Interface

Long River
Vision Systems

For Network Video Recorder Development

Prepared by: P.Shi

Created: 08/14/2016
Last Updated: 10/01/2016

Rev: 1

Revision history

Release No.	Date	Comment
Rev. 1	10/01/2015	Initial draft

TABLE OF CONTENTS

1. INTRODUCTION

Long River Video System's NVR is a Web UI based DVR system that can support up to 8 channels Audio/Video recordings, live streaming and remote playback functions. It can seamlessly work with most popular browsers on any computing devices.

1.1. Scope

The purpose of this document illustrates how to integrate NVR device from system level. The potential reader of this document can be system engineers that will integrate NVR device or APP engineers that attempts to develop specific Apps on different mobile devices to live view and remote control NVR devices, as well as field support engineers for maintenance or troubleshooting etc.

This document mainly involves :

- **Websockets communication.**

All LRVS's NVR device is based on HTML5 Websockets . This document will illustrate how to set up Websockets' communication with NVR device. It is also good for users who want to develop applications or apps without general web browsers.

- **Web APIs**

The rest sections illustrate detail web interfaces of NVR devices to communicate, such as live view , recording control etc.

Note: the following portion is not covered in this document and will be covered in latter released documents.

- RTSP/RTP Communication.
- PTZ camera Pelco camera communication
- Video/Audio frame synchronization.
- Video clips playback, retrieve, disk management.
- Advanced administration including software updating, troubleshooting etc.

1.2. Acronyms and Abbreviations

AAC-LC (Advanced Audio Coding – Low Complexity)
CIF Video resolution (352x288 for PAL; 352x240 for NTSC)
D1 video resolution (704x576 pixels for PAL; 704x480 pixels for NTSC)
DVR- Digital Video Recorder
FPS Frame per second
G.711 Pulse Code Modulation
G.726 Adaptive Differential Pulse Code Modulation
H.264/MPEG-4 Part 10 or **AVC**(Advanced Video coding) a video compression standard defined by ISO/IEC 14496-10

HD1 Half D1 (352x576 pixels for PAL; 352x480 pixels for NTSC)
HDLC High-level Data Link Control
LAN Local Area Network
LRVS Long River Video Systems
NVR Network Video Recorder

Disclaimer

The LRVS does not assume any liability arising out of the application or use of any products, or software described herein. This documentation is subject to change without notice. Please visit http://www.lrvs.net for the latest version.

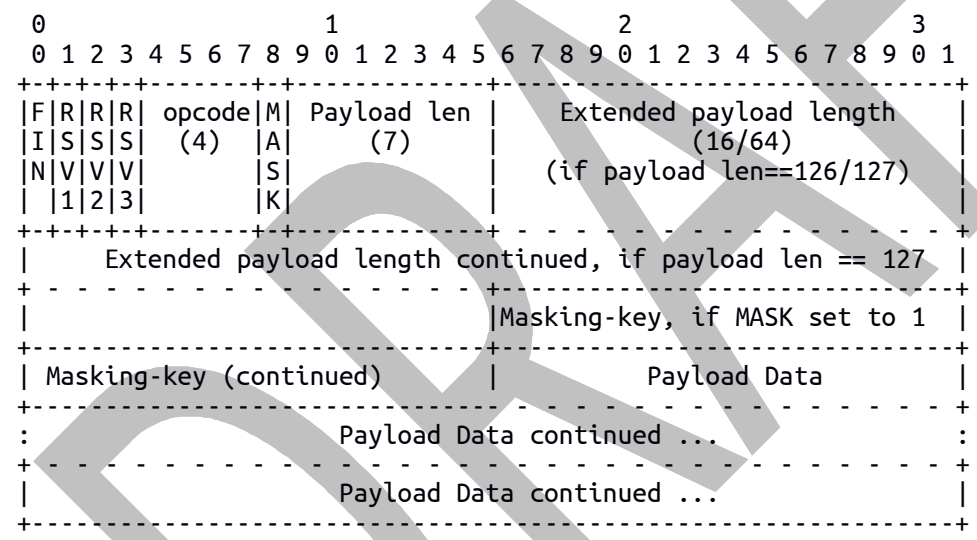
2. OVERVIEW

2.1. Websockets Establishment

All NVR device communication (Liveview and Command) is based on HTML5 Websockets. To talk with NVR device, a HTML Websockets must be established first, please first refer to (RFC6455): websockets communications on how to set up Websockets connection. This document has assumed the NVP client has connected NVR server through Websockets.

