RCP+ Commands for Advanced Integration Package

Bosch Video IP

BOSCH

en Specification

Table of Contents

1 Introduction

This document shall give you an overview of the RCP+ (Remote Control Protocol) commands used within the Starter Integration Package. Each command and available options are described in detail. Furthermore this document explains the identification of the opcode and the usage of the CGI (Common Gateway Interface) for transmitting RCP+ commands.

1.1 RCP+ SDK

RCP+ SDK is the appropriate tool to program your Bosch IP video equipment in detail using the commands of the Remote Control Protocol. It requires the Windows based Bosch Software Development Kit (SDK).

1.2 CGI

By means of CGI you may send RCP+ commands via your standard web browser or a web server based application.

1.3 Access Levels

In RCP+ SDK commands there are four different kinds of access levels.

Access level	Permission
No protection	On this access level there is full access for everyone to the relevant command.
Live	This access level grants you to read information, e. g. read out the camera name.
User	This access level gives you the permission to use write commands, e. g. define the position of the name stamping.
Service	This access level is reserved for administrator use and allows you to change the device settings in detail.

1.4 Payload in general

1.4.1 Read request

The returned payload represents the values that are currently set on the device.

1.4.2 Write request

The payload represents the values that are set on the device after the command was successfully executed.

1.5 RCP errors

The packet will have the standard layout with the method field set to “Error”. The first byte of the payload section contains error cause.

If the error code is RCP_ERROR_COMMAND_SPECIFIC, then the command specific error (see RCP command for details) is included in the second byte.

The following generic error codes are defined:

Error code	
RCP_ERROR_UNKNOWN	0xFF
RCP_ERROR_INVALID_VERSION	0x10
RCP_ERROR_NOT_REGISTERED	0x20
RCP_ERROR_INVALID_CLIENT_ID	0x21
RCP_ERROR_INVALID_METHOD	0x30
RCP_ERROR_INVALID_CMD	0x40
RCP_ERROR_INVALID_ACCESS_TYPE	0x50
RCP_ERROR_INVALID_DATA_TYPE	0x60
RCP_ERROR_WRITE_ERROR	0x70
RCP_ERROR_PACKET_SIZE	0x80
RCP_ERROR_READ_NOT_SUPPORTED	0x90
RCP_ERROR_INVALID_AUTH_LEVEL	0xa0
RCP_ERROR_INVAILD_SESSION_ID	0xb0
RCP_ERROR_TRY_LATER	0xc0
RCP_ERROR_TIMEOUT	0xd0
RCP_ERROR_NO_LICENCE (used by NVR only)	0xe0
RCP_ERROR_COMMAND_SPECIFIC	0xf0
RCP_ERROR_ADDRESS_FORMAT	0xf1

Note!

The error code 0xc0 RCP_ERROR_TRY_LATER indicates that the VideoJet recognizes the command, but it cannot be processed immediately. The client should repeat this command later.

1.6 Basic principle

The examples at the end of each command just cover sending the RCP+ commands over CGI.

1.6.1 Corresponding CGI-Code of the write command

```
http://160.10.0.100/rcp.xml?command=0x0607
&type=T_DWORD&direction=WRITE&payload=10000&num=1
```

consisting of

http://	=	introduction of Hypertext Transfer Protocol (HTTP)
160.10.0.100	=	exemplary IP address
rcp.xml	=	CGI script
command=0x0607	=	tag code of the RCP+ command
type=T_DWORD	=	datatype of the RCP+ command
direction=WRITE	=	direction of the RCP+ command
payload=10000	=	payload value 10000 (target bit rate in kbps)
num=1	=	numeric descriptor 1 (profile preset)

Note:

The values for type and direction are case-sensitive.

2 RCP+ commands

Following the RCP+ commands are described in alphabetical order.

2.1 CONF_ALARM_DISP_VAL

Requires firmware version 3.00 or higher.

2.1.1 General description

This command enables/disables video overlays for the alarm stamping. The selectable options are:

- *alarm display off,*
- *alarm display on,*
- *alarm stamping with custom attributes*.

With the option *alarm stamping with custom attributes* it is possible to display the stamping at a custom x and y position.

For setting the values for x and y see command **CONF_STAMP_ATTR_ALARM** (see page 139).

Tag code: 0x008e

Numeric descriptor: Not used

Direction	**Read**		**Write**	
Access level	No protection		Service	
Data type	T_OCTET	0x01	T_OCTET	0x01

2.1.2 Payload options

1 = alarm display off

2 = alarm display on

3 = alarm stamping with custom attributes

2.1.3 RCP+ over CGI example

Read request

In the example below, the alarm display position value is checked. In this case, alarm display is OFF.

```
http://<Device IP>/rcp.xml?command=0x008e
&type=T_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x008e</hex>
      <dec>142</dec>
   </command>
   <type>T_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x218b</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x01</hex>
      <dec>1</dec>
   </result>
</rcp>
```

Write request

In the example below, the alarm stamping is set to ON.

```
http://<Device IP>/rcp.xml?command=0x008e
&type=T_OCTET&direction=WRITE&payload=2
```

2.1.4 RCP+ SDK example

Read request

```
In the example below, the alarm display position value is checked.
```

```
RcpInputStream response;
client.sendRequest(0, 0x008e0130, response);
```

Write request

In the example below, the alarm display is set to the mode ON.

```
RcpOutputStream payload;
payload.writeOctet(2);
RcpInputStream response;
client.sendRequest(0, 0x008e0131, payload, response);
```

2.2 CONF_AUDIO_INPUT

Requires firmware version 2.52 or higher.

2.2.1 General description

This command enables you to get and set the audio input mode.

Tag code: 0x09b8

Numeric descriptor: Audio line

Direction	Read		Write	
Access level	No protection		Service	
Data type	T_DWORD	0x08	T_DWORD	0x08

2.2.2 Payload options

0 = line

1 = microphone

2 = mute (only supported on CCP3/4 devices)

2.2.3 RCP+ over CGI example

Read request

The example below returns the audio mode on audio line 1, in this case it is set to line.

```
http://<Device IP>/rcp.xml?command=0x09b8
&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x09b8</hex>
      <dec>2488</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000000</hex>
      <dec>0</dec>
   </result>
</rcp>
```

Write request

In the example below, the audio input on audio line 1 is set to microphone.

```
http://<Device IP>/rcp.xml?command=0x09b8
&type=T_DWORD&num=1&direction=WRITE&payload=1
```

2.2.4 RCP+ SDK example

Read request

The example below returns the audio mode on audio line 1.

```
client.sendRequest(0, 0x09b80830, response);
response.wait();
audioInputMode = response.readULong();
```

Write request

In the example below, the audio input on audio line 1 is set.

```
RcpOutputStream payload;
payload.writeULong(audioInputMode);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x09b80831, payload, response);
response.wait();
```

2.3 CONF_AUDIO_INPUT_LEVEL

Requires firmware version 2.52 or higher.

2.3.1 General description

This command enables you to get and set the audio input level.

Tag code: 0x000a

Numeric descriptor: Audio line

Direction	Read		Write	
Access level	No protection		Service	
Data type	T_DWORD	0x08	T_DWORD	0x08

2.3.2 Payload options

Range: 0 to AUDIO_INPUT_MAX (see **CONF_AUDIO_INPUT_MAX**).

2.3.3 RCP+ over CGI example

Read request

The example below returns the current audio input level on line 1, in this case it is 0.

```
http://<Device IP>/rcp.xml?command=0x000a
&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x000a</hex>
      <dec>10</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x1b56</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000000</hex>
      <dec>0</dec>
   </result>
</rcp>
```

Write request

In the example below, the input level on line 1 is set to 5.

```
http://<Device IP>/rcp.xml?command=0x000a
&type=T_DWORD&num=1&direction=WRITE&payload=0x5
```

2.3.4 RCP+ SDK example

Read request

In the example below returns the current audio input level on line 1.

```
client.sendRequest(0, 0x000a0830, response);
response.wait();
inputLevel = response.readULong();
```

Write request

In the example below the input level on line 1 is set.

```
RcpOutputStream payload;
payload.writeULong(inputLevel);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x000a0831, payload, response);
response.wait();
```

2.4 CONF_AUDIO_INPUT_MAX

Requires firmware version 2.52 or higher.

2.4.1 General description

This command enables you to get the maximum input level.

Tag code: 0x09ba

Numeric descriptor: Audio line

Direction	Read		Write
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.4.2 Payload options

Maximal input level value is returned.

2.4.3 RCP+ over CGI example

Read request

The example below returns the maximum audio input level on the first line, in this case 31.

```
http://<Device IP>/rcp.xml?command=0x09ba
&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x09ba</hex>
      <dec>2490</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x0398</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x0000001f</hex>
      <dec>31</dec>
   </result>
</rcp>
```

2.4.4 RCP+ SDK example

Read request

The example below returns the maximum audio input level on the first line.

```
client.setNum(1);
client.sendRequest(0, 0x09ba0830, response);
response.wait();
inputLevelMax = response.readULong();
```

2.5 CONF_AUDIO_MIC_LEVEL

Requires firmware version 2.52 or higher.

2.5.1 General description

This command enables you to get and set the input level of the microphone.

Tag code: 0x09bc

Numeric descriptor: Audio line

Direction	Read		Write	
Access level	No protection		Service	
Data type	T_DWORD	0x08	T_DWORD	0x08

2.5.2 Payload options

Range: 0 to AUDIO_MIC_MAX (see **CONF_AUDIO_MIC_MAX**).

2.5.3 RCP+ over CGI example

Read request

The example below returns the microphone level on audio line 1. In this case it is 0.

```
http://<Device IP>/rcp.xml?command=0x09bc
&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x09bc</hex>
      <dec>2488</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x0710</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000000</hex>
      <dec>0</dec>
   </result>
</rcp>
```

Write request

In the example below, the microphone level on audio line 1 is set to 1.

```
http://<Device IP>/rcp.xml?command=0x09bc
&type=T_DWORD&direction=WRITE&num=1&payload=0x1
```

2.5.4 RCP+ SDK example

Read request

The example below returns the microphone level on audio line 1.

```
client.sendRequest(0, 0x09bc0830, response);
response.wait();
micLevel = response.readULong();
```

Write request

In the example below, the microphone level on audio line 1 is set.

```
RcpOutputStream payload;
payload.writeULong(micLevel);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x09bc0831, payload, response);
response.wait();
```

2.6 CONF_AUDIO_MIC_MAX

Requires firmware version 2.52 or higher.

2.6.1 General description

This command enables you to get the maximum microphone level.

Tag code: 0x09bd

Numeric descriptor: Audio line

Direction	Read		Write
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.6.2 Payload options

Maximal microphone value is returned.

2.6.3 RCP+ over CGI example

Read request

The example below returns the maximum microphone level, in this case 70.

```
http://<Device IP>/rcp.xml?command=0x09bd
&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x09bd</hex>
      <dec>2492</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x1b3d</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000046</hex>
      <dec>70</dec>
   </result>
</rcp>
```

2.6.4 RCP+ SDK example

Read request

The example below returns the maximum microphone level.

```
client.sendRequest(0, 0x09bd0830, response);
response.wait();
micLevelMax = response.readULong();
```

2.7 CONF_AUDIO_ON_OFF

Requires firmware version 2.52 or higher.

2.7.1 General description

This command enables you to get and set the audio status.

Tag code: 0x000c

Numeric descriptor: Not used

Direction	Read		Write	
Access level	No protection		Service	
Data type	F_FLAG	0x00	F_FLAG	0x00

2.7.2 Payload options

0 = audio mode off

1 = audio mode on

Note:

Set audio mode (0 = off, 1 = on) has no effect on a running recording. To have the effect in recording, restart of recording is necessary.

2.7.3 RCP+ over CGI example

Read request

The example below returns the audio mode, in this case it is switched on, which means audio is transmitted.

```
http://<Device IP>/rcp.xml?command=0x000c
&type=F_FLAG&direction=READ
```

```
<rcp>
   <command>
      <hex>0x000c</hex>
      <dec>12</dec>
   </command>
   <type>F_FLAG</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x0037</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x01</hex>
```

```
      <dec>1</dec>
   </result>
</rcp>
```

Write request

In the example below, the audio transmission is disabled.

```
http://<Device IP>/rcp.xml?command=0x000c
&type=F_FLAG&direction=WRITE&payload=0x0
```

2.7.4 RCP+ SDK example

Read request

The example below returns the audio mode.

```
client.sendRequest(0, 0x000c0030, response);
response.wait();
audioMode = response.readOctet();
```

Write request

In the example below, the audio transmission is disabled.

```
RcpOutputStream payload;
payload.writeOctet(0x0);
RcpInputStream response;
client.sendRequest(0, 0x000c0031, payload, response);
response.wait();
```

2.8 CONF_AUDIO_OPTIONS

Requires firmware version 2.52 or higher.

2.8.1 General description

This command enables you to get the audio options.

Tag code: 0x09bf

Numeric descriptor: Audio line

Direction	Read		Write	
Access level	No protection		Service	
Data type	T_DWORD	0x08	T_DWORD	0x08

2.8.2 Payload options

Bit 1 = line in

Bit 2 = line out

Bit 3 = microphone

Bit 4 = loudspeaker

2.8.3 RCP+ over CGI example

Read request

In the example below, the audio options are received. In this case, audio line 1 can be used as line in (bit 1 is set) and line out (bit 2 is set).

```
http://<Device IP>/rcp.xml?command=0x09bf
&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x09bf</hex>
      <dec>2495</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000003</hex>
      <dec>3</dec>
   </result>
</rcp>
```

2.8.4 RCP+ SDK example

Read request

In the example below, the audio options are received.

```
ulong audioOptions;
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x09bf0830, response);
response.wait();
audioOptions = response.readULong();
```

2.9 CONF_AUDIO_OUTPUT_LEVEL

Requires firmware version 2.52 or higher.

2.9.1 General description

This command enables you to get and set the audio output level.

Tag code: 0x09b7

Numeric descriptor: Audio line

Direction	Read		Write	
Access level	No protection		Service	
Data type	T_DWORD	0x08	T_DWORD	0x08

2.9.2 Payload options

Range: 0 to AUDIO_OUTPUT_MAX (see **CONF_AUDIO_OUTPUT_MAX**).

2.9.3 RCP+ over CGI example

Read request

The example below returns the current audio output level, in this case 0.

```
http://<Device IP>/rcp.xml?command=0x09b7
&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x09b7</hex>
      <dec>2487</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x1b5a</</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000000</hex>
      <dec>0</dec>
   </result>
</rcp>
```

Write request

In the example below the output level is set to 5.

```
http://<Device IP>/rcp.xml?command=0x09b7
&type=T_DWORD&num=1&direction=WRITE&payload=0x5
```

2.9.4 RCP+ SDK example

Read request

The example below returns the current audio output level on audio line 1.

```
client.sendRequest(0, 0x09b70830, response);
response.wait();
outputLevel = response.readULong();
```

Write request

In the example below, the audio output level on audio line 1 is set.

```
RcpOutputStream payload;
payload.writeULong(outputLevel);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x09b70831, payload, response);
response.wait();
```

2.10 CONF_AUDIO_OUTPUT_MAX

Requires firmware version 2.52 or higher.

