection.
 The device connection is used to transmit commands or event messages.
 The media connection is used to transmit Audio and Video stream data.



[Device Connection]

This session uses TCP/IP connection.

When VMS application connects to device, VMS application uses the device connection setting.

After completing connection, VMS application must log in to the device, and then VMS application gets the Session ID with the response of Log-in OK.

The Device Session ID is continuously used in order to identify connected VMS application by device.

If Log-in fails, connection must be closed.

[Media Connection]

This connection is for stream transmitting.

The information for Media connection is got by 'MEDIA_INFO' command. ([Refer to 2.3.4](#))

The response of MEDIA_INFO command contains streaming protocol type (TCP, UDP or Multicast), IP address and Port.

VMS application must complete the media connection by using acquired information for media connection.

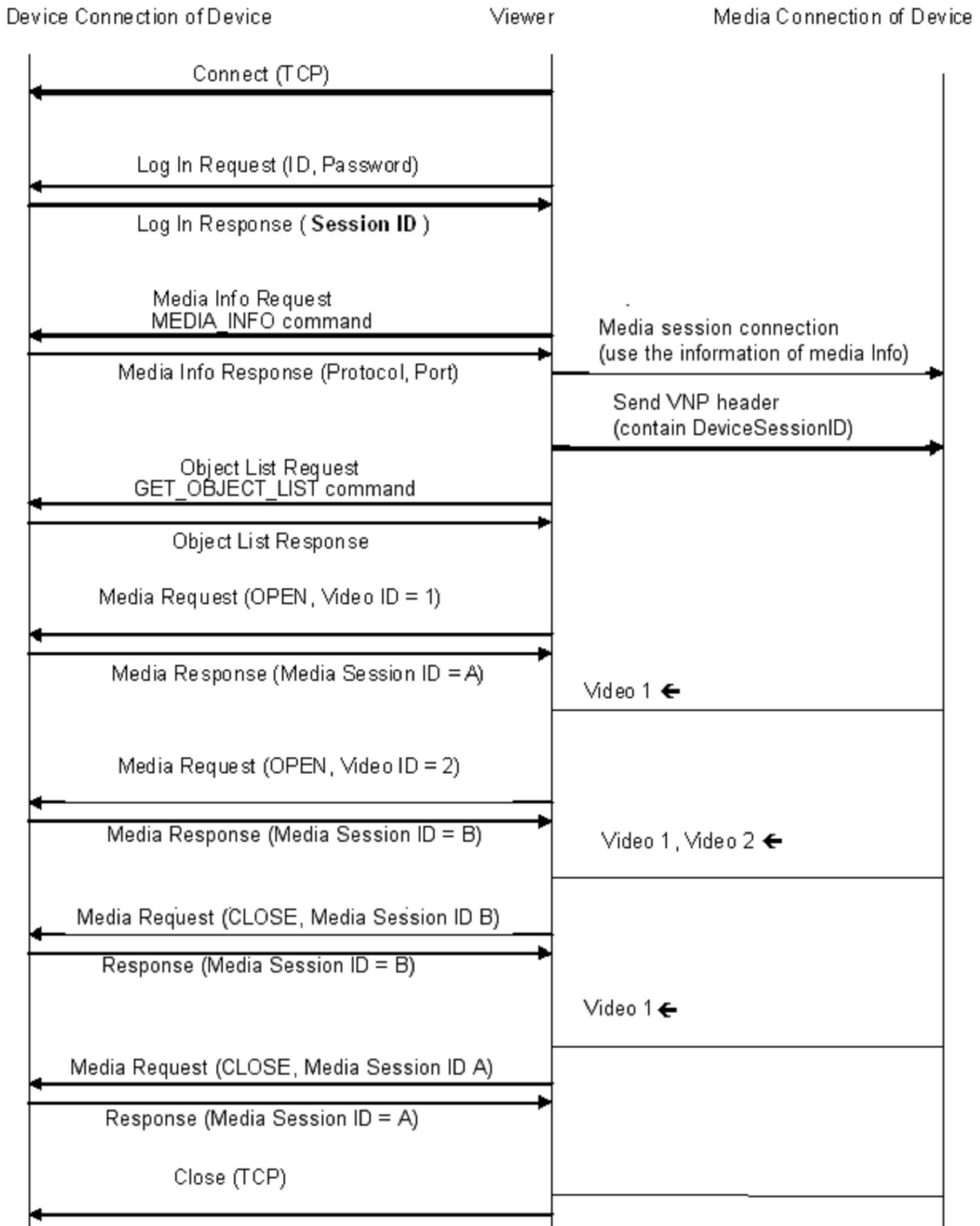
After completing connection, VMS application must send one header data to device for device to identify the VMS application.

All video and audio stream data are sent through the same media connection.

So, VMS application must identify these stream data by using media session ID and data type.

```
typedef enum
{
    MEDIA_UNKNOW = 0,
    MEDIA_VIDEO,
    MEDIA_AUDIO,
} MEDIA_TYPE2;
```

2.1.3 Sequence for connection and acquiring stream data



[Connection Procedure]

1) Connection: VMS application connects to Device using IP address and TCP port of Device. TCP port is “Device Port” of configuration menu in each IP device.

2) Log-In Request: VMS application sends the encrypted ID and PW to Device for Log-in. The Encryption algorithm is explained at [chapter 2.1.8](#).

3) Log-In Response: Device will check the validation of ID/PW and then response with “DeviceSessionID” value.

After finishing Log-In, VMS application communicates with IP Device using this “DeviceSessionID”.

4) Media Info Request: VMS application requests Media Session information to Device for connecting Media Streaming Session.

5) Media Info Response: Device returns Media Session Information to VMS application.

6) Media Session Connection: VMS application connects to Media Streaming Server in the Device by using Media Session information.

If the media session is multicast, VMS application sets multicast IP as multicast member.

7) Sending VNP Header: VMS application sends a VNP Header packet to Device through Media Session connection. This packet is used to making relation between the DeviceSessionID of Device and Media Session connection.

Because a Device can have multiple media session connections from multiple VMS applications at the same time, each Device Session and each Media Session must be recognized as one related connection.

And then a Device can transfer each audio and video stream to each VMS application.

8) Object List Request: VMS application requests object list to Device.

9) Object List Response: The Device return Object List of device.

This Object list includes Object information of Device and capabilities of each object.

The capabilities can be differentiated by Log-In level.

10) Media Stream Request for Video 1(or 2): VMS application requests transferring A/V stream to Device.

11) Media Stream Response: Device returns “Media Session ID” before transferring A/V stream. Then, using Media Session ID, Device will start transferring A/V stream.

※ If VMS application wants to stop receiving the A/V stream data,

12) Request Media Close: Device stop transferring A/V stream data

※ If VMS application wants to disconnect from the device.

13) Log-out: VMS application logs out from the device

2.1.4 About “Object List”

VNP defines independently controllable things as “Object” in the Device. Each Object can have some controllable capabilities.

Device is also one type of Object. So, Device has Object_ID ([for detail of Object_ID, Refer to 2.1.5](#)).

For example, IP Camera SNC-B2315 can be an Object itself(Device Object) and the Object “SNC-B2315” has an analog camera part Object and two video stream Objects. Because SNC-B2315 can support MPEG4/MJPEG Dual Streaming capability.

Therefore SNC-B2315’s object structure is like the below.

SNC-B2315	(Device Object)
└ Camera	(Analog Camera part Object in the SNC-B2315)
└ Video1	(Video Stream Object 1)
└ Video2	(Video Stream Object 2)

Regarding to the above concept, the Object can have the tree structure including sub Object. Each Object’s tree structure may be different because each Device or each Object will have different controllable capabilities.

For example, if an IP camera model supports just single streaming, the tree will be below.

IP Camera model
└ Camera
└ Video

And, each Object will have “Object_ID” in order to be distinguished with other Objects.

The data structure of each Object consists of the below.

```
typedef struct VNP_OBJECT_tag
{
    unsigned char    Object_type;
    unsigned char    Object_ID[10];
    unsigned char    Object_Parent[10];
    unsigned char    Object_Name[41];
    unsigned char    status;
    unsigned int     Object_cap;
    unsigned int     ptz_cap;
} _PACKED(VNP_OBJECT);
```

The parent Object can have multiple sub objects, so an “Object List” will be like the following.
One “Object List” has the number of “Sub Object”s and each data structure of “Sub Object”.

```
typedef struct VNP_OBJECTs_tag
{
    unsigned short   count;
    VNP_OBJECT       Object[count];
} _PACKED(VNP_OBJECT_LIST);
```

2.1.5 How to make the “Object_ID”

Object : It means some logical object which can be handled separately. (Ex, Device, Camera, video)

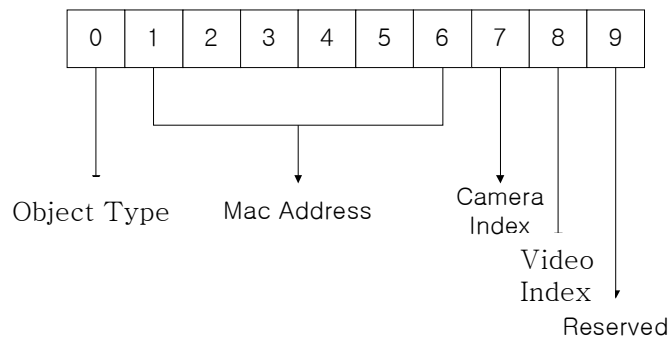
Object_ID : ID value for each object

[Organization of Object_ID]

“Object_ID” consists of 10 Bytes of binary value. This ID is made in the each Device.

The “Object_ID” is unique ID of each Device or each Object.

So, VMS applications just use this ID for recognizing each Device or Object.