After WebSockets between NVR client and Server is established all network communication is through Websockets binary mode.

Note: To simply Server/Client side computing, NVR's websockets do not adopt Mask bit for data Communion, i.e. Mask bit = 0, illustrated as below figure.



If the message Payload is less than 125 byte, the Wbsockets header will be 2 bytes long. Almost all NVR remote commands are less than 125 bytes and fallen into this category.

Field:	**Websockets binary header**	**PAYLOAD**		
	F (0x82)	**Mask bit**	**LEN**	
Length:	8 bit	1(0)	6 bit	Variable

If the message payload is large than 125 bytes (>=125bytes), websockets header will become 4 bytes long.

Field:	**Websocket binary header**	**PAYLOAD**			
	F (0x82)	**Mask bit**	**LEN**	**Extended Length**	
Length:	8 bit	(0)	0x7E	16 bits	Variable

For example 256 bytes binary message in a single unmasked frame is :

* 0x82 0x7E 0x0100 [256 bytes of binary data]

In Current NVR solution, because of TCP socket MTU limitation, to balance network bandwidth and buffer, NVR limits all frames to maximum 1024bytes for HTTP live streaming and 1500bytes for RTSP live streaming.

2.2. Live Streaming Connections

When the Websockets between client and server(NVR device) get established, the NVR will first send a media header frame followed with media data frames:

Media header frame:

MEDIA_HEADER (Video)										
Field:	Start	**PAYLOAD**								
	F (0xE0)	**M**	**T**	**CS**	**LEN (9)**	**VT**	**W**	**H**	FPS	TERM
Length : bytes	1	1	1	4	1	1	2	2	2	2

F- Media Header frame, always 0xE0
M– Media Type:
0x01 - Video Streaming;
0x02 - Audio Streaming;
0x03 – A/V mix streaming
T- Total Channels(Cameras)
CS- Channel recording State (bitwise – host order)
bit0 - Channel 1, 1: recording; 0: not recording
bit1 – Channel 2,
…
bit31 – Channel 32 (if exist)
LEN – Media header segment length (including terminator: 0x0d 0x0a)
VT - Video Type
W - Picture Width
H - Picture Width
FPS – Frame rate (Frames per seconds)

TERM – Terminator , always 0x0d 0x0a

Note: Media head frame will be resent under the current scenarios:

1. Media status get changes such as recording state etc (Indications)
2. Upon Client control request commands(Responses)

MEDIA_HEADER (Audio)												
Field:	Start	**PAYLOAD**										
	F (0xE0)	**M**	**T**	**CS**	**LEN (14)**	**ACh**	**AT**	**SR**	**BR**	BPS	AFS	TERM
Length: bytes	1	1	1	4	1	1	1	2	4	2	2	2

ACh – Audio Channel number
AT - Audio Type
SR - Sample rate
BR - Bit rate
BPS - Bits per seconds
AFS - Audio Framesize
TERM - Terminator , always 0x0d 0x0a

Media data frame:

HTTP live streaming data packets are sent right after media header frame; The maximum frame length is 1024 bytes. The data frame esp video frame whose length is larger than 1024 bytes will be split to multiple segments .

DATA_FRAME										
Field:	Start	**PAYLOAD**								
	F (0xF0)	**T**	**S**	**LEN**	**ID**	**Seq**	**Seg**	**TSeg**		TERM
Length: bytes	1	1	1	2	1	1	2	2	variable	2

F- Flag of frame, always 0xF0
T- Frame Type:
0x01 - Video Streaming;
0x02 - Audio Streaming;
S- Segment Type:
0x01- First Segment;
0x02 - Last Segment;
0x00 - Middle Segment;
Len- Current Segment length
ID - Channel ID
Seq - Sequence number , increment 1
Seg - Current Seqment Index
TSeg - Total Segments: For last segment of one frame: Seg = TSeg -1;
TERM – Terminator , always 0x0d 0x0a

3. NVR DEVICE CONTROL PROTOCOLS

NVR device control protocol is an HDLC-based protocol (RFC1662 and RFC1958) but with more simplicity.