2.10.1 General description

This command enables you to get the maximum output level.

Tag code: 0x09bb

Numeric descriptor: Audio line

Direction	Read		Write
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.10.2 Payload options

Maximal output level value is returned.

2.10.3 RCP+ over CGI example

Read request

In the example below, the maximum output level on line 1 is retrieved, in this case 31.

```
http://<IP>/rcp.xml?command=0x09bb&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x09bb</hex>
      <dec>2491</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x03aa</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x0000001f</hex>
      <dec>31</dec>
   </result>
</rcp>
```

2.10.4 RCP+ SDK example

Read request

In the example below, the maximum output level on line 1 is retrieved.

```
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x09bb0830, response);
response.wait();
outputLevelMax = response.readULong();
```

2.11 CONF_AUDIO_REC_FORMAT

Requires firmware version 4.50 or higher.

2.11.1 General description

This command enables you to get and set the audio recording format.

Tag code: 0x0ae9

Numeric descriptor: Not used

Direction	Read		Write	
Access level	No protection		Service	
Data type	T_OCTET	0x01	T_OCTET	0x01

2.11.2 Payload options

0 = NO

1 = G711

2 = L16

3 = AAC

2.11.3 RCP+ over CGI example

Read request

In the example below, the audio recording format is retrieved. In this case it is set to no recording.

```
http://<Device IP>/rcp.xml?command=0x0ae9
&type=T_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0ae9</hex>
      <dec>2793</dec>
   </command>
   <type>T_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00</hex>
      <dec>0</dec>
   </result>
</rcp>
```

Write request

In the example below, the audio recording format is set to AAC.

```
http://<Device IP>/rcp.xml?command=0x0ae9
&type=T_OCTET&direction=WRITE&payload=3
```

2.11.4 RCP+ SDK example

Read request

In the example below, the audio recording format is retrieved. In this case it is set to no recording.

```
RcpInputStream response;
client.sendRequest(0, 0x0ae90130, response);
response.wait();
audioRecordingMode = response.readOctet();
```

Write request

In the example below, the audio recording format is set.

```
RcpOutputStream payload;
payload.writeOctet(audioRecordingMode);
RcpInputStream response;
client.sendRequest(0, 0x0ae90131, payload, response);
response.wait();
```

2.12 CONF_BICOM_COMMAND - BICOM message over RCP command

Requires firmware version 3.00 or higher.

Note:
This command is also sent as a message every second.

2.12.1 General description

This command redirects the payload to the BICOM interface of the analog camera part. The BICOM protocol can be used only for pure IP devices. Analog cameras connected to PTZ units require the OSRD protocol. Refer to the separate BICOM documentation for details and command syntax.

For some BICOM commands an access level higher than **User** is required.

Tag code: 0x09a5

Numeric descriptor: Not used

Direction	**Read request**		**Write request**	
Access level	No protection		User	
Data type	P_OCTET	0x01	P_OCTET	0x01

2.12.2 Payload options

Payload structure

Without Lease Time:

8	24	40	48
Flags 1 Byte	**BICOM Server ID** 2 Bytes	**Object ID** 2 Bytes	**Operation** 1 Byte
BICOM Payload n Bytes			

With Lease Time:

8	24	56	
Flags 1 Byte	**Lease time** 2 Bytes	**Lease time id** 4 Bytes	
BICOM Server ID 2 Bytes	**Object ID** 2 Bytes	**Operation** 1 Byte	**BICOM Payload** n Bytes

Flags

Transmission flags:

Values	
Return_Payload (must be set to 1 if return payload is expected)	Bit 0
Best_Effort (set to 1 to transmit as best effort frame)	Bit 1
Native_Errors (set to 1 to receive the native BICOM errors)	Bit 2
Lease_Time_Available (set to 1 if a lease time is included in the request)	Bit 3
Unused set to 0	Bit 4
Unused set to 0	Bit 5
Unused set to 0	Bit 6
Flags_Available (must be always set to 1)	Bit 7

Lease time

Time period in seconds the access should be blocked for other clients.

Lease time id

Random number generated by the client. If a lease time > 0 is provided with the first access, further accesses during the lease time are only possible if the same lease time id is provided.

BICOM server ID

Server ID, e.g. 0x0002 for “Device Server” (see BICOM documentation).

Object ID

Object ID, e.g. 0x0100 for “Type” (see BICOM documentation).

Operation

See BICOM documentation.

Values	
GET	0x01
SET	0x02
SET_GET	0x03
INC	0x04
INC_GET	0x05
DEC	0x06
DEC_GET	0x07
SET_DFLT	0x08
SET_GET_DFLT	0x09

2.12.3 RCP+ over CGI examples

Get payload

The example below returns the zoom position.

```
http://<Device IP>/rcp.xml?command=0x09a5
&type=P_OCTET&direction=WRITE&payload=0x810006011401
```

RCP request payload description

81	Return payload expected
00 06	Server ID = 6 ("PTZ server")
01 14	Object "IdString"RCP+
01	Operation GET

RCP reply

81	Return payload expected
00 06	Server ID = 6 ("PTZ server")
01 14	Object "IdString"
01	Operation GET
00 68 00 68 00 68 00 68 00 20	
00 20 00 20 00 20 00 20 00 20	
00 20 00 20 00 20 00 20 00 20	
00 20 00 20	34 Bytes Unicode ID String

Set payload

In the example below, the PTZ device turns left where x is the speed (15 speeds are possible from 0x1 to 0xF).

```
http://<Device IP>/rcp.xml?command=0x09a5
&type=P_OCTET&direction=WRITE&payload=0x8000060110850X0000
```

RCP request payload description

80	Return payload
00 06	Server ID = 6 ("PTZ server")
01 10	Object "IdString"
85	MoveContVarSpeed
0F	pan (1 = right, 0 = left)
00	tilt (1 = up, 0 = down)
00	zoom (1 = in, 0 = out)

2.12.4 RCP+ SDK examples

Get payload

The example below returns the zoom position.

```
payload.writeOctet(0x81);          //flags and return payload expected
payload.writeUShort(0x0006);       //BICOM Server ID: PTZ Server
payload.writeUShort(0x0114);       //Object ID: Zoom Position
payload.writeOctet(0x01);          //Get request
client.sendRequest(0, 0x09A50C30, payload, response);
response.wait();                   //the last two bytes is the hex position
response.skip(6);
zoomPos = response.readUShort();
```

Set payload

In the example below, the PTZ device turns left where x is the speed (15 speeds are possible from 0x1 to 0xF).

```
RcpClient client("<Device IP>", 0, 1);
byte[] payloadBytes = new byte[12];
int speed= 15;

payloadBytes [0] = 0x80;           //Flags available
payloadBytes [1] = 0x00;           //BICOM Server ID: PTZ Server
payloadBytes [2] = 0x06;           //BICOM Server ID: PTZ Server
payloadBytes [3] = 0x01;           //Object ID: Position
payloadBytes [4] = 0x10;           //Object ID: Position
payloadBytes [5] = 0x85;           //Move Cont. with variable Speed
payloadBytes [6] = speed;          //Pan with speed
payloadBytes [7] = 0x00;           //Tilt with speed
payloadBytes [8] = 0x00;           //Zoom with speed
payload.write(payloadBytes, 9);
client.sendRequest(0, 0x09A50C31, payload, response);
```

Note:

Please see BICOM documentation for the command syntax.

2.13 CONF_BICOM_COMMAND - BICOM event over RCP message

Requires firmware version 3.00 or higher.

Note:
This command is also sent as a message every second.

2.13.1 General description

This command redirects the payload to the BICOM interface of the analog camera part. The BICOM protocol can be used only for pure IP devices. Analog cameras connected to PTZ units require the OSRD protocol. Refer to the separate BICOM documentation for details and command syntax.

For some BICOM commands an access level higher than **User** is required.

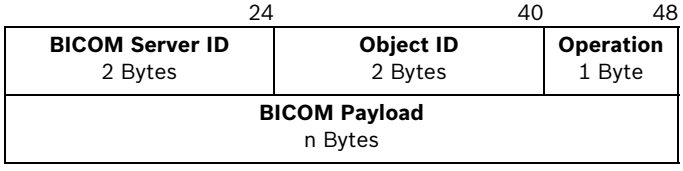
Tag code: 0x09a5

Numeric descriptor: Not used

Direction	**Read request**		**Write request**	
Access level	No protection		User	
Data type	P_OCTET	0x01	P_OCTET	0x01

2.13.2 Payload options

Payload structure



BICOM server ID

Server ID, e.g. 0x0004 for “Camera Server” (see BICOM documentation).

Object ID

Object ID, e.g. 0x0190 for “Color” (see BICOM documentation).

Operation

See BICOM documentation.

Values	
EVENT	0x70 ... 0x7F

2.13.3 RCP+ over CGI example

Receive message

Register to a message.

```
http://<Device IP>/rcp.xml?command=0x09a5
&type=P_OCTET&direction=WRITE&payload=0x00040190700000
```

RCP message

00 04	Server ID = 4 (“Camera Server”)
01 90	Object “Color”
70	Operation EVENT
00 00	2 Byte unsigned short: Color mode is B/W

2.13.4 RCP+ SDK example

Receive message

Register to a message.

```
RcpClient client("<Device IP>", 0, 1);
payload.writeUShort(0x0004);      //BICOM Server ID: Camera Server
payload.writeUShort(0x0190);      //Object ID: Colour
payload.writeOctet(0x70);         //Operation EVENT
payload.writeUShort(0x0000);      // Colour mode B/W
client.sendRequest(0, 0x09A50C30, payload, response);
```

Note:

Please see BICOM documentation for the command syntax.

2.14 CONF_CAMNAME

Requires firmware version 2.53 or higher.

2.14.1 General description

This command enable you to get and set the camera name of the device.

Tag code: 0x0019

Numeric descriptor: Video line

Direction	Read		Write	
Access level	No protection		Service	
Data type	P_UNICODE	0x14	P_UNICODE	0x14

2.14.2 Payload options

It contains the camera name in Unicode as string; the maximum length is 32 characters.

2.14.3 RCP+ over CGI example

Read request

In the example below, the camera name on the first video line is received. In this case it is Office Camera.

```
http://<Device IP>/rcp.xml?command=0x0019
&type=P_UNICODE&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0019</hex>
      <dec>25</dec>
   </command>
   <type>P_UNICODE</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x0282</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>28</len>
      <str> 00 4f 00 66 00 66 00 69  00 63 00 65 00 20 00 43  00 61 00 6d 00
            65 00 72  00 61 00 00
      </str>
   </result>
</rcp>
```

Write request

In the example below, the camera name on the first video line is changed to “Tower Camera”.

```
http://<Device IP>/rcp.xml?command=0x0019
&type=P_UNICODE&num=1&direction=WRITE
&payload=0x0054006f00770065007200200043006100
6d0065007200610000
```

2.14.4 RCP+ SDK example

Read request

In the example below, the camera name on the first video line is received.

```
RcpInputStream response;
Wchar_t CameraName[32];
client.setNum(1);
client.sendRequest(0, 0x00191430, response);
response.wait();
response.readWString(CameraName, response.available());
```

Write request

In the example below, the camera name on the first video line is changed.

```
payload.writeWString(CameraName);
RcpInputStream response;
client.setNum(1);
client.sendRequest(0, 0x00191431, payload, response);
response.wait();
```

2.15 CONF_CAPABILITY_LIST

Requires firmware version 2.52 or higher.

2.15.1 General description

This command provides you the capabilities of the device, these contains the Video, Audio, Serial, IO and camera Data capabilities.

Tag code: 0xff10

Numeric descriptor: Not used

Direction	Read		Write
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.15.2 Payload options

Payload structure

0xBABA 2 Bytes	Version 2 Bytes	NbSection 2 Bytes	Section 1	...	Section N

Version

Version of the capabilities (currently 0x0001).

NbSection

Number of following sections.

Section structure

Type 2 Bytes	Size 2 Bytes	NbElement 2 Bytes	Element 1	...	Element N

Type

Values	
Video	0x0001
Audio	0x0002
Serial	0x0003
IO	0x0004
Camera Data	0x0005

Size

Size of the section including SectionType, Size and NbElement, in case the section is unknown, you can skip to the next using the size.

NbElement

Number of the following elements.

Element structure

For section type Video.

Type 2 Bytes	Identifier 2 Bytes	Compression 2 Bytes	InputNo 2 Bytes	Resolution 2 Bytes

Type

Values	
VIDEO_ENCODER	0x0001
VIDEO_DECODER	0x0002

Identifier

Unique video stream identifier which is used to address this entity.

Compression

One or multiple of the following.

Values	
VIDEO_COMP_MPEG2	0x0001
VIDEO_COMP_MPEG4	0x0002
VIDEO_COMP_H264	0x0004
VIDEO_COMP_JPEG	0x0008

InputNo

InputNo is the number of the physical input from which the entity gets or puts its video.

Resolution

One or multiple of the following.

Values	
VIDEO_RESO_QCIF	0x0001
VIDEO_RESO_CIF	0x0002
VIDEO_RESO_2CIF	0x0004
VIDEO_RESO_4CIF	0x0008
VIDEO_RESO_CUSTOM	0x0010
VIDEO_RESO_QVGA	0x0020
VIDEO_RESO_VGA	0x0040
VIDEO_RESO_HD720	0x0080
VIDEO_RESO_HD1080	0x0100
VIDEO_RESO_WD144	0x0200
VIDEO_RESO_WD288	0x0400

Values	
VIDEO_RESO_WD432	0x0800
VIDEO_RESO_HD2592x1944	0x1000

Element structure

For section type Audio.

Type 2 Bytes	Identifier 2 Bytes	Compression 2 Bytes

Type

Values	
AUDIO_ENCODER	0x0001
AUDIO_DECODER	0x0002

Identifier

Unique audio stream identifier which is used to address this entity.

Compression

One or multiple of the following.

Values	
AUDIO_COMP_MPEG2	0x0001
AUDIO_COMP_G711	0x0002
AUDIO_COMP_AAC	0x0004

Element structure

For section type Serial.

Type 2 Bytes	Identifier 2 Bytes

Type

One or multiple of the following.

Values	
SERIAL_RS232	0x0001
SERIAL_RS485	0x0002
SERIAL_RS422	0x0004

Identifier

Unique serial port identifier which is used to address this entity.

Element structure

For section type IO.

Type 2 Bytes	Identifier 2 Bytes

Type

Values	
IO_INPUT	0x0001
IO_OUTPUT	0x0002
IO_VIRTUAL_IN	0x0003

Identifier

Unique IO identifier which is used to address this entity.

Element structure

For section type Camera Data.