Object Type + Mac Address (6 Bytes) + Camera Index + Video Index + Reserved Byte

* Object Type : Object type has following value for each type.

Object Type	Value
IP Camera Device	2
Analog Camera Part	4
Video	6
Alarm In	5
Alarm Out (Relay Out)	20 (0x14)

* Camera Index : This value is always 1.

* Video Index : The index of Video Stream (Ex. Video1 = 1 and Video2 = 2)

2.2 Basic Communication

All commands from VMS application to device must use Device Session.

Some commands are organized with 'VNP Header + Data'.

Some commands use only 'VNP Header'.

VNP Header contains Command ID and other information which are needed for executing command.

Additional data is attached in Data field following by Header CRC in VNP Header.

If there is no additional data, 'data length' field in VNP Header must be set to '0' and VMS software send only VNP Header (without data field).

Responses of each command have same structure.

Data structure for command and response is described as following.

2.2.1 VNP Header

2.2.1.1 Command Block

VNP Header		Data	
HEADER Data	HEADER CRC	DATA	DATA CRC

This header information is used in following case:

1. Sending command (Client S/W → Device)
2. Sending response (Device → Client S/W)
3. Making relation with device session and media session

To make this relation, send one VNP Header via media session to the device after connecting media session.

2.2.1.2 VNP Header structure in C programming

```
typedef struct vnp_header_tag
{
    unsigned char    majorversion;
    unsigned char    minorversion;
    unsigned char    Object_Type;
    unsigned short   Model_ID;
    unsigned char    Object_ID[10];
    unsigned short   Device_Session_ID;
    unsigned short   sequence;
    unsigned char    direction;
    unsigned char    Command_ID;
    unsigned char    Command_sub;
    unsigned char    Command_type;
    unsigned int     Length;
    unsigned char    Is_encrypted    : 1;
    unsigned char    Reserverd1     : 7;
    unsigned long    v_sequence;
    unsigned long    v_requestor;
    unsigned char    Reserverd2[2];
    unsigned short   Rescode;
    unsigned short   HeaderCRC;
} _PACKED(VNP_HEADER);
```

2.2.1.3 VNP Header Field Description

field	Bytes	Description	Value
Major version	1	Major version (protocol version)	0x01
Minor version	1	Minor version	0x00
Object_Type	1	Object type of device	2
Model_ID	2	Model Name At the initialization, this value is filled by device.	
Object_ID	10	Object_ID of device This value is explained in Object_ID. (Refer to 2.1.5) At the initialization, this value is filled by device.	
Device_Session_ID	2	ID for connection This value is used for command and matching client. This value is created by the device after VMS application log in to device.	
Sequence	2	ID for identifying command Response of one command has the same sequence ID. This value is created by VMS application. VMS application identifies the response that corresponds to request command. Device must echo this value in response data.	
Direction	1	Direction of Command and response (Refer to 2.2.1.4)	
Command_ID	1	Command ID value defined for each command Device must echo this value (Refer to 2.3.1 -Command ID Value Table)	
Command_sub	1	Sub command value Device must echo this value For sub-command refer to each command's description.	
Command_type	1	Not used Device must echo this value	Always 0
Length	4	Data length. It is 'Data + Data CRC' length in byte.	
Is_Encrypted	1bit	Use of data encryption	Always 0
Reserved	7bit	Not used. Reserved field.	
v_sequence	4	This value is used for VMS application.	

		Device must echo this value. VMS application can create this value at its disposal.	
v_requestor	4	This value is used for VMS application. Device must echo this value. VMS application can create this value at its disposal.	
reserved	2	Not used. Reserved field.	
Res code	2	Response code (Refer to 2.1.1.5)	
Header CRC	2	CRC for 'Major version ~ Res Code'	Not used

2.2.1.4 Direction of Command and Response

Value	Code	Direction	Description
Request	0x01 or '1'	V → D	Data from VMS application to Device
Response	0x02 or '2'	D → V	Data from Device to VMS application as response of command
Notify	0x03 or '3'	D → V	Data from Device to VMS application as notification of event
Notify_Response	0x04 or '4'	V → D	Data from VMS application to Device as response of event notification

2.2.1.5 Error Response Code

Value	Code	Description
RES_INVALID_PACKET_LENGTH	0x1001	Received data is not matched to length information
RES_INVALID_CRC_HEADER	0x1002	CRC Error for HEADER
RES_INVALID_CRC_DATA	0x1003	CRC Error for DATA
RES_INVALID_FORMAT	0x1004	Data Format is not valid or VNP Data is not valid
RES_INVALID_COMMAND	0x1005	Cannot find command or Undefined command ID
RES_VERSION_NOT_SUPPORTED	0x1006	Command is not support in this device protocol version
RES_CONTINUE	0x1007	Response data will be continued
RES_BAD_REQUEST	0x1008	
RES_UNAUTHORIZED	0x1009	Command before log in
RES_NOT_ALLOWED	0x100A	Command cannot be executed because user ID doesn't have privilege to execute command. Or Cannot find Object_ID for requesting media open

RES_RESPONSE_TIMEOUT	0x100B	Time out
RES_INVALID_RANGE	0x100C	Invalid data value
RES_INTERNAL_SERVER_ERROR	0x100D	Requested service cannot be provided because of internal server error
RES_SERVICE_UNAVAILABLE	0x100F	Requested service cannot be provided at this moment.

Value	Code	Description
RES_LOGIN_INVALID_ID	0x1101	Requested log in ID is not registered
RES_LOGIN_INVALID_PASSWORD	0x1102	Requested log in password is wrong
RES_LOGIN_USER_FULL	0x1103	Cannot accept more log in of user
RES_LOGIN_USER_CONFLICT	0x1104	Requested log in ID is already in the log-in state
RES_LOGIN_NEW_SESSION	0x1105	New session is created because requested session is not found.

Value	Code	Description
RES_UNSUPPORTED_MEDIA_TYPE	0x1151	
RES_NOT_ENOUGH_BANDWIDTH	0x1152	

Value	Code	Description
RES_UNKNWON	0x1201	Undefined Error
RES_REQUEST_VLOSS	0x1202	Video loss channel is requested for 'media open'
	0x0999	Response of reconnect

※ Reference : Division of error code

Response Code	Range	Description
RES_NOERR	0x0000	No error
	0x0001 ~ 0x3000	Error code related to network
	0x3001 ~ 0x6000	Error code related to device application
	0x6001 ~ 0x9000	Error code related to VMS application
	0x9001~ 0x9999	Etc.

2.2.2 MEDIA Header

2.2.2.1 Media Block (Device → VMS application)

This data structure is used for sending stream data from the device to VMS application.

The stream data contains audio data and video data via media session. Each data is organized with media header and audio header or video header. Basic unit of sending stream data is frame. But this frame data can be separated by packet.

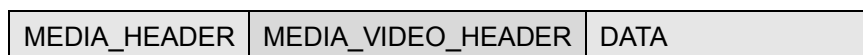
Each stream unit structure is as following.

[Video Frame]

If VNP protocol is working on the TCP/IP, one frame is sent by one packet data. But if VNP protocol is working on the UDP/IP (unicast or multicast), one frame is divided by several packets because maximum data length on the UDP/IP has limitation.

In case of sending frame as divided packet, the first packet of the frame has video header. But other packet of the same frame does not contain the video header.

In case of the first packet of the frame:



In case of the other packet of the same frame:



[Audio Frame]

Audio frame is sent by one packet on both TCP and UDP, because audio data size is small.



2.2.2.2 Media Header structure in C programming

```
typedef struct media_header_tag
{
    unsigned char    Major_version    : 4;
    unsigned char    Minor_version    : 4;
    unsigned long    Payload_length   : 20;
    unsigned long    Media_type       : 6;
    unsigned long    Is_extension     : 1;
    unsigned long    Is_marker        : 1;
    unsigned long    Is_sequence_reset : 1;
    unsigned long    Reserved1        : 3;
    unsigned short   Frame_sequence;
    unsigned short   Packet_sequence;
    unsigned long    Media_session_id;
    unsigned char    Object_ID[10];
} _PACKED(MEDIA_HEADER);
```

2.2.2.3 Media Header Field Description

field	Bytes	Description	Value
Major_version	4bit	Major version information (protocol version)	0x1
Minor_version	4bit	Minor version information	0x0
Payload_length	20bit	Data length in byte	
Media_type	6bit	Video or Audio	Video : 0x01 Audio : 0x02
Is_extension	1bit	Whether the current packet is first packet in the frame	
Is_marker	1bit	Whether the current packet is last packet in the frame	
Is_sequence_reset	1bit	Not used	0x00
Reserved1	3bit	Not used	
Frame_sequence	2	Frame sequence number Successive frame has the frame number that is added 1 from previous frame sequence number.	
Packet_sequence	2	Packet sequence number	

Media_session_ID	4	Media Session ID. This value is used to identify the each video stream. (Refer to 2.3.6 CMD_MEDIA_REQUEST)	
Object_ID	10	Object_ID of device (Refer to 2.1.5 Object_ID)	

Payload_length : Not used in the IP camera

Actual value is set in the frame-size field of Video Header.

Is_extension : This value is set to 1 if the packet is the first packet in the frame.

VMS application can use this value to identify the first packet of the frame.

Is_marker : This value is set to 1 if the packet is the last packet in the frame.

VMS application can use this value to identify the last packet of the frame.

If protocol is working on the TCP, one frame is sent by one packet. Therefore

is_extension and is_marker field are set to 1 for all packets.

Frame_sequence : Increased value by the frame.

VMS application can use this value for following purpose.

For identifying whether received frame data is successive from the previous frame.

If codec is MPEG4 and the received frame data is not I-Frame and not successive value, that frame data can not be decoded.