NVR device control protocol uses the same socket with HTTP live streaming and “steal” bandwidth of Live streaming . Control commands are transmitted with same Websockets (TCP socket),the simplified solutions:

The contents of an HDLC frame are shown in the following table:

Flag	Address	Control	Information	FCS	Flag
8 bits	8 or more bits	8 or 16 bits	Variable length, 0 or more bits	16 or 32 bits	8 bits

CMD_FRAME								
Field:	Start	PAYLOAD						
	F	**A**	**S**	**P**	**LEN**	**CMD**		**TERM**
Length:	1	1	1	1	1	1	Variable	2

Where LEN = PAYLOAD length + 2 (TERM)

The following table describes the values required in the CMD_FRAME in order ensure successful processing of command messages.

3.1. Command frame definition

Table : NVR command Item Summary

Data Item	Command Item Value
F	Flag of frame , it always starts with ‘$’ i.e. 0x24 (hex), note: it is not 0x7E of HDLC
A	**(Dest) Address Field:** it is a single octet. **0x00:** message direction: NVR client -> NVR server: NVR_Server :0x00 **0xFF:** message direction: NVR server -> NVR Client: NVR_Client: 0xFF **Note:** NVR client address can be 0x01-9xFE, this address is allocated by NVR server. NVR client can communicate with address 0xFF without association.
S	**Sequence number : increase by 1**
P	**Protocol type:** **0x00: normal NVR command set** **0X01: PTZ Pelco protocol set**
Command	Message ID: Message identifier. See Table for all command definitions. LSB sent first
LEN	**Length:** Command Length

Data Item	Command Item Value
Payload	**Message data:** Refer to the message details provided in Section x
TERM	**Message Terminator:** Always the carriage return/line feed characters in sequence (0x0D 0x0A). Note: HDLC's CRC is not used and two fixed character used instead.

4. INTERFACE PROTOCOL

The Communion service in Client-Server mode used in this document.

PDU type	Description
Request/Command	This PDU is sent by the client device and/or it expects a response
Response	Sent by the NVR server in response to a request PDU
Indication	NVR server indicates a client of new information and expect to receive a confirmation.

4.1 System command 1-49

0x01: GET-TIME_REQ()
GET-TIME.request parameters - None

Interface format:

Offset (unit: unsigned char)	Description
0 1	Frame flag must be '$' (0x24) Length(not including itself) = 0x08; Server address: 0x00;message from client to server
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =1
5	CMD-GET-TIME.request command token
6-7	Terminator : 0x0d 0x0a

The CMD-GET-TIME.request primitive is generated by the NVR client and queries NVR device(server)'s local system time.

0x02: GET_TIME_RESP(time_t)
GET-TIME.response parameters

Name	Type	Valid range	Description
tm_sec	Integer	0x00-0x4B	0-59 seconds
tm_min	Integer	0x00-0x4B	0-59 minutes
tm_hour	Integer	0x00-0x19	0-23
tm_mday	Integer	0x0 - 0x1F	1-31 day
tm_mon	Integer	0x0 - 0x0B	0-11 month
tm_year	Integer	0x64 - 0xC8	100(1900+100)–200 (1900+200)
tm_wday	Integer	0x0 - 0x06	0-6

Interface format:

Offset (unit: unsigned char)	Description

Offset (unit: unsigned char)	Description
0	Frame flag must be '$' (0x24) Length(not including itself) = 0x08;
1	Client address: 0x01-0xFF;message from server to client.
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =13
5	GET-TIME.response command token: 0x02
6-7	tm_min: LSB first; MSB: 0x00
7-8	tm_hour: LSB first; MSB: 0x00
9-10	tm_mday: LSB first; MSB: 0x00
11-12	tm_mon: LSB first; MSB: 0x00
13-14	tm_year: LSB first
15-16	tm_wday:LSB first; MSB : 0x00
17-18	Terminator : 0x0d 0x0a

The G**ET-TIME.response** is generated by the NVR device (server) and issued to the NVR client in response to GET-TIME.request primitive.

0x03: SET-TIME.request (time_t)
SET-TIME.request parameters: time_t data structure, see GET-TIME.response parameters.

Interface format:

Offset (unit: unsigned char)	Description
0	Frame flag must be '$' (0x24) Length(not including itself) = 0x08;
1	Server address: 0x00;message from client to server
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =13
5	SET-TIME.request command token: 0x03
6-7	tm_min: LSB first; MSB: 0x00
7-8	tm_hour: LSB first; MSB: 0x00
9-10	tm_mday: LSB first; MSB: 0x00
11-12	tm_mon: LSB first; MSB: 0x00
13-14	tm_year: LSB first
15-16	tm_wday:LSB first; MSB : 0x00
17-18	Terminator : 0x0d 0x0a

The SET-TIME.request primitive is generated by the NVR client and issued to the NVR server to update NVR device's system time.