Type 2 Bytes	InputNo 2 Bytes

Type

Values	
CAMDATA_BILINX	0x0001

InputNo

InputNo is the number of the physical video input.

2.15.3 RCP+ over CGI example

Read request

In the example below, the capabilities of an AutoDome JR are retrieved.

```
http://<Device IP>/rcp.xml?command=0xff10&type=P_OCTET&direction=READ
```

```
<str>
ba ba ---> BABA
00 01 ---> Version
00 04 ---> NbSection =4

--------- Section Structure 1
00 01 ---> Section 1 = Video
00 2e ---> Size = 14Byte
00 04 ---> Number of Elements = 4

--------- Element Structure 1
00 01 ---> Type = Video Encoder
00 02 ---> Identifier = absolute coder number =2 = Stream1
```

```
00 04 ---> Compression = VIDEO_COMP_H264
00 01 ---> InputNo = 1
01 8a ---> Resolution = bitweise verknüpft

--------- Element Structure 2
00 01 ---> ype = Video Encoder
00 03 ---> Identifier = absolute coder number = 3 = Stream2
00 04 ---> Compression = VIDEO_COMP_H264
00 01 ---> InputNo = 1
01 8a ---> Resolution = bitweise verknüpft

--------- Element Structure 3
00 01 ---> Type = Video Encoder
00 04 ---> Identifier = absolute coder number =4 = Stream 3
00 08 ---> Compression = VIDEO_COMP_JPEG
00 01 ---> InputNo = 1
01 8a ---> Resolution = bitweise verknüpft

--------- Element Structure 4
00 01 ---> Type = Video Encoder
00 05 ---> Identifier = absolute coder number =5 = Stream 4, i-Frame
00 04 ---> Compression = VIDEO_COMP_H264
00 01 ---> InputNo = 1
01 8a ---> Resolution = bitweise verknüpft

--------- Section Structure 2
00 02 ---> Section 2 = Audio
00 12 ---> Size
00 02 ---> Nb Elements = 2

--------- Element Structure 1
00 01 ---> Type = Audio Encoder (Audio In)
00 01 ---> Identifier = absolute coder number =1
00 02 ---> Compression = AUDIO_COMP_G711

--------- Element Structure 2
00 01 ---> Type = Audio Encoder
00 03 ---> Identifier = absolute coder number   =3
00 04 ---> Compression = AUDIO_COMP_AAC

--------- Section Structure 3
00 03 ---> Section 3 = Serial
00 0a ---> Size
00 01 ---> Number of Elements = 1

--------- Element Structure 1
00 07 ---> Type = Bittweise verknüpft = RS232/485/422
00 01 ---> Identifier =1

--------- Section Structure 4
```

```
00 04 ---> Section 4 = IO
00 22 ---> Size
00 07 ---> Number of Elements = 7

--------- Element Structure 1
00 01 ---> Type = IO_INPUT
00 01 ---> Identifier=1

--------- Element Structure 2
00 01 ---> Type = IO_INPUT
00 02 ---> Identifier=2

--------- Element Structure 3
00 02 ---> Type = IO_OUTPUT
00 01 ---> Identifier=1

--------- Element Structure 4
00 03 ---> Type = IO_VIRTUAL_IN
00 01 ---> Identifier=1

--------- Element Structure 5
00 03 ---> Type = IO_VIRTUAL_IN
00 02 ---> Identifier=2

--------- Element Structure 6
00 03 ---> Type = IO_VIRTUAL_IN
00 03 ---> Identifier=3

--------- Element Structure 7
00 03 ---> Type = IO_VIRTUAL_IN
00 04 ---> Identifier=4

</str>
   </result>
</rcp>
```

2.15.4 RCP+ SDK example

Read request

In the example below, the capabilities are retrieved.

```
client.sendRequest(0, 0xff100c30, response);
response.wait();

response.readUShort(); //BABA

   response.skip(2);//Version
   unsigned short nbrOfSections = response.readUShort();
   for (int i=0; i<nbrOfSections; i++)
   {
```

```
      unsigned short sectionType = response.readUShort();
      unsigned short sectionSize = response.readUShort();
      unsigned short sectionElements = response.readUShort();
      switch (sectionType)
      {
      case 0x0001://video
         printf("Video Section\n");
         for (int j=0; j<sectionElements; j++)
         {
            unsigned short type = response.readUShort();
            unsigned short identifier = response.readUShort();
            unsigned short compression = response.readUShort();
            unsigned short inputNbr = response.readUShort();
            unsigned short resolution = response.readUShort();
            printf("Type %s\tIdentifier %02d\tCompression %02x\tVideoInput
%02d\tResolution %02x\n", Type[type], identifier, compression, inputNbr,
resolution);

         }
         break;
      case 0x0002://audio
         printf("Audio Section\n");
         for (int j=0; j<sectionElements; j++)
         {
            unsigned short type = response.readUShort();
            unsigned short identifier = response.readUShort();
            unsigned short compression = response.readUShort();
            printf("Type %02x\tIdentifier %02x\tCompression %02x\n", type,
identifier, compression);
         }
         break;
      case 0x0003://serial
         printf("Serial Section\n");
         for (int j=0; j<sectionElements; j++)
         {
            unsigned short type = response.readUShort();
            unsigned short identifier = response.readUShort();
            printf("Type %02x\tIdentifier %02x\n", type, identifier);
         }

         break;
      case 0x0004://IO
         printf("I/O Section\n");
         for (int j=0; j<sectionElements; j++)
         {
            unsigned short type = response.readUShort();
            unsigned short identifier = response.readUShort();
            printf("Type %02x\tIdentifier %02x\n", type, identifier);
         }
         break;
```

```
   }//switch
}
unsigned short sectionType = response.readUShort();
```

2.16 CONF_DEVICE_CAPABILITIES

Requires firmware version 5.70 or higher.

2.16.1 General description

This command provides you the capabilities of your device.

Tag code: 0x0b60

Numeric descriptor: Not used

Direction	Read		Write
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.16.2 Payload options

Payload structure

Numer of entries 4 Bytes	
Tag 1 2 Bytes	**Length 1** 2 Bytes
Payload 1 Length 1 Byte	
...	
Tag n 2 Bytes	**Length n** 2 Bytes
Payload n Length n Bytes	

Number of entries:

The total number of capability entries.

Tags

Tag number	Type	Options
0x1	DEVICE_TYPE	1 = encoder 2 = camera 3 = transcoder 4 = vrm 5 = decoder 0 = other
0x2	PTZ_CAMERA	0 = not supported 1 = supported
0x3	ROI_CAMERA	0 = not supported 1 = supported

Tag number	Type	Options
0x4	AUTOTRACKER	0 = not supported 1 = supported
0x5	NBR_VIDEO_IN	number of video inputs
0x6	NBR_TRANSCODER	number of transcoders
0x7	NBR_AUDIO_IN	number of audio inputs
0x8	NBR_AUDIO_OUT	number of audio outputs
0x9	AUDIO_OPTIONS	bit0 line in, bit1 line out, bit2 mic, bit3 loudspeaker (see **CONF_AUDIO_OPTIONS**)
0xA	E-PTZ	0 = not supported 1 = supported
0xB	IMAGE_PIPE_FEATURES	returns the information that is provided by BICOM command 0x0502 Bit field, listing camera features: byte0::bit0: WDR (LSB) byte0::bit1: User modes byte0::bit2: Zoom lens byte0::bit3: IR corrected lens is present (single focus slider shall be shown) byte0::bit4: BLC byte0::bit5: Intelligent AE (needs BLC enabled) byte0::bit6: Contrast enhancement byte0::bit7: ALC slider (MSB) byte1::bit0: Fan byte1::bit1: Default shutter byte1::bit2: Lens stop configuration in property page (see 0x01C.5) byte1::bit3: Lens stop feature present (see 0x01C.5) byte1::bit4: Sharpness slider byte1::bit5: Intelligent DNR byte1::bit6: Day/night switch byte1::bit7: Son/sox ATW mode byte2::bit0: Motorized back focus byte2::bit1: Saturation control byte2::bit2: IR illuminator present This information can be used to make property page dynamic.
0xC	BEST_FACE	0 = not supported 1 = supported

Tag number	Type	Options
0xD	PLATFORM_TYPE	1 = CPP3 2 = CPP4 3 = CPP-ENC 4 = OTHER

2.16.3 RCP+ over CGI example

Read request

In the example below, the capabilities of the device are requested.

```
http://<IP>/rcp.xml?command=0x0b60&type=P_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0b60</hex>
      <dec>2912</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x0027</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>64</len>
      <str>
00 00 00 0c 00 01 00 01 02 00 04 00 01 00 00 0a 00 01 00 00 05 00 01 01 00
06 00 01 00 00 02 00 01 01 00 03 00 01 00 00  07 00 01 01 00 09 00 01 01 00
08 00 01 00 00 0c 00 01 00 00 0d 00 01 01
      </str>
   </result>
</rcp>
```

Payload description

Hex values	Meaning	Human readable value
00 00 00 0c	Number of Entiries	12
00 01	Tag Number	Device type
00 01	Length	1
02	Camera	2
00 04	Tag Number	Autotracker
00 01	Length	1
00	Value	No

Hex values	Meaning	Human readable value
00 0a	Tag Number	E-PTZ
00 01	Length	1
00	Value	No
00 05	Tag Number	Number of video inputs
00 01	Length	1
01	Value	1
00 06	Tag Number	Number of transcoders
00 01	Length	1
00	Value	0
00 02	Tag Number	PTZ camera
00 01	Length	1
01	Value	Yes
00 03	Tag Number	ROI camera
00 01	Length	1
01	Value	Yes
00 07	Tag Number	Number of audio inputs
00 01	Length	1
01	Value	1
00 09	Tag Number	Number of audio outputs
00 01	Length	1
01	Value	1
00 07	Tag Number	Audio options
00 01	Length	1
00	Value	Line In
00 0c	Tag Number	Best face detection
00 01	Length	1
00	Value	No
00 0c	Tag Number	Platform type
00 01	Length	1
01	Value	CPP3

2.16.4 RCP+ SDK example

Read request

In the example below, the device capabilities are retrieved.

```
   unsigned long  nbrOfEntries = response.readULong();
   for (int i=0; i< nbrOfEntries; i++)
   {
      unsigned short tagType = response.readUShort();
      unsigned short tagLength = response.readUShort();
      if (tagLength==1)
         { byte tagPayload = response.readOctet(); }
      else if (tagLength == 2)
         { unsigned short tagPayload = response.readUShort(); }
      else
         { unsigned long tagPayload = response.readULong(); }

switch (tagType)
     {
     case 0x01://device type
        if(tagPayload==0)
        {
           printf("Device Type: Other\n");
        }
        if(tagPayload==1)
        {
           printf("Device Type: Encoder \n");
        }
        if(tagPayload==2)
        {
           printf("Device Type: Camera \n");
        }
        if(tagPayload==3)
        {
           printf("Device Type: Transcoder \n");
        }
        if(tagPayload==4)
        {
           printf("Device Type: VRM\n");
        }
        if(TagPayload==5)
        {
           printf("Device Type: Decoder \n");
        }
        break;
     case 0x02://PTZ camera
        if(tagPayload==0)
        {
           printf("PTZ Camera: NO\n");
        }
        if(tagPayload==1)
        {
           printf("PTZ Camera: YES\n");
        }
        break;
```

```
      case 0x03://ROI camera
         if(tagPayload==0)
         {
            printf("ROI Camera: NO\n");
         }
         if(tagPayload==1)
         {
            printf("ROI Camera: YES\n");
         }
         break;
      case 0x04://Autotracker
         if(tagPayload==0)
         {
            printf("Autotracker: NO\n");
         }
         if(tagPayload==1)
         {
            printf("Autotracker: YES\n");
         }
         break;
      case 0x05://Number of video inputs
         printf("Number of video inputs %02x\n", tagPayload);
         break;
      case 0x06://Number of Transcoder
printf("Number of transcoders  %02x\n", tagPayload);
         break;
      case 0x07://Number of Audio inputs
         printf("Number of audio inputs %02x\n", tagPayload);
         break;
      case 0x08://Number of Audio outputs
         printf("Number of audio outputs %02x\n", tagPayload);
         break;
      case 0x09://Audio options
         printf("Audio options %02x\n", tagPayload);
                                          printf("For more information see
CONF_AUDIO_OPTIONS\n");
         break;
      case 0x0a://ePTZ
         if(tagPayload==0)
         {
            printf("E-PTZ: NO\n");
         }
         if(tagPayload==1)
         {
            printf("E-PTZ: YES\n");
         }
         break;
      case 0x0b://Image pipe features
         printf("Image pipe features %02x\n", tagPayload);
         break;
```

```
case 0x0c://Best face
   if(tagPayload==0)
   {
      printf("Best face: NO\n");
   }
   if(tagPayload==1)
   {
      printf("Best  face: YES\n");
   }
   break;
case 0x0d://Plattform type
   if(tagPayload==1)
   {
      printf("Platform Type: CPP-3\n");
   }
   if(tagPayload==2)
   {
      printf("Platform Type: CPP-4\n");
   }
   if(tagPayload==3)
   {
      printf("Platform Type: CPP-ENC\n");
   }
   if(tagPayload==4)
   {
      printf("Platform Type: CPP-5\n");
   }
   if(TagPayload==256)
   {
      printf("Platform Type: OTHER\n");
   }
   break;
case 0x0e://BICOM Dome
   if(tagPayload==0)
   {
      printf("BICOM Dome: NO\n");
   }
   if(tagPayload==1)
   {
      printf("BICOM Dome: YES\n");
   }
   break;
case 0x0f://FPGA
   if(tagPayload==0)
   {
      printf("FPGA Tag: NO\n");
   }
   if(tagPayload==1)
   {
      printf("FPGA Tag: YES\n");
```

```
      }
      break;
   case 0x10://VOUT
      if(tagPayload==0)
      {
         printf("VOUT Tag: NO\n");
      }
      if(tagPayload==1)
      {
         printf("VOUT Tag: YES\n");
      }
      break;
   case 0x11://Heater Tag
      if(tagPayload==0)
      {
         printf("Heater Tag: NO\n");
      }
      if(tagPayload==1)
      {
         printf("Heater Tag: YES\n");
      }
      break;
   case 0x12://Serial port tag
      if(tagPayload==0)
      {
         printf("Serial Port Tag: NO\n");
      }
      if(tagPayload==1)
      {
         printf("Serial Port Tag: YES\n");
      }
      break;

   }//switch
}
```

2.17 CONF_DEVICE_TYPE_IDS

Requires firmware version 5.00.30 or higher.

2.17.1 General description

This command enables you to retrieve the unique device identification of a device by its Product, Variant and Frontend Family (BICOM) ID all are hexadecimal.

Tag code: 0x0b07

Numeric descriptor: Not used

Direction	Read		Write
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.17.2 Payload options

Payload structure

Product ID 4 Bytes	Variant ID 4 Bytes	Bicom ID 4 Bytes

Product ID

Identifier for the device backend, like platform, Dome, PTZ Dome, etc.