For identifying whether the received packet is data from the same frame if protocol is working on the UDP.

Packet_sequence : Increasing value by the packet.

This value is started from 0 for each frame.

VMS application can use this value for following purpose.

If protocol is working on the UDP, this value tells the order of received packet in the frame.

If protocol is working on the TCP, this value is not used.

Media_session_ID : This value is used to identify the stream of IP camera if multiple streams are sent at the same time.

Media_session_ID is got by using CMD_MEDIA_REQUEST with MEDIA_OPEN sub_command. (Refer to 2.3.6). The response of this command contains the Media_session_ID.

This session ID is used for identifying to corresponding request stream data.

2.2.2.4 Media Video Header structure in C programming

```
typedef struct media_video_header_tag
{
    unsigned char    Media_type;
    unsigned char    Codec_type;
    unsigned short   Frame_number;
    unsigned long    Time_stamp;
    unsigned long    Reserved           : 20;
    unsigned long    Frame_type        : 2;
    unsigned long    Frame_rate        : 5;
    unsigned long    Reserved2         : 5;
    unsigned short   Frame_width;
    unsigned short   Frame_height;
    long             Frame_time;
    unsigned long    Frame_size;
} _PACKED(MEDIA_VIDEO_HEADER);
```

2.2.2.5 Media Video Header Field Description

Media type: Indicates the data type (video data or audio data)

```
typedef enum
{
    MEDIA_UNKNOW = 0,
    MEDIA_VIDEO,
    MEDIA_AUDIO,
} MEDIA_TYPE2;
```

Codec type: Compression type.

Value	Data Type	Codec Type
0x02	Video	MJPEG
0x03		MPEG4
0x11	Audio	G.711_MULAW

Frame Number : The same value with Frame_Number in Media Header

Time_stamp : This value is used to know the time-interval of frame.

When VMS application gets the live stream data, this value is increased by millisecond.

VMS application can get the present time of current frame by using subtraction with previous frame

When VMS application gets the playback stream, this value is the time-interval of frame.

Reserved (20bit) : Not used

Frame_type : Frame Type in VOP. In case of MPEG-4, there are three types of frame. (I, P, B)

But MJPEG uses only IVOP.

```
typedef enum
{
    MPEG_UNKNOWN = 0,
    MPEG_IVOP,           //used in MJPEG
    MPEG_PVOP,
    MPEG_BVOP
} MPEG_FRAME_TYPE;
```

Frame_rate : Not used (always 0)

Reserved2 : Not used (always 0)

Frame_width : Width in pixel of video image

Frame_height : Height in pixel of video image.

Frame_time : Time value that the video image is created. (4-byte time_t value)

Frame_size : frame length in bytes of video frame.

2.2.2.6 Media Audio Header structure in C programming

```
typedef struct media_audio_header_tag
{
    unsigned char    Media_type;
    unsigned char    Codec_type;
    unsigned long    Time_stamp;
    unsigned long    Frame_size        : 20;
    unsigned long    Reserved          : 12;
} _PACKED(MEDIA_AUDIO_HEADER);
```

2.2.2.7 Media Audio Header Field Description

Media type : Indicates the data type (video data or audio data)

```
typedef enum
{
    MEDIA_UNKNOW = 0,
    MEDIA_VIDEO,
    MEDIA_AUDIO,
} MEDIA_TYPE2;
```

Codec type : Compression type.

Value	Data Type	Codec Type
0x02	Video	MJPEG
0x03		MPEG4
0x11	Audio	G.711_MULAW

Time_stamp : In case of audio frame, play time is decided by frame length. So, this value is not used.

Frame_size : Data length in bytes of audio frame.

2.3 Command Protocol Definition

2.3.1 Command ID Value Table

Command ID	Value	
CMD_LOGIN	001	0x01
CMD_LOGOUT	002	0x02
CMD_GET_OBJECT_LIST	004	0x04
CMD_DATA_SEARCH	006	0x06
CMD_MEDIA_INFO	007	0x07
CMD_MEDIA_REQUEST	008	0x08
CMD_SEARCH_CONTROL	010	0x0A
CMD_DEVICE_EVENT	011	0x0B
CMD_CAMERA OSDMENU	012	0x0C
CMD_PTZ	013	0x0D
CMD_PRESET	015	0x0F
CMD_AUTOPAN	016	0x10
CMD_SCAN	017	0x11
CMD_PATTERN	018	0x12
CMD_GET_OBJECT_INFO	035	0x23
CMD_TALK	033	0x21
CMD_ALARM_RESET	034	0x22
CMD_HEARTBEAT	040	0x28
CMD_DEVICE_INFO	042	0x2A
CMD_GET_OBJECTLIST_ADV	043	0x2B
CMD_PTZ_MOVE	046	0x2E
CMD_VERSION	100	0x64
CMD_AUTHORITY	101	0x65
CMD_ALARM_OUT	103	0x67

2.3.2 CMD_LOGIN

2.3.2.1 Description

This is the request command for Log in to the Device after connecting Device Session.

Device decides acceptance by using received ID and Password.

In case of Log in success, device returns Device Session ID. (res_code = 0x00(NO_ERR))

And then all commands must use the returned Device Session ID.

In case of Log in failure, device returns the cause of failure by using res_code.

2.3.2.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x00
Model_ID	0x0000
Object_ID	0x00000000000000000000
Device_Session_ID	0x0000
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x01 (CMD_LOGIN)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_REQ_LOGIN)
Is_Encrypted	0x00 (1bit)
Reserved1	0x00 (7bits)
v_sequence	0x00000000 ~ 0xffffffff
v_requestor	0x00000000 ~ 0xffffffff
Reserved2	0x0000
Res code	0x0000
Header CRC	0x0000

[Data Structure]

```
typedef struct vnp_req_login_tag
{
    unsigned char    id[21];
    unsigned char    password[21];
    unsigned char    mode;
    unsigned char    mac[6];
    unsigned short   dataCRC;
} _PACKED(VNP_REQ_LOGIN);
```

id : Log in ID, Encrypted null-terminated string by Samsung own encryption algorithm.

password : Log in PW, Encrypted null-terminated string by Samsung own encryption algorithm.

mode : Log in mode. (0x00)

mac : Not used

2.3.2.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. This value is set by device.
Object_ID	Object_ID of device. This value is set by device.
Device_Session_ID	session ID for connection. This value is set by device.
Sequence	The same value with Request
Direction	0x02
Command_ID	0x01 (CMD_LOGIN)
Command_sub	0x00
Command_type	0x00
Length	0x00
Is_Encrypted	0x00
Reserved1	0x00
v_sequence	The same value with Request

v_requestor	The same value with Request
Reserved2	0x00
Res code	Result code
Header CRC	0x00

<2009.3.4>

Model_ID : Model ID must be filled by Device.

This value is valid if the IP Camera's firmware version is equal or higher than following table.

Model	F/W version
SNC-B2315	Not Supported
SNC-B5395	Not Supported
SNC-C6225	Ver. 1.01
SNC-C7225	Ver. 1.01
SNC-C7478	Ver. 1.01
SNC-M300	Not Supported

Alias	Value	Model
VNPE_DTYPE_ENC_SNT1010	0x0401	SNT-1010
VNPE_DTYPE_IPC_SNC2315	0x0600	SNC-B2315
VNPE_DTYPE_IPC_SNC5395	0x0610	SNC-B5395
VNPE_DTYPE_IPC_SNC6225	0x0501	SNC-C6225/C7225
VNPE_DTYPE_IPC_SNC7478	0x0501	SNC-C7478
VNPE_DTYPE_IPC_SNCM300	0x0630	SNC-M300
VNPE_DTYPE_NIW_SEV000	0x0700	NET-i ware
VNPE_DTYPE_IPC_SNC2331	0x0601	SNC-B2331
VNPE_DTYPE_IPC_SNC2335	0x0602	SNC-B2335
VNPE_DTYPE_IPC_SNC5368	0x0611	SNC-B5368
VNPE_DTYPE_IPC_SNC5399	0x0612	SNC-B5399

[Res Code]

Value	Code	Description
RES_LOGIN_INVALID_ID	0x1001	Requested log in ID is not registered
ERR_RES_INVALID_CRC_HEADER	0x1002	not used
ERR_RES_INVALID_CRC_DATA	0x1003	not used
ERR_RES_INVALID_FORMAT	0x1004	
ERR_RES_INVALID_COMMAND	0x1005	Command ID don't exist
ERR_RES_VERSION_NOT_SUPPORTED	0x1006	
ERR_RES_CONTINUE	0x1007	
ERR_RES_BAD_REQUEST	0x1008	
ERR_RES_UNAUTHORIZED	0x1009	log-in fail (id or password is wrong)
ERR_RES_NOT_ALLOWED	0x100A	Invalid command
RES_LOGIN_INVALID_ID	0x1101	Requested log in ID is not registered
RES_LOGIN_INVALID_PASSWORD	0x1102	Requested log in password is wrong
RES_LOGIN_USER_FULL	0x1103	Cannot accept more log in of user
RES_LOGIN_USER_CONFLICT	0x1104	Requested log in ID is already in the log-in state
RES_LOGIN_NEW_SESSION	0x1105	New session is created because requested session is not found.

* Request data structure containing log in information must be encrypted by using Samsung own encryption algorithm.