0x04: SET-TIME.response()
(TBD)
0x05: GET-STREAM-FPS.request()
(TBD)
0x06:GET-STREAM-FPS.response()
(TBD)
0x07: SET-STREAM-FPS.request()

(TBD)
0x08: SET_STREAM_FPS.response()
(TBD)

4.2 Live view control commands 50-99

0x32: UPDATE-VIEW.request(div_mode, channel_mask)

UPDATE-VIEW.request parameters

Name	Type	Valid range	Description
div_mode	Integer	0x00-0x20	Screen division mode: 1: single channel 4: 4 channel display mode 9: 9 channel display mode 16: 16 channel display mode 32: 32 channel display mode
channel_mask	Integer	0x00000000-0xFFFFFFFF	Bitwise bit31-bit0: bit0: channel 1 bit1: channel 2 ... bit31: channe 32

Interface format:

Offset (unsigned char)	Description
0	Frame flag must be '$' (0x24)
1	Server address: 0x00;message from client to server
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =6
5	UPDATE-VIEW.request command token: 0x32
6	Division mode
7	Channel mask : bit7-bit0 (channel 1-8)
8	Channel mask : bit15-bit8 (channel 9-16)
9	Channel mask : bit 23- bit16 (channel 17-24)
10	Channel mask: bit31 -bit24 (channel 25-31)

The UPDATE-VIEW.request primitive is generated by the NVR client and issued to the NVR server to update current NVR view. After the NVR server execute this command, NVR server send selected channels' stream to the client.

0x33: UPDATE-VIEW.response(div_mode, channel_mask)

UPDATE-VIEW.request parameters

Name	Type	Valid range	Description
div_mode	Integer	0x00-0x20	Screen division mode: 1: single channel 4: 4 channel display mode 9: 9 channel display mode 16: 16 channel display mode 32: 32 channel display mode
channel_mask	Integer	0x00000000-0xFFFFFFFF	Bitwise bit31-bit0: bit0: channel 1 bit1: channel 2 ...

Name	Type	Valid range	Description
			bit31: channe 32

Interface format:

Offset (unsigned char)	Description
0	Frame flag must be '$' (0x24)
1	Client address: 0x01-0xFF; message from server to client
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =6
5	UPDATE-VIEW.response command token: 0x33
6	Division mode
7	Channel mask : bit7-bit0 (channel 1-8)
8	Channel mask : bit15-bit8 (channel 9-16)
9	Channel mask : bit 23- bit16 (channel 17-24)
10	Channel mask: bit31 -bit24 (channel 25-31)

The UPDATE-VIEW.response is generated by the NVR device (server) and issued to the NVR client in response to UPDATE-Request primitive. Usually, if UPDATE-RESPONSE request gets executed successfully, the UPDATE-VIEW.response will carry same parameters as an acknowledgment.

0x34: TOGGLE-SCREEN.request(current_screen, screen_action)

TOGGLE-SCREEN.request parameters

Name	Type	Valid range	Description
current_screen	Integer	0x00-0x20	Screen number: for div_mode =1, screen number is just channel number for div_mode =4, screen=1 is corresponding to channel1,2,3,4, screen=2 contains channel 5,6,7,8, etc. For div mode = 9, screen = 1 contains channel 1-9, screen 2 contains chan10-18 for div_mode =16, screen number can be 1 & 2, screen must be 1 if div_mod=32
screen_action	Integer	0x01-0x02	0x02: Next screen switch to next screen. e.g. if div_mode =4, channel_mask =0x155 (channel , 3, 5,7,9), screen 1 contains channel 1, 3, 5,7 screen 2 only contains channel 9, toggling screen commands will switch between screen 1 and screen 2. 0x01: Previous screen go to previous screen

Interface format:

Offset (unsigned char)	Description
0	Frame flag must be '$' (0x24)
1	Server address: 0x00;message from client to server
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =3
5	TOGGLE-SCREEN.request command token: 0x34
6	Screen id
7	Next screen:

The TOGGLE-SCREEN.request is generated by the NVR client and issued to the NVR service to change current live screen view. Note: displayed channels in the next screen is calculated with current screen mode settings(div_mode and activated channels), they have following relationships:

TotalScreens = (TotalCameras + div_mode -1) / div_mode;

Next screen:

NextScreen = (CurrentScreen +1) % (TotalScreens +1);

Previous screen:

PreviousScreen= (CurrentScreen + TotalScreens) % (TotalScreens +1);

where '%' is modulo operation.