Variant ID

Identifier for different Product flavours, like with/without IVA, SD Card, etc.

Bicom ID

Identifier for the device frontend (analog camera part).

2.17.3 RCP+ over CGI example

Read request

In the example below, the device ID is retrieved.

```
http://<Device IP>/rcp.xml?command=0x0b07&type=P_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0b07</hex>
      <dec>2823</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x00d8</cltid>
```

```
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>12</len>
      <str>00 00 00 1d 00 00 00 08  00 00 10 13 </str>
   </result>
</rcp>
```

Payload description

Hex values	Meaning	Human readable value
00 00 00 17	Product ID	PRODUCT_ID_DINION
00 00 00 04	Variant ID	VARIANT_ID_DINION_HD_1080p_HDR_IVA
00 00 10 13	Bicom ID	

Note:

A matching chart between the ID's and the devices itself can be found in the ipp.boschsecurity.com download area.

2.17.4 RCP+ SDK example

Read request

In the example below, the device ID is retrieved.

```
client.sendRequest(0, 0x0b070C30, response);
response.wait();
productID = response.readULong();
variantID = response.readULong();
BICOMID = response.readULong();
```

2.18 CONF_DHCP_VAL

Requires firmware version 2.52 or higher.

2.18.1 General description

This command enables you to get and set the DHCP status.

Tag code: 0x00af

Numeric descriptor: Not used

Direction	Read		Write	
Access level	No protection		Service	
Data type	P_OCTET	0x01	P_OCTET	0x01

2.18.2 Payload options

0 = DHCP is switched off

1 = DHCP is switched on

2.18.3 RCP+ over CGI example

Read request

In the example below, the DHCP status is received. In this case DHCP is switched off.

```
http://<Device IP>/rcp.xml?command=0x00af&type=T_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x00af</hex>
      <dec>175</dec>
   </command>
   <type>T_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x026d</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00</hex>
      <dec>0</dec>
   </result>
</rcp>
```

Write request

In the example below, DHCP is switched on.

```
http://<Device IP>/rcp.xml?command=0x00af
&type=T_OCTET&direction=WRITE&payload=1
```

2.18.4 RCP+ SDK example

Read request

In the example below, the DHCP status is received.

```
client.sendRequest(0, 0x00af0130, response);
response.wait();
DHCPValue=response.readOctet();
```

Write request

In the example below, DHCP is switched on.

```
RcpOutputStream payload;
payload.writeOctet(1);
RcpInputStream response;
client.sendRequest(0, 0x00af0131, payload, response);
```

2.19 CONF_GATEWAY_IP_STR

Requires firmware version 2.53 or higher.

2.19.1 General description

This command enable you to read and change the gateway address of the device.

Tag code: 0x007f

Numeric descriptor: Not used

Direction	Read		Write	
Access level	No protection		Service	
Data type	P_STRING	0x10	P_STRING	0x10

2.19.2 Payload options

The payload contains the Gateway address (XXX.XXX.XXX.XXX).

2.19.3 RCP+ over CGI example

Read request

In the example below, the Gateway 160.10.3.0 is received.

```
http://<Device IP>/rcp.xml?command=0x007f&type=P_STRING&direction=READ
```

```
<rcp>
   <command>
      <hex>0x007f</hex>
      <dec>127</dec>
   </command>
   <type>P_STRING</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x001b</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <str>160.10.3.0</str>
   </result>
</rcp>
```

Write request

In the example below, the Gateway is set to 160.10.5.0 on the device.

```
http://<Device IP>/rcp.xml?command=0x007f
&type=P_STRING&direction=WRITE&-payload=160.10.5.0
```

2.19.4 RCP+ SDK example

Read request

In the example below, the Gateway is received.

```
RcpInputStream response;
client.sendRequest(0, 0x007f1030, response);
response.wait();
response.readString(gateway, response.available());
```

Write request

Set the Gateway on the device.

```
RcpOutputStream payload;
payload.writeString(gateway);
RcpInputStream response;
client.sendRequest(0, 0x007f1031, payload, response);
```

2.20 CONF_GET_RTSP_SESSION_ID

Requires firmware version 5.70 or higher.

Note:

Replay of recorded video over RTSP only works for locally managed recordings, not for centrally managed (by VRM) recordings.

This command is NOT writeable.

2.20.1 General description

This command gets the RTSP session ID of the RTSP session, identified by the random value. Precondition is the setup of a replay connection via RTSP:

```
rtsp://<Device IP>/rtsp_tunnel?rec=1&rnd=718
```

Tag code: 0x0ae8

Numeric descriptor: Random value from RTSP session setup

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.20.2 RCP+ over CGI example

Read request

In the example below, the session ID for the RTSP connection number 718 is checked. In this case, the session ID is 154861654.

```
http://<Device IP>/rcp.xml?command=0x0ae8
&type=T_DWORD&num=718&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0ae8</hex>
      <dec>2792</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>718</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <dec>154861654</dec>
   </result>
</rcp>
```

2.20.3 RCP+ SDK example

Read request

In the example below, the session ID for the RTSP connection number is checked.

```
client.setNum(RTSPConnectionNumber);
RcpInputStream response;
client.sendRequest(0, 0x0ae80831, response);
sessionID = response.readULong();
```

2.21 CONF_HD_MGR_REC_STATUS

Requires firmware version 4.10.49 or higher.

2.21.1 General description

This command enables you to retrieve the state of the primary recording. Each state change is also sent as a message.

The response includes the type of recording, e.g. if there is a pre alarm recording or alarm recording, which recording preset and encoder preset is used, and which type of an alarm the recording was triggered, e.g. motion alarm.

This command is also sent as a message. Within the message there is no distinction between the recording state **Off** and **No recording**. If there is a state change between these two states no message will be send. The message will never contain the state **Off** unlike the read response. An **Off** state is the recording configuration by set the recording to stop.

Tag code: 0x0aae

Numeric descriptor: Video line

Direction	**Read request**		**Write request**
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.21.2 Payload options

Payload structure

Recording state 1 Byte	**Recording preset** 1 Byte	**Encoder preset** 1 Byte	**Flags** 1 Byte

Recording state

Values	
0	Off = recording is stopped by the user
1	No Recording = no storage
2	Stand By = no recording on the recording profile
3	Pre-alarm recording
4	Alarm recording
5	Post-alarm recording

Note:

Off means e.g. recording is stopped by the user.

No recording can be caused by many things e.g. no recording on the schedule, no storage present and so on.

Stand by means there is recording on the schedule but not at the moment. The recording scheduler waits for the time to start the recording.

Recording preset

The actual used recording preset from 1 to 10 or 0 if no preset is used.

Encoder preset

The actual used encoder preset from 1 to 8 or 0 if no preset is used.

Flags

These flags show the alarm states and the recording mode.

Values	
Alarm recording mode	0x01
Input alarm recording	0x02
Motion alarm recording	0x04
Video loss recording	0x08
Virtual alarm recording	0x10
Reserved	0x80

2.21.3 RCP+ over CGI example

Read request

In the example below, the recording status from the first video input is retrieved. In this case the recording is in pre-alarm recording, where the video stream is using stream profile 3 and recording configuration is set to recording preset 2.

```
http://<Device IP>/rcp.xml?command=0x0aae
&type=P_OCTET&direction=READ&num=1
```

```
<rcp>
   <command>
      <hex>0x0aae</hex>
      <dec>2734</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
```

```
   <cltid>0x03a1</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>4</len>
      <str>03 02 03 00</str>
   </result>
</rcp>
```

Event message

In the example below, status changes are retrieved as a message.

```
http://<Device IP>/rcp.xml?message=0x0aae&collectms=5000
```

2.21.4 RCP+ SDK example

Read request

```
static void getRecordingState(RcpClient m_RcpClient, byte VideoLine)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   m_RcpClient.setNum(VideoLine);
   m_RcpClient.sendRequest(2000, (OpCode)0x0aae0c30, m_RcpInputStream);
   m_RcpInputStream.wait();
   byte RecState = m_RcpInputStream.readOctet();
   byte RecProfile = m_RcpInputStream.readOctet();
   byte VideoProfile = m_RcpInputStream.readOctet();
   byte Flags = m_RcpInputStream.readOctet();
   string []RecordingStateList = {"Off", "No Recording", "Stand By",
      "Pre Alarm Recording", "Alarm Recording", "Post Alarm Recording"};
   Console.WriteLine("Recording State: "+RecordingStateList[RecState]+",
      used Recording Profile: "+RecProfile+", used Video Profile:
      "+VideoProfile);
   if (Flags>0)
      {
      Console.WriteLine("Alarm Recording Configuration: ");
      if ((byte)(Flags & 0x01) > 0)
         Console.WriteLine("Alarm recording mode");
      if ((byte)(Flags & 0x02) > 0)
         Console.WriteLine("Input Alarm Recording");
      if ((byte)(Flags & 0x04) > 0)
         Console.WriteLine("Motion Alarm Recording");
      if ((byte)(Flags & 0x08) > 0)
         Console.WriteLine("Video Loss Alarm Recording");
      if ((byte)(Flags & 0x10) > 0)
         Console.WriteLine("Virtual Alarm Recording");
      }
   }
```

Event message

```
//Register to messages
OpCode[] p = new OpCode[1];
p[0] = unchecked((OpCode)0x01c00030);
m_RcpClient.subscribeMessageCallback(MessageCallback, m_RcpClient, p);

//Receive the messages
static private void MessageCallback(Object obj, RcpHeader hdr,
RcpInputStream ris)
   {
      switch (hdr.opCode)
      {
         case (OpCode)0x0aae0c30: //Recording state
            byte RecState = ris.readOctet();
            byte RecProfile = ris.readOctet();
            byte VideoProfile = ris.readOctet();
            byte Flags = ris.readOctet();
            bool state = ris.readFlag();
            string []RecordingStateList = {"Off", "No Recording", "Stand By",
               "Pre Alarm Recording", "Alarm Recording",
               "Post Alarm Recording"};
            Console.WriteLine("Recording
               State:"+RecordingStateList[RecState]+",
               used Recording Profile: "+RecProfile+",
               used Video Profile: "+VideoProfile);
            if (Flags > 0)
               {
               Console.WriteLine("Alarm Recoring Configuration: ");
               if ((byte)(Flags & 0x01) > 0)
                  Console.WriteLine("Alarm recording mode");
               if ((byte)(Flags & 0x02) > 0)
                  Console.WriteLine("Input Alarm Recording");
               if ((byte)(Flags & 0x04) > 0)
                  Console.WriteLine("Motion Alarm Recording");
               if ((byte)(Flags & 0x08) > 0)
                  Console.WriteLine("Video Loss Alarm Recording");
               if ((byte)(Flags & 0x10) > 0)
                  Console.WriteLine("Virtual Alarm Alarm Recording");
               }
         break;
      }
   }
```

2.22 CONF_HD_MGR_REC_STATUS_SECONDARY

Requires firmware version 4.10.49 or higher.

2.22.1 General description

This command enables you to retrieve the state of the secondary recording. Each state change is also sent as a message.

The response includes the type of recording, e.g. if there is a pre alarm recording or alarm recording, which recording preset and encoder preset is used, and which type of an alarm the recording was triggered, e.g. motion alarm.

This command is also sent as a message. Within the message there is no distinction between the recording state **Off** and **No recording**. If there is a state change between these two states no message will be send. The message will never contain the state **Off** unlike the read response. An **Off** state is the recording configuration by set the recording to stop.

Tag code: 0x0aaf

Numeric descriptor: Video line

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.22.2 Payload options

Payload structure

Recording state 1 Byte	Recording preset 1 Byte	Encoder preset 1 Byte	Flags 1 Byte

Recording state

Values	
0	Off = recording is stopped by the user
1	No Recording = no storage
2	Stand By = no recording on the recording profile
3	Pre-alarm recording
4	Alarm recording
5	Post-alarm recording

Note:

Off means e.g. recording is stopped by the user.

No recording can be caused by many things e.g. no recording on the schedule, no storage present and so on.

Stand by means there is recording on the schedule but not at the moment. The recording scheduler waits for the time to start the recording.

Recording preset

The actual used recording preset from 1 to 10 or 0 if no preset is used.

Encoder preset

The actual used encoder preset from 1 to 8 or 0 if no preset is used.

Flags

These flags show the alarm states and the recording mode.

Values	
Alarm recording mode	0x01
Input alarm recording	0x02
Motion alarm recording	0x04
Video loss recording	0x08
Virtual alarm recording	0x10
Reserved	0x80

2.22.3 RCP+ over CGI example

Read request

In the example below, the recording status from the first video input is retrieved. In this case the recording is in pre-alarm recording, where the video stream is using stream profile 3 and recording configuration is set to recording preset 2.

```
http://<Device IP>/rcp.xml?command=0x0aaf
&type=P_OCTET&direction=READ&num=1
```

```
<rcp>
   <command>
      <hex>0x0aaf</hex>
      <dec>2734</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x03a1</cltid>
```

```
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>4</len>
      <str>03 02 03 00</str>
   </result>
</rcp>
```

Event message

In the example below, status changes are retrieved as a message.

```
http://<Device IP>/rcp.xml?message=0x0aaf&collectms=5000
```

2.22.4 RCP+ SDK example

Please see example in the **CONF_HD_MGR_REC_STATUS RCP+** section (see page 69).

2.23 CONF_HD_PARTITION_FILE_INFO

Requires firmware version 5.70 or higher.

Note:

Replay of recorded video over RTSP only works for locally managed recordings, not for centrally managed (by VRM) recordings.

This command is NOT writeable.

2.23.1 General description

This RPC+ command is used to get a recording list for locally or centrally managed recording.

Note:

If this command is used to get file information on a span formatted disk, the session ID parameter must be set (a connect primitive must have been preceded). The num paramter (partition) has no meaning then, since a span has always only one partition. Alarm Recording and Time Recording flags changed semantic. Alarm Recording flag marks a file that includes a pre-alarm recording configured by a pre-alarm time in the recording profiles. Time Recording flag marks the files including normal time recording and/or post-alarm. That means a closed file including pre- and post-alarm recordings will always have alarm and time recording flag set. The file ID always increases on span recording regions if new files will be created.