2.3.2.4 Encryption Algorithm

```
#define LEN_MAC      6
#define LEN_STRING  21
const unsigned char ID_KEY[] = "VSS_VNP2.0";
const unsigned char PW_KEY[] = "VNP2.0_PW_VSS";
const unsigned char ENC_TABLE[] = {
    0x52, 0x50, 0x07, 0x15, 0x23, 0x33, 0x41, 0x49,
    0x50, 0x53, 0x61, 0x00, 0x08, 0x16, 0x24, 0x26,
    0x34, 0x42, 0x10, 0x28, 0x36, 0x44, 0x63, 0x56,
    0x03, 0x11, 0x19, 0x29, 0x37, 0x45, 0x33, 0x56,
    0x57, 0x04, 0x12, 0x20, 0x30, 0x38, 0x46, 0x58,
    0x05, 0x13, 0x21, 0x31, 0x39, 0x47, 0x62, 0x06,
```

```

    0x14, 0x22, 0x32, 0x40, 0x48, 0x08, 0x37, 0x18,
    0x34, 0x26, 0x12, 0x38, 0x26, 0x58, 0x45, 0x08
};

void EncLoginData( VNP_REQ_LOGIN *pData )
{
    unsigned int i;
    unsigned char Key = 0;
    unsigned char flag=0;
    for( i = 0 ; i < strlen( (char*)ID_KEY ) ; i++ )
        Key ^= ID_KEY[i];
    for( i = 0 ; i < LEN_MAC ; i++ )
        Key ^= pData->mac[i];

    for( i = 0 ; i < LEN_STRING ; i++ )
    {
        pData->id[i] ^= Key;
        flag = (Key&0x80);
        Key = ((Key<<1)+ !flag)^(ENC_TABLE[i]);
    }

    Key = 0;
    for( i = 0 ; i < strlen( (char*)PW_KEY ) ; i++ )
        Key ^= PW_KEY[i];
    for( i = 0 ; i < LEN_MAC ; i++ )
        Key ^= pData->mac[i];

    for( i = 0 ; i < LEN_STRING ; i++ )
    {
        pData->password[i] ^= Key;
        flag = (Key&0x80);
        Key = ((Key<<1)+ !flag)^(ENC_TABLE[(63-i)%64]);
    }
}

```

2.3.3 CMD_LOGOUT

2.3.3.1 Description

Request to disconnect Device Session

Device disconnects the client by using Device Session ID.

2.3.3.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x02 (CMD_LOGOUT)
Command_sub	0x00
Command_type	0x00
Length	0x00
Is_Encrypted	0x00
Reserved1	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
Reserved2	0x00
Res code	0x00.
Header CRC	0x00

2.3.3.3 Response Data Structure

None

2.3.4 CMD_MEDIA_INFO

2.3.4.1 Description

Get the information to establish Media Session connection.

Media session is the connection to receive stream data.

VMS application must establish media session by using media session information according to TCP or UDP (unicast, multicast) setting.

After connecting to device with the information by using this command, VMS application must send VNP Header data with Device Session ID that makes relation between media session and device session in the device.

This command is used to keep connection also. Client S/W must send this command every 10 seconds. If device doesn't receive this command more than 10 seconds, device may force to disconnect.

2.3.4.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x07 (CMD_MEDIA_INFO)
Command_sub	0x00
Command_type	0x00
Length	0x00
Is_Encrypted	0x00
Reserved1	0x00

v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
Reserved2	0x00
Res code	0x00.
Header CRC	0x00

2.3.4.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x07 (CMD_MEDIA_INFO)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_MEDIA_INFO)
Is_Encrypted	0x00
Reserved1	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
Reserved2	0x00
Res code	Result Code
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_media_info_tag
{
    unsigned char    protocol;
    unsigned char    ip_address[4];
    unsigned int     port;
} _PACKED (VNP_MEDIA_INFO);
```

protocol : Current setting protocol value that is used for sending stream data.

TCP (0), UDP-unicast (1), UDP-multicast (2)

ip_address : This value is valid only if protocol type is UDP-multicast.

If protocol set to UDP-unicast or TCP, VMS application must use the same IP address with Device connection IP.

port : Port number to connect. VMS application uses this port number regardless protocol setting.

2.3.5 CMD_GET_OBJECT_LIST

2.3.5.1 Description

Get the Object_ID and capability of each object to control the object.

By using this command, VMS application can get the structure of object and object's capability.

VMS application has to use Object_ID which is acquired by using this command when controlling the each object. Also VMS application can use functions which are described in capability.

2.3.5.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x04 (CMD_GET_OBJECT_LIST)
Command_sub	0x00
Command_type	0x00
Length	0x00
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

2.3.5.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x04 (CMD_GET_OBJECT_LIST)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_OBJECTS)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	Result Code
Header CRC	0x00

[Data Structure]

```
typedef struct VNP_OBJECTs_tag
{
    unsigned short    count;
    VNP_OBJECT      Object[count];
} _PACKED(VNP_OBJECTS);

typedef struct VNP_OBJECT_tag
{
    unsigned char     Object_type;
    unsigned char     Object_ID[10];
    unsigned char     Object_Parent[10];
    unsigned char     Object_Name[41];
    unsigned char     Status;
    unsigned int      Object_cap;
    unsigned int      Ptz_cap;
} _PACKED(VNP_OBJECT);
```

Count : Number of total Objects

Object_type : [Refer to 2.1.5](#)

Object_ID : Object_ID of current object

Object_Parent : Parent object's Object_ID.

Object_Name : Name of the object , Null-terminated string

Status : Current state (possible to use or not) of Object

```
typedef enum
{
    OSTATUS_OFF = 0x0,
    OSTATUS_ON,
} VNPE_OSTATUS;
```

Object_cap : Object Capability. Capability means the functions that device can support.

If device can transfer live image to the VMS application, this device has the capability of 'Live monitoring'.

In this case, Object capability has the value of 'Live monitoring' (0x01).

Device has capability of bit-wise ORing value of each capability value that device can provide.

Value	Description
0x0000 0001	Possible to use Live monitoring
0x0000 0002	Possible to use Remote search from SD memory in the Device
0x0000 0004	Possible to receive audio data from Device
0x0000 0008	Possible to send audio data to Device
0x0000 0010	Possible to control PTZ operation

Ptz_cap : This value indicates the capability of PTZ function. Each bit match the PTZ operation as following table.

Value	Description
0x0000 0001	CAP_PTZ_BASIC (Zoom In/Out, 방향 control, menu)
0x0000 0002	CAP_PTZ_POWER (Power PTZ)
0x0000 0004	CAP_PTZ_FUNCTION
0x0000 0008	CAP_PTZ_IRIS
0x0000 0010	CAP_PTZ_FOCUS
0x0000 0020	CAP_PTZ_BRIGHTNESS
0x0000 0040	CAP_PTZ_CONTRAST

ex) 0x9D means (b'1001 1101)

the camera support BASIC operation, Function, Iris, focus, contrast.

2.3.6 CMD_MEDIA_REQUEST

2.3.6.1 Description

Request to start sending stream data or stop sending stream data.

Start or stop is identified by cmd field in data structure(VNP_MEDIA_REQUEST).

Parameter and function are decided according to mode type.

If the mode is live, this command controls to start or stop sending the live stream data.

If the mode is playback, this command makes device to set waiting mode. In the waiting mode, device will wait till getting search control command ([Refer to 2.3.8](#)).

Device will decides to start or stop according to parameter command.

If there are some error to execute, error code is returned by using res-code. (ex, user limitation for playback)

In case of sending multi-stream data at the same time, all stream data are sent via the same connection (media session), it is required to separate each stream. For this purpose, media session id is used. Media_session ID value is returned by the result of MEDIA_OPEN command in the data structure, and then device send video or audio stream with this media session id that is contained in the Media Header ([Refer to 2.2.2](#)).

When VMS application requests to start sending stream data, Object_ID is used to identify request stream data. But When VMS application requests to stop sending stream data, media_sessionID vaule is used to identify the stream.

This command is supported if the IP Camera's firmware vision is equal or higher than following table.

Command (mode)	Model	F/W version
Live (mode=0)	SNC-B2315	Ver. 2.01
	SNC-B5395	Ver. 2.01
	SNC-C7478	Ver. 1.01
	SNC-C7225	Ver. 1.01
	SNC-C6225	Ver. 1.01
	SNC-M300	Ver. 2.01
Search (mode = 1)	SNC-B2315	Ver. 2.01
	SNC-B5395	Ver. 2.01
	SNC-C7478	Ver. 1.01
	SNC-C7225	Ver. 1.01

	SNC-C6225	Ver. 1.01
	SNC-M300	Ver. 2.01

2.3.6.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x08 (CMD_MEDIA_REQUEST)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_MEDIA_REQUEST)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_media_request_tag
{
    unsigned char cmd;                //MREQ_OPEN(0), MREQ_CLOSE(1),
    unsigned char mode;              //live(0), search(1)
    unsigned char Object_ID[10];
    long start_time;                 //Search mode or backup mode
    long end_time;                   //Search mode or backup mode
    unsigned long replay_frames;     //not used
    unsigned long v_tile_id          //0x00000000~0xFFFFFFFF
    unsigned long media_session_id;
} _PACKED (VNP_MEDIA_REQUEST);
```

cmd : identify to start or stop of stream data. Refer to the below table (Command value).

mode : identify to media stream mode. Refer to below table (Mode).

Mode	Command value	Parameter	Description
Live(0)	MREQ_OPEN (0)	Object_ID	Request to start sending stream data
	MREQ_CLOSE (1)	media_session_id	Request to stop sending stream data
Search(1)	MREQ_OPEN (0)	Object_ID	Seek to requested time and wait to media control command
	MREQ_CLOSE (1)	media_session_id	Close sending data mode.

Object_ID : Object_ID for applying command. This is used only if the command is MREQ_OPEN.

start_time : start time for searching recorded data in the SD memory

end_time : end time for searching recorded data in the SD memory.

replay_frames : not used (0x00)

v_tile_id : This value is for the Client S/W. Device must echo this value in the response.

media_session_id : returned value as the result of MREQ_OPEN command. Device must set this value to identify video 1 or video 2.