Note: Screen number is counted from 1.

0x35: TOGGLE-SCREEN.response(current_screen)

Name	Type	Valid range	Description
current_screen	Integer	0x00-0x20	Current active screen id.

Interface format:

Offset (unsigned char)	Description
0	Frame flag must be '$' (0x24)
1	Client address: 0x01-0xFF;message from server to client
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =2
5	TOGGLE-SCREEN.response command token: 0x35
6	Screen id

The TOGGLE-SCREEN.response is generated by the NVR device (server) and issued to the NVR client in response to TOGGLE-SCREEN.request primitive.

0x36: CAM-SELECT.request(camera_id)

CAM-SELECT.request parameters

Name	Type	Valid range	Description

Name	Type	Valid range	Description
camera_id	Integer	0x01-0x20	Camera(Channel) ID: 1-32

Interface format:

Offset (unsigned char)	Description
0	Frame flag must be '$' (0x24)
1	Server address: 0x00 ; message sent from client to server
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =5
5	CAM-SELECT.request command token: 0x36
6-9	Camera (Channel) id

The CAM-SELECT.request is generated by the NVR client and issued to the NVR server to choose and control current camera .

0x37: CAM-SELECT.response(camera_id)

The CAM-Select.response's parameter is same with CAM-Select request.

The interface format is same with CAM-SELECT.request except that CAM-SELECT.response command token is 0x37.

0x38: GET-ACTIVE-CAM.request()

GET-ACTIVE-CAM.request parameters (none)

Interface format:

Offset (unsigned char)	Description
0	Frame flag must be '$' (0x24)
1	Server address: 0x00;message from client to server
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =1
5	GET-ACTIVE-CAM.request command token: 0x38

The GET-ACTIVE-CAM.request primitive is generated by the NVR client and issued to the NVR server to get current active camera.

0x39: GET-ACTIVE-CAM.response(camera_id)

GET-ACTIVE-CAM.Response parameters

Name	Type	Valid range	Description
camera_id	Integer	0x00-0x20	Current active camera

Interface format:

Offset (unsigned char)	Description
0	Frame flag must be '$' (0x24)
1	Client address: 0x01-0xFF;message from server to

Offset (unsigned char)	Description
	client
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =2
5	GET-ACTIVE-CAM.response command token: 0x39
6-9	camera_id

The GET-ACTIVE-CAM.response is generated by the NVR device (server) and issued to the NVR client in response to GET-ACTIVE-CAM.request primitive.

0x40: VIDEO-ADJUST.request(channel, item_sel, hue, contrast, brightness, saturation)

VIDEO-ADJUST.request parameters

Name	Type	Valid range	Description
channel	Integer	0x00000000-0x00000020	Selected channel (bitwise) bit0 – channel 1 bit1 – channel 2 .. bit31-channel 32 e.g. 0x03, select channel 1&2
item_sel	Integer	0x00000000-0x0000000F	SET_HUE 0x01 SET_CONTRAST 0x02 SET_BRIGHT 0x04 SET_SATURATION 0x08
hue	Integer	0x00-0xff (FIXME on max value)	New hue value
contrast	Integer	0x00-0xff(FIXME on max value)	New contrast value
brightness	Integer	0x00-0xff (FIXME on max value)	New brightness value
saturation	integer	0x00-0xff(FIXME on max value)	New saturation value

Interface format:

Offset (unsigned char)	Description
0	Frame flag must be '$' (0x24)
1	Server address: 0x00;message from client to server
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =13
5	VIDEO-ADJUST.request command token: 0x40
6 -9	Channel number
10-13	Adjust items
14	Hue
15	Contrast

Offset (unsigned char)	Description
16	Brightness
17	Saturation

The VIDEO-Adjust.request is generated by the NVR client and issued to the NVR service to adjust selected channel image (hue, constrast, brightness, saturation).

4.3 Recording control 100-149

0x64:RECORD.request(channel,record_action)

RECORD.request parameters

Name	Type	Valid range	Description
channel	Integer	0x00000000-0xFFFFFFFF	Selected channel (bitwise) bit0 – channel 1 bit1 – channel 2 .. bit31-channel 32 e.g. 0x03, select channel 1&2
record_action	Integer	0x00000000-0xFFFFFFFF	Recording action (bitwise) bit0- channel 1 bit1- channel 2 .. bit31- channel 32 e.g. case 1: channel=0x03, record_action= 0x03, start channel 1&2 recording. Case 2: channel = 0x03 record_action= 0x00, stop channel 1&2 recording.