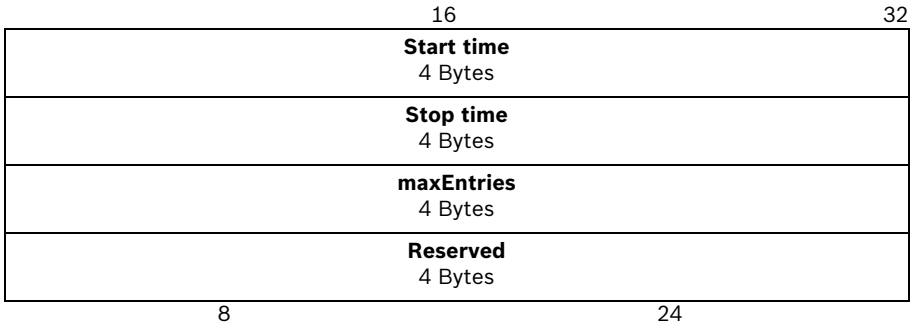
Tag code: 0x0901

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.23.2 Payload options

Payload structure



Start time = seconds since 2000

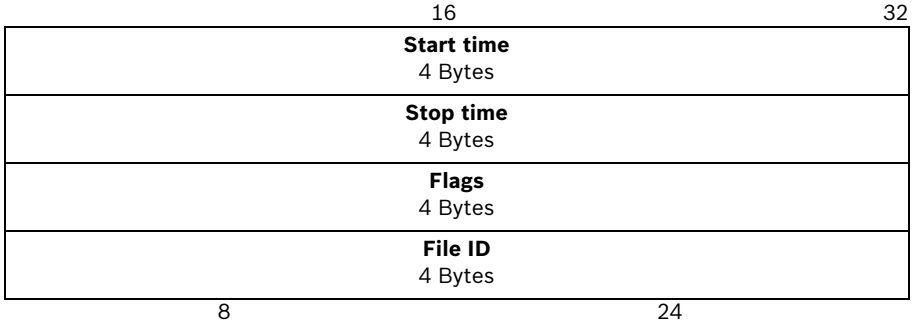
Stop time = seconds since 2000

maxEntries = maximum number of entries

Reserved = not used

2.23.3 Reply

Payload structure (sequence of)



Start time = seconds since 2000

Stop time = seconds since 2000

Flags

Values	
Bit 0	Recording running (actual recording is running on this file or recording not closed regularly)
Bit 1	Recording overwriting (recording takes place in a ring and old recording data will be overwritten)
Bit 2	Alarm input (there are input alarms in this file)
Bit 3	Alarm motion (there are motion alarms in this file)
Bit 4	New alarm (only file info message)
Bit 5	Video loss (there are video loss in this file)

Values	
Bit 6 ... 7	Recording mode: 1 - Time recording 2 - Alarm recording pre-alarm 3 - Alarm recording post-alarm
Bit 8 ... 15	Track fill level (fill level in percent, always 100 % on filled ring recording)
Bit 16	Alarm remote (there are virtual/remote alarms in this file), see **CONF_HD_MGR_SIGNAL_ALARM**
Bit 17 ... 20	Reserved
Bit 21	Offline (VRM only)
Bit 22	Protected (VRM only)
Bit 23 ... 28	Time zone (quarter hours)
Bit 29	Time zone sign
Bit 30 ... 31	Reserved

2.23.4 RCP+ over CGI example

Read request

In the example below, the recorded slices are requested. The start and end time of the search is defined in seconds since 01.01.2000 00:00 h. Also the maximum entry number must be set. The start time is set to 29.11.2012 06:00:00 and the end time is set to 29.11.2012 06:00:00. The maximum entry number is 4.

```
http://<Device IP>/rcp.xml?command=0x0901
&type=P_OCTET&direction=READ&protocol=TCP
&payload=0x1849B660184A0AC00000000400000000
&num=1&sessionid=154861654
```

2.23.5 RCP+ SDK example

Read request

In the example below, the recorded slices are requested.

```
client.setSessionId(sessionID);
RcpInputStream response;
payload.writeULong(startTime);
payload.writeULong(stopTime);
payload.writeULong(maxEntries);
payload.writeULong(0);
client.sendRequest(0, 0x09010C30, payload, response);
response.wait();
```

2.24 CONF_HD_REPLAY_SEEK_TIME

Requires firmware version 5.70 or higher.

Note:

Replay of recorded video over RTSP only works for locally managed recordings, not for centrally managed (by VRM) recordings.

2.24.1 General description

This command gets the current replay time position. Precondition is the setup of a replay connection via RTSP:

```
rtsp://<Device IP>/rtsp_tunnel?rec=1&rnd=718
```

This command will return a write error if the time position is outside a recording set. The session ID is needed.

Tag code: 0x0905

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	P_OCTET	0x0c	P_OCTET	0x0c

2.24.2 Payload options

Payload structure

Seconds 4 Bytes
RTP time stamp 4 Bytes

Seconds = Absolute time in seconds since 1.1.2000 00:00 h.

RTP time stamp = Only in message. RTP time stamp of the first replayed RTP packet of this second. On devices with transcoding functionality, an extended payload can be provided in the write request to specify the Region of Interest.

Extended payload structure

Seconds 4 Bytes	
Millisconds 2 Bytes	**Reserved** 2 Bytes
conf_roi: hPos 2 Bytes	**conf_roi: vPos** 2 Bytes
conf_roi: hSize 2 Bytes	**conf_roi: vSSize** 2 Bytes

Seconds	=	Absolute time in seconds since 1.1.2000 00:00 h.
Milliseconds	=	Milliseconds
RTP time stamp	=	Only in message. RTP time stamp of the first replayed RTP packet of this second. On devices with transcoding functionality, an extended payload can be provided in the write request to specify the Region of Interest.
conf_roi	=	Select region of interest hPos, vPos, hSize, vSize (each entry 2 bytes): starting left upper edge, each 2bytes 0...32768, vSize==0 means: keep aspect ratio.

2.24.3 RCP+ over CGI example

Read request

In the example below, returns the replay seek time for the session ID 154861654. The replay time is an absolute time in seconds since 01.01.2000 00:00 h.

```
http://<Device IP>/rcp.xml?command=0x0905
&type=P_OCTET&direction=READ&sessionid=0x45ae0022
```

```
<rcp>
   <command>
      <hex>0x0905</hex>
      <dec>2309</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x1b5a</</cltid>
   <sessionid>0x45ae0022</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>4</len>
      <str>19 b1 e8 2b</str>
   </result>
</rcp>
```

Write request

In the example below, the seek time is set as absolute time in seconds since 01.01.2000 00:00 h, here: 29.11.2012 11:38:20 (local time).

```
http://<Device IP>/rcp.xml?command=0x0905
&type=P_OCTET&direction=WRITE&sessionid=154861654&payload=0x45ae0022
```

2.24.4 RCP+ SDK example

Read request

In the example below, returns the replay seek time.

```
client.setSessionId(sessionID)
client.sendRequest(0, 0x09050c30, response);
response.wait();
seconds = response.readULong();
```

Write request

In the example below, the replay seek time is set to 29.11.2012 11:38:20.

```
RcpOutputStream payload;
client.setSessionId(sessionID)
payload.writeULong(0x184A05AC);
client.sendRequest(0, 0x09050c31, payload, response);
response.wait();
```

2.25 CONF_HD_REPLAY_START

Requires firmware version 5.70 or higher.

Note:

Replay of recorded video over RTSP only works for locally managed recordings, not for centrally managed (by VRM) recordings.

This command is also sent as a message.

2.25.1 General description

This command returns the t_int parameter in percent of real-time replay (default +100%); 0 if suspended or stopped. The session ID is needed.

The command starts a HD replay at the current position; t_int parameter in percent of real-time replay (default +100%). Negative values will result in a reverse replay.

Tag code: 0x0902

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		User	
Data type	T_INT	0x04	T_INT	0x04

2.25.2 Payload options

Replay speed

Values	
0	Stopped/suspended
100	Real-time replay with 1x speed (100%)
<0	Reverse replay
N × 100	N × speed

2.25.3 RCP+ over CGI example

Read request

In the example below, returns the replay speed.

```
http://<Device IP>/rcp.xml?command=0x0902
&type=T_INT&direction=READ&sessionID=152
```

Write request

In the example below, the current replay speed of real-time replay is set to 100 %.

```
http://<Device IP>/rcp.xml?command=0x0902
&type=T_INT&direction=WRITE&sessionID=152&payload=0x64
```

2.25.4 RCP+ SDK example

Read request

In the example below, returns the replay speed.

```
client.setSessionId(sessionID);
RcpInputStream response;
client.sendRequest(0, 0x09020430, response);
response.wait();
replaySpeed = response.readInt();
```

Write request

In the example below, the replay speed is set.

```
RcpOutputStream payload;
payload.writeInt(replaySpeed);
client.setSessionId(sessionID);
RcpInputStream response;
client.sendRequest(0, 0x09020431, payload, response);
response.wait();
```

2.26 CONF_INPUT_PIN_STATE

Requires firmware version 2.52 or higher.

2.26.1 General description

This command enables you to retrieve the status of the device alarm inputs.

Note:

This command is also sent as a message.

Tag code: 0x01c0

Numeric descriptor: Alarm input

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	F_FLAG	0x00	

2.26.2 Payload options

0 = Alarm input is off

1 = Alarm input is on

2.26.3 RCP+ over CGI example

Read request

In the example below, the status of the first alarm input is retrieved.

http://<Device IP>/rcp.xml?command=0x01c0&type=F_FLAG&direction=READ&num=1

2.26.4 RCP+ SDK example

Read request

```
static bool getInputState(RcpClient m_RcpClient, byte InputNumber)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   m_RcpClient.setNum(InputNumber);
   m_RcpClient.sendRequest(2000, (OpCode)0x01c00030,
   m_RcpInputStream);
   m_RcpInputStream.wait();
   bool state = m_RcpInputStream.readFlag();Console.WriteLine
   ("Input state is " + state); return state;
}
```

Event message

```
//Register to messages
OpCode[] p = new OpCode[1];
```

```
p[0] = unchecked((OpCode)0x01c00030);
m_RcpClient.subscribeMessageCallback(MessageCallback,
m_RcpClient, p);

//Receive the messages
static private void MessageCallback(Object obj, RcpHeader hdr,
RcpInputStream ris)
{
   switch (hdr.opCode)
   {
      case (OpCode)0x01c00030: //Input state
         Console.WriteLine("Input state is " + ris.readFlag());
   }
}
```

2.27 CONF_IP_STR

Requires firmware version 2.53 or higher.

2.27.1 General description

This command enable you to read and change the IP-Address of the device.

Tag code: 0x007c

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	P_STRING	0x10	P_STRING	0x10

2.27.2 Payload options

The payload contains the IP-Address (XXX.XXX.XXX.XXX).

Note:

After changing the IP address and device reboot is needed to apply the settings.

2.27.3 RCP+ over CGI example

Read request

In the example below, the IP-Address 160.10.124.240 is received.

```
http://<Device IP>/rcp.xml?command=0x007c&type=P_STRING&direction=READ
```

```
<rcp>
   <command>
      <hex>0x007c</hex>
      <dec>124</dec>
   </command>
   <type>P_STRING</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x001c</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <str>160.10.124.240</str>
   </result>
</rcp>
```

Write request

In the example below the IP-Address 192.168.1.100 is set on the device.

```
http://<Device IP>/rcp.xml?command=0x007c
&type=P_STRING&direction=WRITE&payload=192.168.1.100
```

2.27.4 RCP+ SDK example

Read request

In the example below, the IP-Address is received.

```
RcpInputStream response;
client.sendRequest(0, 0x007c1030, response);
response.wait();
response.readString(ipAddress, response.available());
```

Write request

Set the IP-Address on the device.

```
RcpOutputStream payload;
payload.writeString(ipAddress);
RcpInputStream response;
client.sendRequest(0, 0x007c1031, payload, response);
```

2.28 CONF_JPEG_BANDWIDTH_KBPS

Requires firmware version 4.00 or higher.

2.28.1 General description

This command enables you to get and set the JPEG bandwidth.

Note:

This command is valid for CPP-ENC devices only. For other devices, use the **CONF_JPEG_STREAM_SETUP** command.

Tag code: 0x061d

Numeric descriptor: Profile preset

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	T_DWORD	0x08	T_DWORD	0x08

2.28.2 Payload options

Get or set the JPEG bandwidth (in KBPS) of the selected preset for the JPEG streaming.

2.28.3 RCP+ over CGI example

Read request

In the example below the JPEG bandwidth of the first preset is retrieved. In this case it is set to 6000 kbps.

```
http://<Device IP>/rcp.xml?command=0x061d
&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x061d</hex>
      <dec>1565</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00001770</hex>
      <dec>6000</dec>
```

```
   </result>
</rcp>
```

Write request

In the example below the JPEG bandwidth of the first preset is set to 5000 kbps.

```
http://<Device IP>/rcp.xml?command=0x061d
&type=T_DWORD&num=1&direction=WRITE&payload=5000
```

2.28.4 RCP+ SDK example

Read request

In the example below the JPEG bandwidth of the first preset is retrieved.

```
unsigned long bandwidthJPEG;
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x061d0830, response);
response.wait();
bandwidthJPEG = response.readULong();
```

Write request

In the example below the JPEG bandwidth of the first preset is set.

```
RcpOutputStream payload;
payload.writeULong(bandwidthJPEG);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x061d0831, payload, response);
response.wait();
```

2.29 CONF_JPEG_STREAM_SETUP

Requires firmware version 4.21.19 or higher.

2.29.1 General description

This command represents the JPEG stream configuration.

Note:

This command is not valid for CPP-ENC devices. For CPP-ENC devices use the **CONF_JPEG_BANDWIDTH_KBPS** command instead.

Tag code: 0x0ad5

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	P_OCTET	0x0c	P_OCTET	0x0c

2.29.2 Payload options

Payload structure

Resolution 4 Bytes	Frames in MHz 4 Bytes	Quality 4 Bytes

Resolution

Values	
0	QCIF
1	CIF
2	2CIF
3	4CIF
6	QVGA
7	VGA
12	720p
14	1080p
15	5MP
16	1280x960
17	1440x1080

Frames in MHz

Frames in MHz. To get fps, the value has to be divided by 1000.

Quality

Quality of the JPEG in the range 1 ... 100.

0 = Automatic quality settings

1 = Worst quality

100 = Best quality

2.29.3 RCP+ over CGI example

Read request

In the example below, the jpeg stream configuration is retrieved.

```
http://<Device IP>/rcp.xml?command=0x0ad5&type=P_OCTET&direction=READ
```

Write request

In the example below, the jpeg resolution is set to 720p with full 30 fps and medium quality settings.

```
http://<Device IP>/rcp.xml?command=0x0ad5
&type=P_OCTET&direction=WRITE&payload=00000012000753000000032
```

2.29.4 RCP+ SDK example

Read request

```
static void getJpegStreamSetup(RcpClient m_RcpClient)
{

   RcpInputStream m_RcpInputStream = new RcpInputStream();
   m_RcpClient.sendRequest(2000, (OpCode)0x0ad50c30, m_RcpInputStream);
   m_RcpInputStream.wait();
   ulong Resolution = m_RcpInputStream.readULong();
   ulong FPS = m_RcpInputStream.readULong() / 1000; //Framerate in MHZ
   ulong Quality = m_RcpInputStream.readULong();
   Console.WriteLine("Jpeg Stream: Resolution: " +
   g_resolutionList[Resolution] + ", FPS: " + FPS + ", Quality: " + Quality);
}
```

Write request

```
static void setJpegStreamSetup(RcpClient m_RcpClient, ulong Resolution,
ulong FPS, ulong Quality)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   RcpOutputStream m_RcpOutputStream = new RcpOutputStream();
   m_RcpOutputStream.writeULong(Resolution);
   m_RcpOutputStream.writeULong(FPS * 1000);
   m_RcpOutputStream.writeULong(Quality);
   m_RcpClient.sendRequest(2000, (OpCode)0x0ad50c31, m_RcpOutputStream,
   m_RcpInputStream);
```

```
            m_RcpInputStream.wait();
        }
```

2.30 CONF_MAC_ADDRESS

Requires firmware version 2.53 or higher.