Once media is opened, other commands use this value to control (ex, play, stop, close) each video.

2.3.6.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x08 (CMD_MEDIA_REQUEST)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_MEDIA_REQUEST)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	Result Code
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_media_request_tag
{
    unsigned char cmd;                //MREQ_OPEN(0), MREQ_CLOSE(1),
    unsigned char mode;              //live(0), search(1)
    unsigned char Object_ID[10];
    long start_time;                 //only for search mode
    long end_time;                   //0x00
    unsigned long replay_frames;     //not used
    unsigned long v_tile_id          //0x00000000~0xFFFFFFFF
    unsigned long media_session_id;
} _PACKED (VNP_MEDIA_REQUEST);
```

cmd : echo of the request value

mode : echo of the request value

Object_ID : echo of the request value

start_time : This value is valid only if the mode is Search.

Device seeks the time position and set the searched time value to this field. If there are some recorded data at the seek position, this value is set the same value of request time.

But if there are no data, this value is set to the time value that exist recorded data close to requested time.

After seek, device wait for media control command.

end_time : This value is valid only if the mode is Search

replay_frames : not used (0x00)

v_tile_id : not used (0x00)

media_session_id : returned value as the result of MREQ_OPEN command. Device must set this value to identify video 1 or video 2.

Once media is opened, other commands use this value to control (ex, play, stop, close) each video.

[Res Code]

Value	Code	Description
ERR_SUCCESS	0x00	Start or stop command is executed.

2.3.7 CMD_DATA_SEARCH

2.3.7.1 Description

This command is used for getting the information of recorded data in the SD memory of IP Camera. The response data contains the interval of recorded data and the description of recording.

Search Condition	Sub command	Sub Command value	Parameter	Result
Calendar search	CALENDAR_SEARCH	0x01	Camera list Search start day	VNP_RESULT_DAY
Date search	DATA_SEARCH	0x02	Camera Object_ID Search start day	VNP_RESULT_TIME_LIST

Each search condition is identified using Sub-Command in VNP_HEADER. VMP application can request several channel information using one command.

Calendar Search : Getting the days information that have recorded data.

Date Search : Getting the time interval corresponding to recorded data interval within the selected day.

Time interval can be separated by cause of recording. The search result can have several recorded time interval, data length of response data cannot be fixed. The response data length can be got in Data length field of VNP Header.

2.3.7.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	0x0000 ~ 0xffff

Direction	0x01
Command_ID	0x06 (CMD_DATA_SEARCH)
Command_sub	0x01 or 0x02
Command_type	0x00
Length	sizeof(VNP_QUERY_TIME)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	Result Code
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_query_tag
{
    unsigned long  record_type;        //0x00
    time_t  time_start;                //month,day search
    time_t  time_end;                  //only event
    unsigned short count;              // 0x01
    unsigned char Object_ID[count][10];
} _PACKED(VNP_QUERY_TIME);
```

record_type : not used. This field must be set to 0.

time_start : start time for search. This value has time_t format(4-byte).

In case of calendar search, this time value must be set to 00h 00m 00sec, 1'st day of each month

time_end : end time for search. This value has time_t format(4-byte).

This value must set as time value larger than time_start value and within the same day.

count : This field must be set to 1.

Object_ID[10] : Object_ID of camera for search

2.3.7.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x06 (CMD_DATA_SEARCH)
Command_sub	0x01 of 0x02
Command_type	0x00
Length	Different according to Command_sub and result of search
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value to request value
reserved	0x00
Res_code	0x00.
Header CRC	0x00

In case of Command_sub set to CALENDAR_SEARCH(0x01), length value of response will be sizeof(VNP_RESULT_DAY) (32 bytes).

In case of Command_sub set to DATA_SEARCH(0x02), length value of response will be varied according to time-interval count of recorded data.

[Data Structure]

1) In case of calendar search

```
typedef struct vnp_result_day_tag
{
    unsigned char days[32];
} _PACKED (VNP_RESULT_DAY);
```

days[32] : set 1 if that day have recording data, set 0 if that day do not have recording data

day[0] : 1'st day, day[31] : 31th day, day[32] : always 0

Result of search is returned with 32-byte binary format. The value of days[0] is correspond to 1'st day of the month, the value of days[30] corresponding to 31'th day of the month. The value of days[31] is always 0.

If there is recorded-data within the day, days[] value of corresponding date set to 1, otherwise set to 0.

2) In case of Data search

```
typedef struct vnp_result_time_list_tag
{
    unsigned long      dev_count;
    VNP_RESULT_TIME   result_time[1];
} _PACKED (VNP_RESULT_TIME_LIST);
```

```
typedef struct vnp_result_time_tag
{
    unsigned char   Dev_Object_ID [10];
    unsigned char   Object_ID[10];
    char            Object_Name[20];
    unsigned long   count;
    VNP_TIMELINE    timeline[count];
} _PACKED (VNP_RESULT_TIME);
```

```
typedef struct vnp_timeline_tag
{
    time_t          time_start;
```

```

time_t      time_end;
unsigned int record_type;
} _PACKED (VNP_TIMELINE);

```

dev_count : 0x01

Dev_Object_ID: Object_ID of Device

Object_ID : Object_ID of that contains recording data.

Object_Name : camera (or channel) name

count : count of recording interval (number of event) in the requested time period

time_start : recording start time

time_end : recording end time

record_type : recording event type

Define	Value	Applied device
REC_TYPE_MOTION	0x00000002	
REC_TYPE_ALARM	0x00000004	
REC_TYPE_ALL	0X000000FF	

2.3.8 CMD_SEARCH_CONTROL

2.3.8.1 Description

This command is used to control playback of recorded data in the device.

Before using this command, Media Request command (with MEDIA_OPEN sub_command) must be sent to device.

Target camera is decided by media_session id.

Playback control contains backward play, forward play, pause and speed control.

2.3.8.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x0A (CMD_SEARCH_CONTROL)
Command_sub	0x00
Command_type	0x00
Length	0x00
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_search_control_tag
{
    unsigned char    cmd;
    char            speed;
    unsigned long    time_pos;           //be valid only if (mtype == search)

    unsigned long    mode;
    unsigned long    media_session_id;
} _PACKED (VNP_SEARCH_CONTROL);
```

cmd : command list is following

Command	Value
SEARCH_STOP	0x01
SEARCH_PLAY	0x02
SEARCH_BACK_PLAY	0x03
SEARCH_SEEK	0x04
SEARCH_STEP_FORWARD	0x05
SEARCH_STEP_BACKWARD	0x06

speed : speed can be set one of these values : 1, 2, 4, 8

If command is play or backward play, it is possible to select speed..

time_pos : this value is used only if the command is SEARCH_SEEK. This value has the time_t format.

Device must be pause after seek.

mode : always 0

media_session_id : media_session id to identify camera

2.3.8.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x0A (CMD_SEARCH_CONTROL)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_SEARCH_CONTROL)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_search_control_tag
{
    unsigned char    cmd;
    char            speed;
    unsigned long    time_pos;        //be valid only if (mtype == search)

    unsigned long    mode;            // NON_SYNC_MODE, SYNC_MODE
    unsigned long    media_session_id;
} _PACKED (VNP_SEARCH_CONTROL);
```

cmd : The same value of request

speed : The same value of request

time_pos : If there is recorded data at requested time after seek, this value has the same time_t value to the requested time. But if there are no recorded data, result of seek has the close position of requested time. And actual seek time is set to this value.

mode : always 1

media_session_id : media_session id to identify camera

2.3.9 CMD_TALK

2.3.9.1 Description

This command is used to start or stop sending audio data from VMS application to device.

Device must get audio data after receiving this command (sub-command = 0) and device output audio data to the speaker (or line-out)

Object identifies whether it start or stop receiving audio data from VMS application with Command_sub value.

Sub command can be set following value:

Audio stream data send via media connection.

Command_sub	Value	Description
TALK_START	0x00	Start sending audio data
TALK_STOP	0x01	Stop sending audio data

2.3.9.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x21 (CMD_TALK)
Command_sub	0x00 (TALK_START) or 0x01(TALK_STOP)
Command_type	0x00
Length	0x00
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff

v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

2.3.9.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x21 (CMD_TALK)
Command_sub	0x00 (TALK_START) or 0x01(TALK_STOP)
Command_type	0x01
Length	0x00
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	Result Code
Header CRC	0x00

2.3.9.4 Audio data format for talk command

- Before Encoding

Audio format : PCM

Sampling rate : 8KHz

Number of channel : Mono (1 channel)

Bits per sample : 16bits

Average bytes per second : 16,000 bytes

- After encoding

Audio format : ADPCM mu-law

Bits per sample : 8bits

Average bytes per second : 8,000 bytes

- Send data format & size

Send data length in the one packet : VNP Media Header + 1000 Byte data (after encoding)

2.3.10 CMD_ALARM_RESET

2.3.10.1 Description

This command is used to clear alarm state in device.

If device have no button to clear alarm state, VMS application can control alarm state by using this command.

2.3.10.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x22 (CMD_ALARM_RESET)
Command_sub	0x00
Command_type	0x00
Length	0x00
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

2.3.10.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x22 (CMD_ALARM_RESET)
Command_sub	0x00
Command_type	0x01
Length	0x00
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	Result Code
Header CRC	0x00

2.3.11 CMD_ALARM_OUT

2.3.11.1 Description

This command is used to control Alarm(Relay) out device of IP device.

This command is supported if the IP Camera's firmware version is equal or higher than following table.