Interface format:

Offset (unsigned char)	Description
0	Frame flag must be '$' (0x24)
1	Server address: 0x00;message from client to server
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =9
5	RECORD.request command token: 0x64
6	Channel1-8 selection
7	Channel 9-16 selection
9	Channel 17-24 selection
10	Channel 25-32 selection
11	Channel 1-8 recording action
12	Channel 9-16 recording action
13	Channel 17-24 recording action

Offset (unsigned char)	Description
13	Channel 25-32 recording action

The RECORD.request is generated by the NVR client and issued to the NVR service to start/stop channel recording manually.Note: this commands can start/stop multiple channel recording at a time.

0x65: RECORD.Indication(channel)
RECORD.Indication parameters

Name	Type	Valid range	Description
channel	Integer	0x00000000-0xFFFFFFFF	Recorded channel (bitwise : 1: recording; 0: not recording) bit0 – channel 1 bit1 – channel 2 .. bit31-channel 32 e.g. 0x03, select channel 1&2

Interface format:

Offset (unsigned char)	Description
0	Frame flag must be '$' (0x24)
1	Client address: 0x00-0xFF;message from server to client
2	Current message sequence number
3	Protocol type: 0x00 – regular NVR control command
4	Command length =5
5	RECORD.request command token: 0x65
6	Channel1-8 recording state
7	Channel 9-16 recording state
9	Channel 17-24 recording state
10	Channel 25-32 recording state

The Record.indication is generated by the NVR device (server) and issued to the NVR client whenever any channel's recording state gets changed.
Note: any manual/auto/motion sensor inputs which change NVR's recording state will trigger this message. The NVR will broadcast this message to all subscribers.

0x66: VOLUME.request()
(TBD)
0x67: VOLUME.response()
(TBD)

4.4 Play back command 150-199

0x96 PLAY.request()
(TBD)

0x97 PLAY.response()
(TBD)
0x98 PAUSE.request()
(TBD)
0x99 PAUSE.response()
(TBD)

4.5 AutoZoom camera settings 200-255

0xC8: AUTOZOOM.request()
(TBD)

0xC9: AUTOZOOM.response()
(TBD)

0xCA: BACKLIGHT.request()
(TBD)

0xCB: BACKLIGHT.response()
(TBD)

0xCC: ZOOMIN.request()
(TBD)

0xCD: ZOOMIN.response()
(TBD)

0xCE: ZOOMOUT.request()
(TBD)

0xCF: ZOMOUT.response()
(TBD)

5. WEB INTERFACE

NVR's web interface is CGI communication between the client and NVR embedded web server. Prior to using WEB interface, the valid HTTP session with user name , password must be established first. The embedded web server will assign and renew client session id. The session id must be included in all HTTP headers.

5.1 Video clips management

Video clips searching, downloading/deleting are over HTTP POST/GET.

5.1.1 List all video clips

http://xxx.xxx.xxx.xxx/cgi-bin/dskmgr.cgi?xml=1,

```
<videoclips>
        <clip></clip>
        <clip></clip>
        <clip></clip>
</videoclips>
```

5.1.2 Searching video clips

http://xxx.xxx.xxx.xxx/cgi_bin/Search.cgi?startdate=MM/DD/YYYY/HH/MM/SS&enddate=MM/DD/YYYY/HH/MM/SS&cameraid=x&xml=1,

The video clip file result are returned in an XML stream.
Such as

```
<videoclips>
      <clip>
       < start_time></start_time>
       <end_time></end_time>
       <duration></duration>
        <camera_id> </camera_id>
         <file_name></file_name>
         <url></url>
        </clip>
    <clip>
       < start_time></start_time>
       <end_time></end_time>
       <duration></duration>
        <camera_id> </camera_id>
         <file_name></file_name>
         <url></url>
        </clip>
</videoclips>
```

5.1.3 Retrieve (Download) video clips (TBD)

5.1.4 Deleting video clips (TBD)

http://xxx.xxx.xxx.xxx/cgi_bin/dskmgr.cgi?delsel=FileName&delse=FileName2&delsel=FileName3

5.1.5 Change video clip attribute (TBD)

http://xxx.xxx.xxx.xxx/cgi_bin/dskmgr.cgi?filename=FileName&chmod=xxx