2.30.1 General description

This command enable you to read the MAC address of the device.

Tag code: 0x00bc

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.30.2 Payload options

The payload contains the MAC address.

2.30.3 RCP+ over CGI example

Read request

In the example below, the MAC address 00 04 63 3f ce f8 is received.

```
http://<Device IP>/rcp.xml?command=0x00bc&type=P_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x00bc</hex>
      <dec>188</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x001d</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>6</len>
      <str>00 04 63 3f ce f8 </str>
   </result>
</rcp>
```

2.30.4 RCP+ SDK example

Read request

```
client.sendRequest(0, 0x00bc0C30, response);
response.wait();
response.read(MACAddress, response.available());
```

2.31 CONF_MANAGING_VRM

Requires firmware version 4.50 or higher.

2.31.1 General description

This RPC+ command can be used to manage a VRM. Set or read the IP address, port and user and to set the password of the managing VRM and the backup VRM.

Tag code: 0x0aeb

Numeric descriptor: Recording index (1 = primary, 2 = secondary)

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	P_OCTET	0x0c	P_OCTET	0x0c

2.31.2 Payload options

Payload structure

IP 4 Bytes		
Port 2 Bytes	**Flag** 1 Byte	**Reserved** 1 Byte
User 32 Byte		
Password 32 Byte		
Backup IP 4 Bytes		
Backup port 2 Bytes	**Reserved** 2 Bytes	

Note:

The command cannot be used for reading the VRM password, it will return the string ********** instead.

IP

IP address of the managing VRM.

Port

Port of the managing VRM.

Flags

Additional flags.

Values	
0x01	USE_SSL

User

Max 31 ASCII character string with zero termination. In case the user is shorter than 31 characters, the remaining bytes need to be filled up with 0x00 values.

Password

Max 31 ASCII character string with zero termination. In case the password is shorter than 31 characters, the remaining bytes need to be filled up with 0x00 values.

Backup IP

IP address of the backup VRM.

Backup port

Port of the backup VRM.

2.31.3 RCP+ over CGI example

In the example below, the managing VRM IP is retrieved. Please see payload details below.

```
http://<Device IP>/rcp.xml?command=0x0aeb
&type=P_OCTET&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0aeb</hex>
      <dec>2795</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x1b6a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>80</len>
      <str>
a0 0a 7c fc 06 dc 00 00 5f 5f 64 65 76 69 63 65  72 65 70 6c 61 79 5f 5f  00
00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 2a 2a 2a 2a 2a 2a 2a 2a  00 00
00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 00 00 00
00 06 dc 00 00
</str>
   </result>
</rcp>
```

Payload description

Hex values	Meaning	Human readable meaning
a0 0a 7c fc	VRM IP	160.10.124.252
06 dc	VRM Port	1756
00	Flag	No SSL
00	Reserved	--
5f 5f 64 65 76 69 63 65 72 65 70 6c 61 79 5f 5f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	User Name	__devicereplay__
2a 2a 2a 2a 2a 2a 2a 2a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	Password	********
00 00 00 00	Backup VRM IP	No backup VRM
06 dc	Backup VRM Port	
00 00	Reserved	--

2.31.4 RCP+ SDK example

Read request

In the example below, the managing VRM IP is retrieved.

```
client.sendRequest(0, 0x0aeb0C30, response);
response.wait();
      vrmip = response.readULong(); //VRM IP
      vrmport = response.readUShort();// VRM Port
      flags = response.readOctet();//1= SSL
      response.readOctet(); //skip the reserved

      for (int i=0; i<32; i++)
         username[i] = response.readOctet();
      for (int i=0; i<32; i++)
         password[i] = response.readOctet();

      backupip = response.readULong();
      backupport = response.readUShort();


      convertIP(vrmip, *vrmipstr);
      convertIP(backupip, *backupipstr);

      printf("\n%04x --> VRM IP: %s\n", vrmip, vrmipstr);
      printf("%02d --> VRM Port: %d\n", vrmport, vrmport);
      printf("%01x --> Flags: %d\n", flags, flags);

      printf("Username: %s\n", username);
      printf("Password: %s\n", password);
```

```
printf("%04x --> Backup VRM IP: %d\n", backupip, backupip);
printf("%02d --> Backup VRM Port: %d\n", backupport, backupport);
```

2.32 CONF_MOTION_ALARM_STATE

Requires firmware version 2.52 or higher.

2.32.1 General description

This command enables you to retrieve the status of the device motion detection.

If Intelligent Video Alarm (IVA) is enabled, this command reports a motion alarm when at least one of the IVA tasks is triggered. For detailed IVA alarms, please see the **CONF_VIPROC_ALAM** section (see page 168).

Note:
This command is also sent as a message.

Tag code: 0x01c3

Numeric descriptor: Video line

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	F_FLAG	0x0c	

2.32.2 Payload options

0 = Motion is off

1 = Motion is on

2.32.3 RCP+ over CGI example

Read request

In the example below, the status of the motion detection is retrieved.

```
http://<Device IP>/rcp.xml?command=0x01c3&type=F_FLAG&direction=READ&num=1
```

Message

In the example below, status changes are retrieved as a message.

```
http://<Device IP>/rcp.xml?message=0x01c3&collectms=5000
```

2.32.4 RCP+ SDK example

```
static bool getMotionState(RcpClient m_RcpClient, byte VideoLine)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   m_RcpClient.setNum(VideoLine);
   m_RcpClient.sendRequest(2000, (OpCode)0x01c30030,
   m_RcpInputStream);
   m_RcpInputStream.wait();
   bool state = m_RcpInputStream.readFlag();
```

```
   Console.WriteLine("Motion state is " + state);
   return state;
}
```

Message

In the example below, status changes are retrieved as a message.

```
//Register to messages
OpCode[] p = new OpCode[1];
p[0] = unchecked((OpCode)0x01c30030);
m_RcpClient.subscribeMessageCallback(MessageCallback,
m_RcpClient, p);

//Receive the messages
static private void MessageCallback(Object obj, RcpHeader hdr,
RcpInputStream ris)
{
   switch (hdr.opCode)
   {
      case (OpCode)0x01c30030: //Motion state
         Console.WriteLine("Motion state is " + ris.readFlag());
   }
}
```

2.33 CONF_MPEG4_BANDWIDTH_KBPS

Requires firmware version 3.00 or higher.

2.33.1 General description

This command reads out and sets the target bit rate (in kbps) of the selected preset.

Tag code: 0x0607

Numeric descriptor: Profile preset

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	T_DWORD	0x08	T_DWORD	0x08

2.33.2 Payload options

Payload is the target bit rate in kbps.

2.33.3 RCP+ over CGI example

Read request

In the example below the bandwidth of the first preset is retrieved. In this case it is set to 600 kbps.

```
http://<Device IP>/rcp.xml?command=0x0607
&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0607</hex>
      <dec>1543</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000258</hex>
      <dec>600</dec>
   </result>
</rcp>
```

Write request

In the example below, the bandwidth of the first preset is set to 3000 kbps.

```
http://<Device IP>/rcp.xml?command=0x0607
&type=T_DWORD&num=1&direction=WRITE&payload=3000
```

2.33.4 RCP+ SDK example

Read request

In the example below the bandwidth of the first preset is retrieved.

```
unsigned long bandwidth;
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x06070830, response);
response.wait();
bandwidth = response.readULong();
```

Write request

In the example below, the bandwidth of the first preset is set.

```
RcpOutputStream payload;
payload.writeULong(bandwidth);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x06070831, payload, response);
response.wait();
```

2.34 CONF_MPEG4_BANDWIDTH_KBPS_SOFT_LIMIT

Requires firmware version 3.00 or higher.

2.34.1 General description

By means of this command you may read out and set the maximum bit rate (in kbps) of the selected preset.

Tag code: 0x0612

Numeric descriptor: Profile preset

Direction	**Read request**		**Write request**	
Access level	No protection		Service	
Data type	T_DWORD	0x08	T_DWORD	0x08

2.34.2 Payload options

Payload is the maximum bit rate in kbps.

2.34.3 RCP+ over CGI example

Read request

In the example below the bandwidth soft limit of the first preset is retrieved. In this case it is set to 22528 kbps.

```
http://<Device IP>/rcp.xml?command=0x0612
&type=T_DWORD&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0612</hex>
      <dec>1554</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00005800</hex>
      <dec>22528</dec>
   </result>
</rcp>
```

Write request

In the example below, the bandwidth soft limit of the first preset is set to 20000.

```
http://<Device IP>/rcp.xml?command=0x0612
&type=T_DWORD&num=1&direction=WRITE&payload=20000
```

2.34.4 RCP+ SDK example

Read request

In the example below the bandwidth soft limit of the first preset is retrieved.

```
unsigned long bandwidthMax;
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x06120830, response);
response.wait();
bandwidthMax = response.readULong();
```

Write request

In the example below, the bandwidth soft limit of the first preset is set.

```
RcpOutputStream payload;
payload.writeULong(bandwidthMax);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x06120831, payload, response);
response.wait();
```

2.35 CONF_MPEG4_CURRENT_PARAMS_REL_CODER

Requires firmware version 3.00 or higher.

2.35.1 General description

This command gets/sets a video configuration profile for a specified video stream.

Tag code: 0x061c

Numeric descriptor: Not used

Direction	**Read request**		**Write request**	
Access level	No protection		Service	
Data type	P_OCTET	0x0c	P_OCTET	0x0c

2.35.2 Payload options

Payload structure

Line 1 Byte	**Coder** 1 Byte	**Coder capabilities** 2 Bytes
Preset 1 Byte	**Reserved** 3 Bytes	

Line = Video input line

Coder = 1= First video stream
2= Second video stream
3= Third video stream (JPEG)

Note:

This command needs a payload in the read request.

On read requests the coder capabilities and preset parameters can be ignored. The according bytes will be set in the reply payload.

Coding capabilities

Values	
0x0002	H.263
0x0004	MPEG 4
0x0008	MPEG 2
0x0040	H.264
0x0080	JPEG

Note:

Please see **CONF_CAPABILITY_LIST** command for detailed line, coder, and coding capabilities of the device.

Preset

Number of the profile the coder is set to/has to be set to.

2.35.3 Reply

Reply Payload Structure

The replay payload structure is identical to the request payload structure, please see above.

2.35.4 RCP+ over CGI example

Read request

In the example below, the profile for the second stream on video input 1 is requested.

```
http://<Device IP>/ rcp.xml?command=0x061c
&type=P_OCTET&direction=READ&payload=0x0102000000000000
```

Write request

In the example below, profile 4 is assigned to video stream 1.

```
http://<Device IP>/ rcp.xml?command=0x061c
&type=P_OCTET&direction=WRITE&payload=0x0101000004000000
```

2.35.5 RCP+ SDK example

Read request

In the example below, the profile for the given line/stream requested.

```
static int getVideoProfile(RcpClient m_RcpClient, byte Line, byte Stream)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   RcpOutputStream m_RcpOutputStream = new RcpOutputStream();
   m_RcpOutputStream.writeOctet(Line); //Line
   m_RcpOutputStream.writeOctet(Stream); //Stream
   m_RcpOutputStream.writeUShort(0);
   m_RcpOutputStream.writeOctet(0);

   m_RcpClient.sendRequest(2000, (OpCode)0x061c0c30, m_RcpOutputStream,
   m_RcpInputStream);
   m_RcpInputStream.wait();
   int videoLine, videoStream, codingFlags, videoProfile;
   videoLine = m_RcpInputStream.readOctet();
   videoStream = m_RcpInputStream.readOctet();
   codingFlags = m_RcpInputStream.readUShort();
   videoProfile = m_RcpInputStream.readOctet();
```

```
   Console.WriteLine(videoLine + ". Line, " + videoStream + ".
   Stream is set to profile: " + videoProfile);
   return videoProfile;
}
```

Write request

In the example below, profile is set to the given line/stream.

```
static void setVideoProfile(RcpClient m_RcpClient, byte Line, byte Stream,
byte Profile)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   RcpOutputStream m_RcpOutputStream = new RcpOutputStream();
   m_RcpOutputStream.writeOctet(Line);
   m_RcpOutputStream.writeOctet(Stream);
   m_RcpOutputStream.writeUShort(0);
   m_RcpOutputStream.writeOctet(Profile);
   m_RcpOutputStream.writeOctet(0);
   m_RcpOutputStream.writeUShort(0);

   m_RcpClient.sendRequest(2000, (OpCode)0x061c0c31, m_RcpOutputStream,
   m_RcpInputStream);
   m_RcpInputStream.wait();
}
```

2.36 CONF_MPEG4_FRAME_SKIP_RATIO

Requires firmware version 2.52 or higher.

2.36.1 General description

With this command the number of encoded frames can be changed/retrieved for a video profile.

First, the device base frame rate has to be retrieved with the command **CONF_VIDEO_INPUT_FORMAT_EX** (see page 163), according to the formula:

base frame rate/**MPEG4_FRAME_SKIP_RATIO** = frames per second

The value **1** means all frames are encoded.

Tag code: 0x0606

Numeric descriptor: Video profile preset number

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	T_DWORD	0x08	T_DWORD	0x08

2.36.2 Payload options

Payload structure

Payload is encoding interval. 1 means all frames are encoded, 2 every second frame is skipped, etc.

2.36.3 RCP+ over CGI example

Read request

In the example below, the frame skip value is retrieved for the second video profile.

```
http://<Device IP>/rcp.xml?command=0x0606
&type=T_DWORD&direction=WRITE&num=1&payload=3
```

2.36.4 RCP+ SDK example

Read request

In the example below, the frame skip value is retrieved.

```
static uint getFrameSkipRatio(RcpClient m_RcpClient, byte Profile
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   m_RcpClient.setNum(Profile);
   m_RcpClient.sendRequest(2000, (OpCode)0x06060830, m_RcpInputStream);
   m_RcpInputStream.wait();
   uint skipRatio = m_RcpInputStream.readULong();
   Console.WriteLine("Skip Ratio for Profile " + Profile + " is set to "
   + skipRatio);
```

```
   return skipRatio;
}
```

Write request

In the example below, the frame rate is set for the given profile.

```
static void setFrameSkipRatio(RcpClient m_RcpClient, byte Profile, uint
SkipRatio)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   RcpOutputStream m_RcpOutputStream = new RcpOutputStream();
   m_RcpOutputStream.writeULong(SkipRatio);
   m_RcpClient.setNum(Profile);
   m_RcpClient.sendRequest(2000, (OpCode)0x06060831, m_RcpOutputStream,
   m_RcpInputStream);
   m_RcpInputStream.wait();
}
```

2.37 CONF_MPEG4_RESOLUTION

Requires firmware version 2.52 or higher.