Model	F/W version
SNC-B2315	Not Support
SNC-B5395	Not Support
SNC-C7478	1.01
SNC-C7225	1.01
SNC-C6225	1.01
SNC-M300	Not Support

2.3.11.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x67 (CMD_ALARM_OUT)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_ALARM_OUT_CONTROL)

Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char event_status;
} _PACKED (VNP_ALARM_OUT_CONTROL);
```

Object_ID : Object ID of target alarm-out (relay-out)

This ObjectID is made using MAC address. MAC address can be got by Camera's ObjectID.

Just copy the Camera's objected and modify the 1'st, 7'th and 8'th byte as following.

1	2	3	4	5	6	7	8	9	10
0x14	MAC Address						0x01	index	0x00

The 9'th byte is target alarm-out port number. This number starts from 1.

SCC-C7478 has 4 alarm-out ports. So, index value has the range 1~4.

event_status : Target status (On/Off)

```
typedef enum
{
    OSTATUS_OFF = 0x0,
    OSTATUS_ON,
} VNPE_OSTATUS;
```

2.3.11.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x67 (CMD_ALARM_OUT)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_ALARM_OUT_CONTROL)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char event_status;
} _PACKED (VNP_ALARM_OUT_CONTROL);
```



All the fields are the same to request data structure.

2.3.12 CMD_VERSION

2.3.12.1 Description

Get the firmware version of device.

This command is supported by IP Camera with firmware version 2.0 or higher.

2.3.12.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x64 (CMD_VERSION)
Command_sub	0x00
Command_type	0x00
Length	0x00
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

2.3.12.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x64 (CMD_VERSION)
Command_sub	0x00
Command_type	0x01
Length	sizeof(VNP_VERSION)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	Result Code
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_version_tag
{
    unsigned char    version[21]
} _PACKED (VNP_VERSION);
```

version : firmware version information. Null-terminated string.

firmware version information has following format.

Vx.xx-yyyy.mm.dd

x.xx : FW version, **yyyy** : year, **mm** : month, **dd** : date

Copyright © 2008-2009, **SAMSUNG Electronics Co., Ltd.** All Rights reserved.

2.3.13 CMD_CAMERA_OSDMENU

2.3.13.1 Description

This command is used to control the camera OSD menu.

Detailed operation is decided by Command_sub value.

[Command_sub value]

Command	Value
MENU_UP	0x01
MENU_DOWN	0x02
MENU_LEFT	0x03
MENU_RIGHT	0x04
MENU_ENTER	0x05
MENU_ON	0x06
MENU_OFF	0x07
MENU_CANCEL	0x08

2.3.13.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x0C (CMD_OSD_MENU)
Command_sub	0x01 ~ 0x08
Command_type	0x00

Length	sizeof(VNP_OSD_UNIT)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char index; //hex
    char name[20];
} _PACKED (VNP_PTZ_UNIT);
```

Object_ID : Object ID of target camera

index : 0

name : null

2.3.13.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.

Sequence	The same value with request value
Direction	0x02
Command_ID	0x0C (CMD_OSD_MENU)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_OSD_UNIT)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char index; //hex
    char name[20];
} _PACKED (VNP_OSD_UNIT);
```

Object_ID : Object ID of target camera

index : 0

name : null

2.3.14 CMD_PTZ

2.3.14.1 Description

This command is used to control the direction and zoom of PTZ camera.
Detailed operation is decided by Command_sub value.

[Command_sub value]

Command	Value
VNP_PTZ_UP	0x01
VNP_PTZ_DOWN	0x02
VNP_PTZ_LEFT	0x03
VNP_PTZ_RIGHT	0x04
VNP_PTZ_UP_LEFT	0x05
VNP_PTZ_UP_RIGHT	0x06
VNP_PTZ_DOWN_LEFT	0x07
VNP_PTZ_DOWN_RIGHT	0x08
VNP_PTZ_ZOOM_IN	0x09
VNP_PTZ_ZOOM_OUT	0x0A
VNP_PTZ_STOP	0x0B

2.3.14.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff

Direction	0x01
Command_ID	0x0D (CMD_PTZ)
Command_sub	0x01 ~ 0x0B
Command_type	0x00
Length	sizeof(VNP_PTZ_VALUE)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    int Value; // 0~100, -1
} _PACKED (VNP_PTZ_VALUE);
```

Object_ID : Object ID of target camera

Value : amount of movement. (0~100). If this value is set to '-1', camera continue to move until receive the stop command

2.3.14.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in

Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x0D (CMD_PTZ)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_PTZ_VALUE)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    int Value; // 0~100, -1
} _PACKED (VNP_PTZ_VALUE);
```

All the fields are the same to request data structure.

2.3.15 CMD_PTZ_MOVE

2.3.15.1 Description

This command is used to move PTZ using absolute coordinates.

This command uses spherical coordinates.

VMS SW can get the current PTZ position using SUBCMD_PTZ_GET_POS.

[Command_sub value]

Command	Value
SUBCMD_PTZ_GET_POS	0x01
SUBCMD_PTZ_SET_POS	0x02

2.3.15.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x2E (CMD_PTZ_MOVE)
Command_sub	0x01 or 0x02
Command_type	0x00
Length	0 or sizeof(VNP_PTZ_MOVE)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff

reserved	0x00
Res_code	0x00.
Header CRC	0x00

1) SUBCMD_PTZ_GET_POS

[Data Structure]

None (Length field in the VNP header = 0x00)

2) SUBCMD_PTZ_SET_POS

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    unsigned short int TargetOperate; // indicate the operation using each Bit
    short int pan_int; // 0 ~ 360
    short int pan_dec; // 0 ~ 99
    short int tilt_int; // 0 ~ 180
    short int til_dect; // 0 ~ 99
    int zoom; // 1 ~ 432
} _PACKED (VNP_PTZ_MOVE);
```

Object_ID : Object ID of target camera

TargetOperate : This value indicate the operation using each bit. Each operation is done only if the bit is set to 1.

ex) If this value is set to 0x0005, camera operate Pan and Tile function and ignore zoom operation.

Value	Operation
0x0001	Pan operation
0x0002	Tilt operation
0x0004	Zoom operation

pan_int, pan_dec : This value defines the pan value of target position. Pan_int defines the integer part of target pan value. Pan_dec defines the decimal part of target pan part. If the target pan value is 100.98, VMS S/W set the pan_int value to 100 and pan_dec value to 98.

tilt_int, tilt_dec : This value defines the tilt value of target position. Tilt_int defines the integer part of target

tilt value. Tilt_dec defines the decimal part of target tilt part. If the target tilt value is 70.38, VMS SW set the tilt_int value to 70 and tilt_dec value to 38.

zoom : Zoom value

SCC-C7478 have the Zoom range (1~432)

SNC-C6225 and SNC-C7225 have the Zoom range (1~100)

2.3.15.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x2E (CMD_PTZ_MOVE)
Command_sub	0x01 or 0x02
Command_type	0x00
Length	sizeof(VNP_PTZ_MOVE)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    unsigned short int TargetOperate; // 0x00
    short int pan_int; // 0 ~ 360
    short int pan_dec; // 0 ~ 99
    short int tilt_int; // 0 ~ 180
    short int til_dect; // 0 ~ 99
    int zoom; // 1 ~ 432
} _PACKED (VNP_PTZ_MOVE);
```

Object_ID : Object ID of target camera

TargetOperate : This value is fixed to 0x0000.

pan_int, pan_dec : This value defines the pan value of current position. Pan_int defines the integer part of target pan value. Pan_dec defines the decimal part of target pan part. If the current pan value is 100.98, the pan_int value is set to 100 and pan_dec value to 98.

tilt_int, tilt_dec : This value defines the tilt value of current position. Tilt_int defines the integer part of target tilt value. Tilt_dec defines the decimal part of target tilt part. If the current tilt value is 70.38, the tilt_int value is set to 70 and tilt_dec value is set to 38.

zoom : Zoom value

SCC-C7478 have the Zoom range (1~432)

SNC-C6225 and SNC-C7225 have the Zoom range (1 ~ 100)

**** When the command_sub is 'SUBCMD_PTZ_SET_POS'.**

All the fields are the same to request data structure.

2.3.16 CMD_PRESET

2.3.16.1 Description

This command is used for the preset control.

Detailed operation is decided by Command_sub value.

[Command_sub value]

Command	Value
VNP_PRESET_MOVE	0x01
VNP_PRESET_ADD	0x02
VNP_PRESET_DELETE	0x03
VNP_PRESET_DELETE_ALL	0x04
VNP_PRESET_GETLIST	0x05

2.3.16.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x0F (CMD_PTZ_PRESET)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_PTZ_UNIT)
Is_Encrypted	0x00
Reserved	0x00

v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char Index;
    char name[20];
} _PACKED (VNP_PTZ_UNIT);
```

Object_ID : Object ID of target camera

Index : The index of the list. This value is used if command_sub is 'add, move, delete'.

This value start from 0

name : Null-terminated string. Preset name(alias) can be set using this value

2.3.16.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02

Command_ID	0x0F (CMD_PTZ_PRESET)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_PRESET_UNIT) or sizeof(VNP_PRESET_LIST)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

1) When the command_sub is 'move, delete add'.

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char Index;
    char name[20];
} _PACKED (VNP_PRESET_UNIT);
```

All the fields are the same to request data structure.

2) When the command_sub is 'add, delete, get list'

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char count;
    VNP_PRESET_LIST list[20];
} _PACKED (VNP_PRESET_LIST);
```

Object_ID : Object ID of target camera.

count : the count of valid preset list value.

list : detail information of each list. Object_ID field of this value is not used.

2.3.17 CMD_AUTOPAN

2.3.17.1 Description

This command is used for the autopan operation of the camera.

Detailed operation is decided by Command_sub value.