2.37.1 General description

This command gets/sets the resolution for a video profile.

Note:

The resolution in the video profile are only valid for SD streams, not for HD streams. For changing HD resolution please use the **CONF_VIDEO_H264_ENC_BASE_OPERATION_MODE** command (see page 157).

Tag code: 0x0608

Numeric descriptor: Video profile preset

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	T_DWORD	0x08	T_DWORD	0x08

2.37.2 Payload options

Values	
0	QCIF
1	CIF
2	2CIF
3	4CIF
4	$^1/_2$ D1
5	$^2/_3$ D1
6	QVGA
7	VGA
8	WD144 (256 × 144)
9	WD288 (512 × 288)
10	WD432 (768 × 432)
12	720p
14	1080p
15	5MP

2.37.3 RCP+ over CGI example

Read request

In the example below, the resolution of profile 7 is checked.

```
http://<Device IP>/rcp.xml?command=0x0608
&type=T_DWORD&direction=READ&num=7
```

Write request

In the example below, VGA resolution is assigned for profile 6.

```
http://<Device IP>/rcp.xml?command=0x0608
&type=T_DWORD&direction=WRITE&num=6&payload=7
```

2.37.4 RCP+ SDK example

Read request

In the example below, the resolution for the given profile is checked.

```
static uint getResolution(RcpClient m_RcpClient, byte Profile)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   m_RcpClient.setNum(Profile);
   m_RcpClient.sendRequest(2000, (OpCode)0x06080830, m_RcpInputStream);
   m_RcpInputStream.wait();
   uint resolution = m_RcpInputStream.readULong();
   Console.WriteLine("Resolution for Profile " + Profile + " is set to " +
   g_resolutionList[resolution]);
   return resolution;
}
```

Write request

In the example below, the resolution for the given profile is set.

```
static void setResolution(RcpClient m_RcpClient, byte Profile, uint
Resolution)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   RcpOutputStream m_RcpOutputStream = new RcpOutputStream();
   m_RcpOutputStream.writeULong(Resolution);
   m_RcpClient.setNum(Profile);
   m_RcpClient.sendRequest(2000, (OpCode)0x06080831, m_RcpOutputStream,
   m_RcpInputStream);
   m_RcpInputStream.wait();
}
```

2.38 CONF_NAME_STAMP_VAL

Requires firmware version 3.00 or higher.

2.38.1 General description

This command enables/disables video overlays for stamping of the camera name.

Tag code: 0x0084

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		User	
Data type	T_OCTET	0x01	T_OCTET	0x01

2.38.2 Payload options

0 = camera name stamping off

1 = camera name stamping at the bottom

2 = camera name stamping at the top

3 = camera name stamping with custom attributes

With the option 3 it is possible to display the stamping at a custom x and y position. For setting the values for x and y see command **CONF_STAMP_ATTR_NAME** (see page 141).

2.38.3 RCP+ over CGI example

Read request

In the example below, the value of the name stamping is requested. In this case, it is set to custom attributes.

```
http://<Device IP>/rcp.xml?command=0x0084&type=T_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0084</hex>
      <dec>132</dec>
   </command>
   <type>T_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x03</hex>
      <dec>3</dec>
```

```
   </result>
</rcp>
```

Write request

In the example below, the name stamping mode is set to bottom.

```
http://<Device IP>/rcp.xml?command=0x0084
&type=T_OCTET&direction=WRITE&payload=1
```

2.38.4 RCP+ SDK example

Read request

In the example below, the name stamping mode is checked.

```
RcpInputStream response;
client.sendRequest(0, 0x00840130, response);
response.wait();
```

Write request

In the example below, the name stamping mode is set to bottom.

```
RcpOutputStream payload;
payload.writeOctet(1);
RcpInputStream response;
client.sendRequest(0, 0x00840131, payload, response);
response.wait();
```

2.39 CONF_NBR_OF_ALARM_IN

Requires firmware version 2.52 or higher.

2.39.1 General description

This command enables you to get the number of installed alarm inputs.

Tag code: 0x01db

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.39.2 Payload options

Returns the number of Alarm inputs.

2.39.3 RCP+ over CGI example

Read request

In the example below, the number of Alarm inputs is retrieved. In this case there are two inputs available.

```
http://<Device IP>/rcp.xml?command=0x01db&type=T_DWORD&direction=READ
```

```
<rcp>
   <command>
      <hex>0x01db</hex>
      <dec>475</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x03a4</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000002</hex>
      <dec>2</dec>
   </result>
</rcp>
```

2.39.4 RCP+ SDK example

Read request

In the example below, the number of alarm inputs is retrieved.

```
unsigned long NbrAlarmIn;
client.sendRequest(0, 0x01db0830, response);
response.wait();
NbrAlarmIn = response.readULong();
```

2.40 CONF_NBR_OF_ALARM_OUT

Requires firmware version 2.52 or higher.

2.40.1 General description

This command enables you to get the number of installed alarm output contacts.

Tag code: 0x01dc

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.40.2 Payload options

Returns the number of alarm outputs.

2.40.3 RCP+ over CGI example

Read request

In the example below, the number of Alarm outputs is retrieved. In this case there are two outputs available.

```
http://<Device IP>/rcp.xml?command=0x01dc&type=T_DWORD&direction=READ
```

```
<rcp>
   <command>
      <hex>0x01dc</hex>
      <dec>476</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x03aa</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000001</hex>
      <dec>1</dec>
   </result>
</rcp>
```

2.40.4 RCP+ SDK example

Read request

In the example below, the number of alarm outputs is retrieved.

```
unsigned long NbrAlarmOut;
client.sendRequest(0, 0x01dc0830, response);
response.wait();
NbrAlarmOut = response.readULong();
```

2.41 CONF_NBR_OF_AUDIO_IN

Requires firmware version 2.52 or higher.

2.41.1 General description

This command enables you to get the number of audio inputs.

Tag code: 0x01d8

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.41.2 Payload options

Returns the number of audio inputs.

2.41.3 RCP+ over CGI example

Read request

In the example below, the number of audio inputs is retrieved. In this case there is one Audio input available.

```
http://<Device IP>/rcp.xml?command=0x01d8&type=T_DWORD&direction=READ
```

```
<rcp>
   <command>
      <hex>0x01d8</hex>
      <dec>475</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x03ab</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000001</hex>
      <dec>1</dec>
   </result>
</rcp>
```

2.41.4 RCP+ SDK example

Read request

In the example below, the number of audio inputs is retrieved.

```
unsigned long NbrAudioIn;
client.sendRequest(0, 0x01d80830, response);
response.wait();
NbrAudioIn = response.readULong();
```

2.42 CONF_NBR_OF_AUDIO_OUT

Requires firmware version 2.52 or higher.

2.42.1 General description

This command enables you to get the number of installed audio outputs.

Tag code: 0x01d9

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.42.2 Payload options

Returns the number of audio outputs.

2.42.3 RCP+ over CGI example

Read request

In the example below, the number of audio outputs is retrieved. In this case there is one audio output available.

```
http://<Device IP>/rcp.xml?command=0x01d9&type=T_DWORD&direction=READ
```

```
<rcp>
   <command>
      <hex>0x01d9</hex>
      <dec>476</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x03ad</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000001</hex>
      <dec>1</dec>
   </result>
</rcp>
```

2.42.4 RCP+ SDK example

Read request

In the example below, the number of audio outputs is retrieved.

```
unsigned long NbrAudioOut;
client.sendRequest(0, 0x01d90830, response);
response.wait();
NbrAudioOut = response.readULong();
```

2.43 CONF_NBR_OF_VIDEO_IN

Requires firmware version 2.52 or higher.

2.43.1 General description

This command returns the number of video inputs.

Tag code: 0x01d6

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.43.2 Payload options

Returns the number of installed video inputs.

2.43.3 RCP+ over CGI example

Read request

In the example below, the number of video inputs is retrieved. In this case there is one video input available.

```
http://<Device IP>/rcp.xml?command=0x01d6&type=T_DWORD&direction=READ
```

```
<rcp>
   <command>
      <hex>0x01d6</hex>
      <dec>470</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000001</hex>
      <dec>1</dec>
   </result>
</rcp>
```

2.43.4 RCP+ SDK example

Read request

In the example below, the number of video inputs is retrieved.

```
unsigned long NbrVideoIn;
client.sendRequest(0, 0x01d60830, response);
response.wait();
NbrVideoIn = response.readULong();
```

2.44 CONF_NBR_OF_VIDEO_OUT

Requires firmware version 2.52 or higher.

2.44.1 General description

This command returns the number of video outputs.

Tag code: 0x01d7

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.44.2 Payload options

Returns the number of installed video outputs.

2.44.3 RCP+ over CGI example

Read request

In the example below, the number of video outputs is retrieved. In this case there are two video outputs available.

```
http://<Device IP>/rcp.xml?command=0x01d7&type=T_DWORD&direction=READ
```

```
<rcp>
   <command>
      <hex>0x01d7</hex>
      <dec>471</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000002</hex>
      <dec>2</dec>
   </result>
</rcp>
```

2.44.4 RCP+ SDK example

Read request

In the example below, the number of video outputs is retrieved.

```
unsigned long NbrVideoOut;
client.sendRequest(0, 0x01d70830, response);
response.wait();
NbrVideoOut = response.readULong();
```

2.45 CONF_NBR_OF_VIRTUAL_ALARMS

Requires firmware version 4.50 or higher.

2.45.1 General description

This command returns the number of virtual alarm inputs.

Tag code: 0x0aed

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	T_DWORD	0x08	

2.45.2 Payload options

Returns the number of virtual alarm inputs.

2.45.3 RCP+ over CGI example

Read request

In the example below, the number of virtual inputs is retrieved. In this case there are four inputs available.

```
http://<Device IP>/rcp.xml?command=0x0aed&type=T_DWORD&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0aed</hex>
      <dec>2797</dec>
   </command>
   <type>T_DWORD</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00000004</hex>
      <dec>4</dec>
   </result>
</rcp>
```

2.45.4 RCP+ SDK example

Read request

In the example below, the number of virtual inputs is retrieved.

```
unsigned long NbrVirtualIn;
client.sendRequest(0, 0x0aed0830, response);
response.wait();
NbrVirtualIn = response.readULong();
```

2.46 CONF_OEM_DEVICE_NAME

Requires firmware version 2.52 or higher.

2.46.1 General description

This command enable you to get the OEM device name of the device.

Tag code: 0x097c

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	P_STRING	0x10	

2.46.2 Payload options

It contains the OEM device name.

2.46.3 RCP+ over CGI example

Read request

In the example below, the OEM name of the device is returned. In this case it is a DINION HD 720p IVA.

```
http://<Device IP>/rcp.xml/?command=0x097c&type=P_STRING&direction=READ
```

```
<rcp>
   <command>
      <hex>0x097c</hex>
      <dec>2428</dec>
   </command>
   <type>P_STRING</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x029f</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <str>DINION HD 720p IVA</str>
   </result>
</rcp>
```

2.46.4 RCP+ SDK example

Read request

In the example below, the OEM name of the device is returned.

```
char OEMDeviceName[128];
client.sendRequest(0, 0x097c1030, response);
response.wait();
response.readString(OEMDeviceName, response.available());
```

2.47 CONF_RCP_TRANSFER_TRANSPARENT_DATA

Requires firmware version 3.00 or higher.

Note:

This command is also send as a message.

2.47.1 General description

The transparent data from and to the serial interfaces is handled by the RCP to achieve reliable transfer of information. To gather control over the remote serial interface a successful registration is necessary.

Tag code: 0xffdd

Numeric descriptor: COM port

Direction	Read request	Write request	
Access level	Not supported	User	
Data type		P_OCTET	0x0c

2.47.2 Payload options

Payload structure

Options 1 Byte	Reserved 1 Byte	Lease time 2 Bytes
Trans. data 1 1 Byte	N × 1 Byte	Trans. data N 1 Byte

Options

Currently no options used.

Lease time

Time in seconds the lease is requested.

Values	
0x0000	Only this packet should be sent out; no further control is requested.
0xFFFF	Indefinite lease time; request lease as long as the current registration is valid.

Note:

The lease time should be treated as a request; the VideoJet may switch leases before the requested time is over due to a higher prioritized RCP client.

For this direction, an RCP write command to a specific port (addressed by the numeric descriptor) is necessary.

Once the VideoJet has received a **TRANSFER_TRANSPARENT_DATA** command, it checks whether the RCP client is in control or not. If the RCP client is allowed to send data to the serial interface, the reply will present an **OK**. If the RCP client is not allowed to send data, a **FAIL** will be returned. In this case another RCP client is controlling the serial out. The timeout and priority handling of the serial ports is beyond the scope of this document.

2.47.3 Write reply packet

Write request

Code 1 Byte	**Reserved** 1 Byte	**Reserved** 2 Bytes

Code

Values	
0x00	Access to the serial port denied.
0x01	Access to the serial port granted.

2.47.4 Read request

The reply to the read request command will be the same as the reply to the write request command. The returned code will present the availability of the serial port.

Note:

Despite a positive reply to a read command, the port may be locked by another RCP client in the time slice between the read and a following write command.

2.47.5 Serial IN -> RCP Client

The data coming from the serial input is delivered using an RCP message. All RCP clients which want to receive this data must be registered for the message 0xFFDD. Data is posted if the corresponding RCP client is in control only.

If no client has a lease on the serial port, a message to all registered clients will be generated.

Note:

The received message will carry NO header.

2.47.6 RCP+ over CGI example

Write request

In the example below, the Bosch Autodome protocol is used to continuous pan to the left with maximum speed.

For other options and the command syntax, please check the OSRD documentation.

```
http://<Device IP>/rcp.xml?command=0xFFDD&
type=P_OCTET&direction=WRITE&num=1&payload=0x000000058700000500780206
```

2.47.7 RCP+ SDK example

Write request

In the example below, the Bosch Autodome protocol is used to stop all PTZ operations.

```
byte[] payloadBytes = new byte[12];

payloadBytes [0] = 0x00;  //Options - currently not used
payloadBytes [1] = 0x00;  /Reserved
payloadBytes [2] = 0x00;  /lease time high
payloadBytes [3] = 0x00;  //lease time low
payloadBytes [4] = 0x87;  //8x where x is the length
payloadBytes [5] = 0x00;  //camera address hi order
payloadBytes [6] = 0x00;  //camera address low order
payloadBytes [7] = 0x05;  //opcode//variable speed-should be good
                          //for the most cases
payloadBytes [8] = 0x00;  //d1: 0-3 Tilt speed, 4-6 Zoom speed
payloadBytes [9] = 0x00;  //d2: 0:Focus far, 1:Iris Close, 2:Iris Open,
                          //3-6 Pan speed
payloadBytes [10] = 0x00; //d3: 0:Pan Right, 1:Pan Left, 2: Tilt Down,
                          //3: Tilt Up
                          //4: Zoom Out, 5: Zoom In, 6: Focus Near

payloadBytes [11] = (byte)(( payloadBytes [4] +
   payloadBytes [5] +
   payloadBytes [6] +
   payloadBytes [7] +
   payloadBytes [8] +
   payloadBytes [9] +
   payloadBytes [10]) & 0x7F); //checksum

payload.write(payloadByte, 11);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0xFFDD0C31, payload, response);
```

2.48 CONF_REC_MGNT

Requires firmware version 4.10 or higher.