[Command_sub value]

Command	Value
VNP_AUTOPAN_START	0x06
VNP_AUTOPAN_STOP	0x07

2.3.17.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x10 (CMD_PTZ_AUTOPAN)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_AUTOPAN_UNIT)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00

Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char Index;
    char name[16];
} _PACKED (VNP_AUTOPAN_UNIT);
```

Object_ID : Object ID of target camera

Index : 0

name : NULL

2.3.17.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x10 (CMD_PTZ_AUTOPAN)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_AUTOPAN_LIST)

Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char count;
    VNP_PTZ_LIST list[20];
} _PACKED (VNP_AUTOPAN_LIST);
```

Object_ID : Object ID of target camera.

count : 0

list : all fields are set to 0.

2.3.18 CMD_SCAN

2.3.18.1 Description

This command is used for the scan operation of the camera.
Detailed operation is decided by Command_sub value.

[Command_sub value]

Command	Value
VNP_ACTION_START	0x06
VNP_ACTION_STOP	0x07

2.3.18.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x11 (CMD_PTZ_SCAN)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_PTZ_UNIT)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00

Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char Index;
    char name[16];
} _PACKED (VNP_PTZ_UNIT);
```

Object_ID : Object ID of target camera

Index : 0

name : NULL

2.3.18.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x11 (CMD_PTZ_SCAN)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_PTZ_LIST)

Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char count;
    VNP_PTZ_LIST list[20];
} _PACKED (VNP_PTZ_LIST);
```

Object_ID : Object ID of target camera.

count : 0

list : all fields are set to 0.

2.3.19 CMD_PATTERN

2.3.19.1 Description

This command is used for the pattern operation of the camera.

Detailed operation is decided by Command_sub value.

[Command_sub value]

Command	Value
VNP_ACTION_START	0x06
VNP_ACTION_STOP	0x07

2.3.19.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x12 (CMD_PTZ_PATTERN)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_PTZ_UNIT)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00

Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char Index;
    char name[16];
} _PACKED (VNP_PTZ_UNIT);
```

Object_ID : Object ID of target camera

Index : 0

name : NULL

2.3.19.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x12 (CMD_PTZ_PATTERN)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_PTZ_LIST)

Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_ptz_unit_tag
{
    unsigned char Object_ID[10];
    char count;
    VNP_PTZ_LIST list[20];
} _PACKED (VNP_PTZ_LIST);
```

Object_ID : Object ID of target camera.

count : 0

list : all fields are set to 0.

2.3.20 CMD_AUTHORITY

2.3.20.1 Description

This command is used to get the privilege to search and playback. IP-Camera cannot support for multi user to playback. VMS S/W must get the privilege to search and playback using this command and release this privilege after end of playback using this command

This command is supported if the IP Camera's firmware version is equal or higher than following table.

Model	F/W version
SNC-B2315	Ver. 2.01
SNC-B5395	Ver. 2.01
SNC-C7478	Ver. 1.01
SNC-C6225	Ver. 1.01
SNC-C7225	Ver. 1.01
SNC-M300	Ver. 2.01

2.3.20.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x65 (CMD_AUTHORITY)
Command_sub	0x00
Command_type	0x00
Length	0x00

Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

2.3.20.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x65 (CMD_AUTHORITY)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_AUTHORITY)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	Result Code
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_authority_tag
{
    unsigned char    authority_type; // 0x00 – none, 0x01 - Search
    unsigned char    action_type;   // 0x00 – Get, 0x01 - Release
} _PACKED(VNP_AUTHORITY);
```

authority_type : To search information of the recorded data, this value must be set to 0x01

action_type : To get the privilege of search, this value must be set to 0x00. After finishing search or playback, send this command with 0 to release the privilege of search.

2.3.21 CMD_GET_OBJECT_INFO

2.3.21.1 Description

This command is used to get the information of file upload.

Command_Sub is fixed to 0x01.

Command_sub	Value	Description
SUBCMD_UPLOAD_SERVER_INFO	0x01	Get information of Upload server to upload new Firmware

This command is supported if the IP Camera's firmware version is equal or higher than following table.

Command_sub	Model	F/W version
SUBCMD_UPLOAD_SERVER_INFO	SNC-B2315	2.03
	SNC-B5395	2.03
	SNC-C7478	1.01
	SNC-C7225	1.01
	SNC-C6225	1.01
	SNC-M300	2.03

2.3.21.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x23 (CMD_GET_OBJECT_INFO)
Command_sub	0x01
Command_type	0x00
Length	0x00
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

Length field in the VNP Header : 0x00

2.3.21.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02

Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x23 (CMD_GET_OBJECT_INFO)
Command_sub	0x01
Command_type	0x00
Length	sizeof(VNP_MEDIA_INFO)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	Result Code
Header CRC	0x00

[Response Data Structure]

VNP Header Length : sizeof(VNP_MEDIA_INFO)

[Data Structure]

```
typedef struct vnp_media_info_tag
{
    unsigned char    protocol;
    unsigned char    ip_address[4];
    unsigned int     port;
} _PACKED (VNP_MEDIA_INFO);
```

protocol : Current setting protocol value that is used for sending stream data.

TCP (0), UDP-unicast (1), UDP-multicast (2)

ip_address : This value is valid only if protocol type is UDP-multicast.

If protocol set to UDP-unicast or TCP, VMS application must use the same IP address with Device connection IP.

Copyright © 2008-2009, **SAMSUNG Electronics Co., Ltd.** All Rights reserved.



port : Port number to connect. VMS application uses this port number regardless protocol setting.

2.3.22 CMD_HEARTBEAT

2.3.22.1 Description

This command is used to keep connection also. Client S/W must send this command every 10 seconds. If device doesn't receive this command more than 1 minute, device may force to disconnect.

This command is supported if the IP Camera's firmware version is equal or higher than following table.

Model	F/W version
SNC-B2315	Not Support
SNC-B5395	Not Support
SNC-C7478	1.01
SNC-C7225	1.01
SNC-C6225	1.01
SNC-M300	Not Support

2.3.22.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x28 (CMD_HEARTBEAT)
Command_sub	0x00
Command_type	0x00

Length	0x00
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

[Data Structure]

None

2.3.22.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x28(CMD_HEARTBEAT)
Command_sub	0x00
Command_type	0x00
Length	0
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00

Res_code	Result Code
Header CRC	0x00

[Response Data Structure]

None

2.3.23 CMD_DEVICE_INFO

2.3.23.1 Description

This command is used to get the information of streaming and uploading connection. This command can substitute the CMD_MEDIA_INFO and CMD_VERSION command

This command is supported if the IP Camera's firmware version is equal or higher than following table.

Model	F/W version
SNC-B2315	Not Support
SNC-B5395	Not Support
SNC-C7478	1.01
SNC-C7225	1.01
SNC-C6225	1.01
SNC-M300	Not Support

2.3.23.2 Request Data Structure

[VNP Header Value]

Field	Request Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with

	response of log in.
Sequence	0x0000 ~ 0xffff
Direction	0x01
Command_ID	0x2A (CMD_DEVICE_INFO)
Command_sub	0x00
Command_type	0x00
Length	0x00
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00.
Header CRC	0x00

2.3.23.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x2A (CMD_DEVICE_INFO)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_DEVICE_INFO),
Is_Encrypted	0x00

Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00
Res_code	Result Code
Header CRC	0x00

[Response Data Structure]

[Data Structure]

```

typedef struct VNP_Resolution__tag
{
    unsigned short    Width;
    unsigned short    Height;
} _PACKED(VNP_RESOLUTION);

typedef struct VNP_VideoStatue_tag
{
    unsigned char     Object_ID[10];
    unsigned char     Codec_id; // Index value of codec. this value start from 1
    unsigned short    FrameRateValue; // Real frame rate value * 1000
    unsigned char     QualityValue; // current quality value. 1 is the worst quality.
    VNP_RESOLUTION    ResolutionValue; // current resolution vlaue
} _PACKED(VNP_VIDEO_STATUS);

typedef struct vnp_media_info_tag
{
    unsigned char     protocol; // The same information of CMD_MEDIA_INFO
    unsigned char     ip_address[4];
    unsigned int      port;
} _PACKED (VNP_MEDIA_INFO);

typedef struct VNP_DeviceInfo_tag
{
    unsigned char     FW_version[21]; // F/W version. Null-terminated string
    VNP_MEDIA_INFO    MediaInfo; //
    unsigned int      Upload_Port; // port number for File Upload

```

```
    unsigned int    Download_Port; // Not used. this value is fixed to 0
    unsigned int    VideoCount;    // Video Source count. Samsung IP-Camera is fixed to 2
    VNP_VIDEO_STATUS VideoStatus[VideoCount];
} _PACKED(VNP_DEVICE_INFO);
```

VNP_DEVICE_INFO :

FW_version : F/W version. Null-terminated string.

VNP_MEDIA_INFO : The same information of CMD_MEDIA_INFO.

Upload_port : port number for File Upload

Download_port : Not used. this value is fixed to 0.

VideoCount : Video Source count. Samsung IP-Camera is fixed to 2

VNP_VIDEO_STATUS : Current setting information of video object

Object_Id : Object ID of Video object

Codec_Id : Index value of codec. this value start from 1

FrameRateValue : Real frame rate value * 1000

QualityValue : current quality value. 1 is the worst quality

ResolutionValue : current resolution vlaue

2.3.24 CMD_GET_OBJECTLIST_ADV

2.3.24.1 Description

This command is used to get Object List and status of each port. This command can substitute the CMD_GET_OBJECTLIST command.

Get the Object_ID and capability of each object to control the object.

By using this command, VMS application can get the structure of object and object's capability.

VMS application has to use Object_ID which is acquired by using this command when controlling the each object. Also VMS application can use functions which are described in capability.

This command is supported if the IP Camera's firmware version is equal or higher than following table.

Model	F/W version
SNC-B2315	Not Support
SNC-B5395	Not Support
SNC-C7478	1.01
SNC-C7225	1.01
SNC-C6225	1.01
SNC-M300	Not Support

2.3.24.2 Request Data Structure

[VNP Header Value]

Field		Request Value
Major version	1	0x01
Minor version	1	0x00
Object_Type	1	0x02
Model_ID	2	Device Model ID. The same value with response of log in
Object_ID		Object_ID of device. The same value with response of log in.
Device_Session_ID		Device session ID for connection. The same value with response of log in.
Sequence		0x0000 ~ 0xffff
Direction		0x01

Command_ID		0x2B (CMD_GET_OBJECTLIST_ADV)
Command_sub		0x00
Command_type		0x00
Length		0x00
Is_Encrypted		0x00
Reserved		0x00
v_sequence		0x0000 ~ 0xffff
v_requestor		0x0000 ~ 0xffff
reserved		0x00
Res_code		0x00.
Header CRC		0x00

2.3.24.3 Response Data Structure

[VNP Header Value]

Field	Response Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log in
Object_ID	Object_ID of device. The same value with response of log in.
Device_Session_ID	Device session ID for connection. The same value with response of log in.
Sequence	The same value with request value
Direction	0x02
Command_ID	0x2B (CMD_GET_OBJECTLIST_ADV)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_OBJECTS_ADV)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	The same value with request value
v_requestor	The same value with request value
reserved	0x00

Res_code	Result Code
Header CRC	0x00

[Data Structure]

```

typedef struct VNP_OBJECTs_Adv_tag
{
    unsigned short    count;
    VNP_AUDIO_INFO    sListenInfo;
    VNP_AUDIO_INFO    sTalkInfo;
    VNP_OBJECT_ADV    Object[count];
} _PACKED(VNP_OBJECTS_ADV);
typedef struct VNP_AudioInfo_tag
{
    unsigned char     Codec_Type;
    unsigned char     BitPerSample;
    unsigned char     ChannelCount;
    unsigned short    SampleRate;
    unsigned short    PacketSize;
} _PACKED(VNP_AUDIO_INFO);
typedef struct VNP_OBJECT_Adv_tag
{
    unsigned char     Object_type;
    unsigned char     Object_ID[10];
    unsigned char     Object_Parent[10];
    unsigned char     Object_Name[41];
    unsigned char     Possibility_Status; // this value is fixed to 1
    unsigned int      Object_cap;
    unsigned int      Ptz_cap;
    unsigned int      Event_stauts; // Alarm-in, Alarm-out 상태
    unsigned int      Playback_cap;
} _PACKED(VNP_OBJECT_ADV);

```

Count : Number of total Objects

Object_type : [Defined constant for Object Type](#)

Object Type	Value
IP Camera Device	2
Analog Camera Part	4
Video	6
Alarm In	5
Alarm Out (Relay Out)	20 (0x14)

Object_ID : Object_ID of current object. (10 byte)

Object_Parent : Parent object's Object_ID.

Object_Name : Name of the object , Null-terminated string

Possibility_Status : always 1. Current state (possible to use or not) of Object

```
typedef enum
{
    OSTATUS_OFF = 0x0,
    OSTATUS_ON,
} VNPE_OSTATUS;
```

Object_cap : Object Capability. Capability means the functions that device can support.

If device can transfer live image to the VMS application, this device has the capability of 'Live monitoring'.

In this case, Object capability has the value of 'Live monitoring' (0x01).

Device has capability of bit-wise ORing value of each capability value that device can provide.

This value is possible for each Object Type.

Object Type	Object Value	Object_cap	Description
IP Camera Device	2	0x0000 0000	None
Analog Camera Part	4	0x0000 0000	None
Video	6	0x0000 0001	Live Monitoring
		0x0000 0002	Playback from SD memory
		0x0000 0004	possible to get audio (Device → VMS)
		0x0000 0008	possible to send audio (VMS → Device)
		0x0000 0010	possible to control PTZ functions
Alarm In	5	0x0010 0000	possible to clear the alarm-in state

			(use CMD_ALARM_RESET(2.3.10))
Alarm Out (Relay Out)	20 (0x14)	0x0020 0000	possible to control the alarm-out (use CMD_ALARM_OUT(2.3.11))

Ptz_cap : This value indicates the capability of PTZ function. Each bit matches the PTZ operation as following table. This value is valid if PTZ control bit of Object_cap is 1 (0x0000 0010).

Value	Description
0x0000 0001	CAP_PTZ_BASIC (Zoom In/Out, direction control, menu)
0x0000 0002	CAP_PTZ_POWER (Power PTZ)
0x0000 0004	CAP_PTZ_FUNCTION
0x0000 0008	CAP_PTZ_IRIS
0x0000 0010	CAP_PTZ_FOCUS
0x0000 0020	CAP_PTZ_BRIGHTNESS
0x0000 0040	CAP_PTZ_PAN
0x0001 0000	CAP_PTZ_TILT
0x0002 0000	CAP_PTZ_ZOOM
0x0004 0000	CAP_PTZ_PRESET
0x0008 0000	
0x0010 0000	CAP_PTZ_SCAN
0x0020 0000	CAP_PTZ_AUTOPAN
0x0040 0000	CAP_PTZ_PATTERN
0x0080 0000	

ex) 0x9D means (b'1001 1101)

The camera support BASIC operation, Function, Iris, focus, contrast.

CAP_PTZ_BASIC value is 1 if one or more value of CAP_PTZ_PAN, CAP_PTZ_TILT, CAP_PTZ_ZOOM is 1

Event_status : Current Status of each object.

This value indicates the current status of each event. This value has bit-wise or value of following table.

Each Object Type can be set following table value

Object Type	Object Value	Event_stauts	Description
IP Camera Device	2	0x0000 0007	VNP_EVENT_FAN_BROKEN
Analog Camera Part	4	0x0000 0000	None
Video	6	0x0000 0010	VNP_EVENT_MOTION
		0x0000 0001	VNP_EVENT_VIDEO_LOSS
		0x0000 0002	VNP_EVENT_ALARM

		0x0000 0008	VNP_EVENT_RECORDING
Alarm Out (Relay Out)	20 (0x14)	0x0000 000B	VNP_EVENT_ALARM_OUT

Playback_cap : This value indicates the capability of Playback function. Each bit matches the Playback operation as following table.

. This value is valid if Playback of Object_cap is 1 (0x0000 0002).

Value	Define	Description
0x0000 0001	CAP_PB_PLAY	1x play
0x0000 0002	CAP_PB_STOP	STOP
0x0000 0004	CAP_PB_PB	1x play backward
0x0000 0008	CAP_PB_SEEK	
0x0000 0010	CAP_PB_FF	Fast forward
0x0000 0020	CAP_PB_FB	Fast backward
0x0000 0040	CAP_PB_SF	Slow forward
0x0001 0000	CAP_PB_SB	Slow backward
0x1000 0000	CAP_PB_EXCLUSIVE	device support only exclusive playback

<Reference>

object list structure sample

1) support dual stream and 2 of alarm-in port, 2 of alarm-out port

IP Camera model

- └ Camera
 - └ Video 1
 - └ Video 2
- └ alarm-in 1
- └ alarm-in 2
- └ alarm-out 1
- └ alarm-out 2

2.4 Notification Protocol Definition

2.4.1 Event

2.4.1.1 Description

Device sends this event if some events are occurred.

There are two event type : Motion Detection and Alarm Input.

In case of alarm, device sends alarm notification with related camera information.

2.4.1.2 Notification Data Structure

[VNP Header Value]

Field	Value
Major version	0x01
Minor version	0x00
Object_Type	0x02
Model_ID	Device Model ID. The same value with response of log-in
Object_ID	Object_ID of device. The same value with response of log-in.
Device_Session_ID	Device session ID for connection. The same value with response of log-in.
Sequence	0x0000 ~ 0xffff
Direction	0x02
Command_ID	0x0B (CMD_DEVICE_EVENT)
Command_sub	0x00
Command_type	0x00
Length	sizeof(VNP_DEVICE_EVENT)
Is_Encrypted	0x00
Reserved	0x00
v_sequence	0x0000 ~ 0xffff
v_requestor	0x0000 ~ 0xffff
reserved	0x00
Res_code	0x00
Header CRC	0x00

[Data Structure]

```
typedef struct vnp_device_event_tag
{
    short int event_duration;
    unsigned short int event_type;
    char object_id[10];          //camera object id
    char object_name[41]; //camera object name
    char alarm_name[21];       //alarm object name
    int alarm_port;            //alarm port
} _PACKED (VNP_DEVICE_EVENT)
```

event_duration : 0x00 (Not used)

event_type :

Object Type	Object Value	Object_cap	Description
IP Camera Device	2	0x0007	EVENT_FAN_BROKEN
		0x0107	EVENT_FAN_BROKEN_END
Analog Camera Part	4	0x0000	None
Video	6	0x0000	EVENT_MOTION
		0x0001	EVENT_VIDEOLOSS
		0x0002	EVENT_ALARM
Alarm Out (Relay Out)	20 (0x14)	0x000B	VNP_EVENT_ALARM_OUT_SATART
		0x010B	VNP_EVENT_ALARM_OUT_END

Object_ID : Object_ID of camera that has event

Object_Name : camera's name that has event

alarm_name :

1) If event_type is EVENT_ALARM

This field has alarm name like Alarm1, Alarm2, Alarm3, Alarm4 ...

The other field must be set to 0

2) If event_type is EVENT_PTZ_CHANGE

First 4 byte indicates the changed PTZ_CAPABILITY value.

The other field must be set to 0.

alarm_port : port number that alarm event occurs.

value : 0 or 1