2.48.1 General description

This command enables you to get and edit the recording type settings.

Tag code: 0x0a89

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	T_OCTET	0x01	T_OCTET	0x01

2.48.2 Payload options

Values	Recording types
0	Local
1	VRM
2	VRM+ANR
3	Only local (obsolete)
4	DUAL VRM

Note:
It is not possible to change settings while recording.

Local recording is only possible at devices with integrated storage, therefore check device capabilities first.

2.48.3 RCP+ over CGI example

Read request

In the example below, returns the current recording mode, in this case VRM recording.

```
http://<Device IP>/rcp.xml?command=0x0a89&type=T_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0a89</hex>
      <dec>2697</dec>
   </command>
   <type>T_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
```

```
   <idstring/>
   <payload/>
   <cltid>0x0385</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x01</hex>
      <dec>1</dec>
   </result>
</rcp>
```

Write request

In the example below, the current recording mode is changed to VRM+ANR.

```
http://<Device IP>/rcp.xml?command=0x0a89
&type=T_OCTET&direction=WRITE&payload=0x2
```

2.48.4 RCP+ SDK example

Read request

In the example below, returns the current recording mode.

```
client.sendRequest(0, 0x0a890130, response);
response.wait();
recordingMode= response.readOctet();
```

Write request

In the example below, the current recording mode is changed to VRM+ANR.

```
payload.writeOctet(0x2);
RcpInputStream response;
client.sendRequest(0, 0x0a890131, payload, response);
response.wait();
```

2.49 CONF_RELAY_OUTPUT_STATE

Requires firmware version 2.52 or higher.

2.49.1 General description

This command enables you to set and retrieve the status of the device alarm output (relay).

Note:
This command is also sent as a message.

Tag code: 0x01c1

Numeric descriptor: Video line

Direction	Read request		Write request	
Access level	No protection		User	
Data type	F_FLAG	0x00	F_FLAG	0x00

2.49.2 Payload options

0 = Relay logical value is off

1 = Relay logical value is on

2.49.3 RCP+ over CGI example

Read request

In the example below, the logical value of the first relay is retrieved.

```
http://<Device IP>/rcp.xml?command=0x01c1
&type=F_FLAG&direction=READ&num=1
```

Write request

In the example below, the logical value of the second relay is set to 1.

```
http://<Device IP>/rcp.xml?command=0x01c1
&type=F_FLAG&direction=WRITE&num=2&payload=1
```

Event message

In the example below, status changes are retrieved as a message.

```
http://<Device IP>/rcp.xml?message=0x01c1&collectms=5000
```

2.49.4 RCP+ over CGI example

Read request

```
static bool getOutputState(RcpClient m_RcpClient, byte OutputNumber)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
```

```
   m_RcpClient.setNum(OutputNumber);
   m_RcpClient.sendRequest(2000, (OpCode)0x01c10030, m_RcpInputStream);
   m_RcpInputStream.wait();
   bool state = m_RcpInputStream.readOctet();
   Console.WriteLine("Output state is " + state);
   return state;
}
```

Write request

```
static bool setOutputState(RcpClient m_RcpClient, byte OutputNumber,
bool On)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   RcpOutputStream m_RcpOutputStream = new RcpOutputStream();
   m_RcpOutputStream.writeOctet(On);
   m_RcpClient.setNum(OutputNumber);
   m_RcpClient.sendRequest(2000, (OpCode)0x01c10031, m_RcpOutputStream,
   m_RcpInputStream);
   m_RcpInputStream.wait();
   bool state = m_RcpInputStream.readOctet();
   Console.WriteLine("Output state is " + state);
   return state;
}
```

Event message

```
//Register to messages
   OpCode[] p = new OpCode[1];
   p[0] = unchecked((OpCode)0x01c10030);
   m_RcpClient.subscribeMessageCallback(MessageCallback, m_RcpClient, p);

//Receive the messages
   static private void MessageCallback(Object obj, RcpHeader hdr,
   RcpInputStream ris)
   {
      switch (hdr.opCode)
      {
         case (OpCode)0x01c10030: //Output state
         Console.WriteLine("Output state is " + ris.readOctet());
      }
   }
```

2.50 CONF_ROI

Requires firmware version 4.21 or higher.

2.50.1 General description

This command enables you to retrieve and set region of interest properties such as position and size explicitly. This properties are can be retrieved/are set implicitly with the PTZ commands.

Note:

This setting needs a session ID.

Tag code: 0x0b48

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		Live	
Data type	P_OCTET	0x0c	P_OCTET	0x0c

2.50.2 Payload options

Payload structure

Horizontal position 2 Bytes	Vertical position 2 Bytes
Size 2 Bytes	Reserved 2 Bytes

Horizontal position

0 ... 32768, where 0 is left and 32768 is right. The anchor point is the middle of the ROI, so the minimum coordinates are size/2 and maximum 32768 – size/2.

Vertical position

0 ... 32768, where 0 is up and 32768 is down.

Size

Size of the cropping window in relative coordinates (0 ... 32768). The aspect ratio will be preserved.

2.50.3 RCP+ over CGI example

Read request

In the example below, the ROI settings are retrieved for the video connection with the session ID 0xc216002d.

```
http://<Device IP>/rcp.xml?command=0x0b48
&type=P_OCTET&direction=READ&sessionid=0xc216002d
```

Write request

In the example below, the ROI is set to the quarter of the original video, pointing to the middle of the picture.

```
http://<Device IP>/rcp.xml?command=0x0b48
&type=P_OCTET&direction=WRITE&sessionid=0xc216002d
&payload=0x4000400040000000
```

2.50.4 RCP+ SDK example

Read request

In the example below, the ROI settings are retrieved for the video connection with the session ID 0xc216002d.

```
static void getROI(RcpClient m_RcpClient, uint sessionID)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   m_RcpClient.setSessionId(sessionID);
   m_RcpClient.sendRequest(2000, (OpCode)0x0b480c30, m_RcpInputStream);
   m_RcpInputStream.wait();
   ushort xPos = m_RcpInputStream.readUShort();
   ushort yPos = m_RcpInputStream.readUShort();
   ushort Size = m_RcpInputStream.readUShort();
   Console.WriteLine("X/Y Pos: ("+xPos+"/"+yPos+"), Size: "+Size);
}
```

Write request

In the example below, the ROI is set to the quarter of the original video, pointing to the middle of the picture.

```
static void setROI(RcpClient m_RcpClient, uint sessionID,
ushort xPos, ushort yPos, ushort Size)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   RcpOutputStream m_RcpOutputStream = new RcpOutputStream();
   m_RcpOutputStream.writeUShort(xPos);
   m_RcpOutputStream.writeUShort(yPos);
   m_RcpOutputStream.writeUShort(Size);
   m_RcpOutputStream.writeUShort(0);
   m_RcpClient.setSessionId(sessionID);
   m_RcpClient.sendRequest(2000, (OpCode)0x0b480c31, m_RcpOutputStream,
   m_RcpInputStream);
   m_RcpInputStream.wait();
   ushort xPosNew = m_RcpInputStream.readUShort();
   ushort yPosNew = m_RcpInputStream.readUShort();
   ushort SizeNew = m_RcpInputStream.readUShort();
   Console.WriteLine("X/Y Pos: (" + xPosNew + "/" + yPosNew + "),
   Size: " + SizeNew);
}
```

2.51 CONF_SOFTWARE_VERSION

Requires firmware version 2.52 or higher.

2.51.1 General description

This command enables you to retrieve the 8 digit software version of the device.

Tag code: 0x002f

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	P_STRING	0x10	

2.51.2 Payload options

Payload structure

The first two bits shows the build number, the next two the OEM ID (00 and 50 is reserved for Bosch) and the last four bits the Major version number.

2.51.3 RCP+ over CGI example

Read request

In the example below, the software version retrieved. The payload <str>39500570</str> is read as Firmware 5.70.0039

```
http://<Device IP>/rcp.xml?command=0x002f&type=P_STRING&direction=READ
```

```
<rcp>
   <command>
      <hex>0x002f</hex>
      <dec>47</dec>
   </command>
   <type>P_STRING</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x0095</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <str>39500570</str>
   </result>
</rcp>
```

2.51.4 RCP+ over CGI example

Read request

In the example below, the software version retrieved.

```
char SWVersion[8];
client.sendRequest(0, 0x002f1030, response);
response.wait();
response.readString(SWVersion, response.available());
```

2.52 CONF_STAMP_ATTR_ALARM

Requires firmware version 3.00 or higher.

2.52.1 General description

This command enables you to retrieve and set the current position of the alarm stamping. You may define the exact position of the alarm stamping by setting the x position with the first byte of the payload and the y position with the second byte of the payload.

Note:

The use of the **CONF_STAMP_ATTR_ALARM** command demands the prior use of the command **CONF_ALARM_DISP_VAL** with payload value 3 (see page 13).

Tag code: 0x0938

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	P_OCTET	0x0c	P_OCTET	0x0c

2.52.2 Payload options

Payload structure

X position 1 Byte	Y position 1 Byte	Reserved 10 Bytes

Values

The position range can be set from 0 to 255.

2.52.3 RCP+ over CGI example

Read request

In the example below, the custom alarm stamp position is checked. In this case it is x = y = 0.

```
http://<Device IP>/rcp.xml?command=0x0938&type=P_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0938</hex>
      <dec>2360</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
```

```
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>12</len>
      <dec>00 00 00 00 00 00 00 00 00 00 00 00</dec>
   </result>
</rcp>
```

Write request

In the example below, the alarm stamp position is set to x = y = 1.

```
http://<Device IP>/rcp.xml?command=0x0938
&type=P_OCTET&direction=WRITE&payload=010100000000000000000000
```

2.52.4 RCP+ SDK example

Read request

In the example below, the custom alarm stamp position is checked.

```
client.sendRequest(0, 0x09380C30, response);
response.wait();
x-pos = response.readOctet();
y-pos = response.readOctet();
```

Write request

In the example below, the alarm stamp position is set.

```
payload.writeOctet(x-pos);
payload.writeOctet(y-pos);
payload.write(zeroBuffer, 10)
RcpInputStream response;
client.sendRequest(0, 0x09380C31, payload, response);
```

2.53 CONF_STAMP_ATTR_NAME

Requires firmware version 3.00 or higher.

2.53.1 General description

This command enables you to retrieve the current position of the camera name stamping. You may define the exact position of the camera name stamping by setting the x position with the first byte of the payload and the y position with the second byte of the payload.

Note:

The use of the **CONF_STAMP_ATTR_NAME** command demands the prior use of the command **CONF_NAME_STAMP_VAL** with payload value 3 (see page 110).

Tag code: 0x0936

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	P_OCTET	0x0c	P_OCTET	0x0c

2.53.2 Payload options

Payload structure

X position 1 Byte	Y position 1 Byte	Reserved 10 Bytes

Values

The position range can be set from 0 to 255.

2.53.3 RCP+ over CGI example

Read request

In the example below, the camera name stamp position is checked. In this case it is x = 5 and y = 245.

```
http://<Device IP>/rcp.xml?command=0x0936&type=P_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0936</hex>
      <dec>2358</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
```

```
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>12</len>
      <dec>05 f5 00 00 00 00 00 00 00 00 00 00</dec>
   </result>
</rcp>
```

Write request

In the example below, the camera name stamp position is set to x = y = 5.

```
http://<Device IP>/rcp.xml?command=0x0936
&type=P_OCTET&direction=WRITE&payload=050500000000000000000000
```

2.53.4 RCP+ SDK example

Read request

In the example below, the camera name stamp position is checked.

```
client.sendRequest(0, 0x09360C30, response);
response.wait();
x-pos = response.readOctet();
y-pos = response.readOctet();
```

Write request

In the example below, the camera name stamp position is set.

```
payload.writeOctet(x-pos);
payload.writeOctet(y-pos);
payload.write(zeroBuffer, 10)
RcpInputStream response;
client.sendRequest(0, 0x09360C31, payload, response);
```

2.54 CONF_STORAGE_MEDIUM_AVAIL

Requires firmware version 2.52 or higher.

2.54.1 General description

This command enables you to get the available integrated hardware storage medium types of the device.

Tag code: 0x09d4

Numeric descriptor: Not used

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.54.2 Payload options

The command returns a list of D_WORDS (4 Bytes) with available storage medium types.

Medium types

Values	Meaning
0x00 00 00 00	None
0x00 00 00 05	USB
0x00 00 00 08	IDE
0x00 00 00 09	CF
0x00 00 00 0a	IMG File
0x00 00 00 0b	SD

2.54.3 RCP+ over CGI example

Read request

The example below returns the storage medium types. In this case it is the value **None** and **SD**.

```
http://<Device IP>/rcp.xml?command=0x09d4&type=P_OCTET&direction=READ
```

```
<rcp>
   <command>
      <hex>0x09d4</hex>
      <dec>2516</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
```

```
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>8</hex>
      <str>00 00 00 00 00 00 00 0b</dec>
   </result>
</rcp>
```

2.54.4 RCP+ SDK example

Read request

The example below returns the storage medium types.

```
RcpInputStream response;
client.sendRequest(0, 0x0ae90130, response);
response.wait();
while (response.available())
{
   mediumType[i++] = response.readULong();
}
```

2.55 CONF_SUBNET_STR

Requires firmware version 2.53 or higher.

2.55.1 General description

This command enable you to get and set the Subnet mask of the device.

Tag code: 0x007d

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	P_STRING	0x10	P_STRING	0x10

2.55.2 Payload options

The payload contains the Subnet mask (XXX.XXX.XXX.XXX).

2.55.3 RCP+ over CGI example

Read request

In the example below, the Subnet mask 255.255.0.0 is received.

```
http://<Device IP>/rcp.xml?command=0x007d&type=P_STRING&direction=READ
```

```
<rcp>
   <command>
      <hex>0x007d</hex>
</command>
   <type>P_STRING</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x0153</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <str>255.255.0.0</str>
   </result>
</rcp>
```

Write request

Set the Subnet mask 255.255.255.0 on the device.

```
http://<Device IP>/rcp.xml?command=0x007d
&type=P_STRING&direction=WRITE&payload=255.255.255.0
```

2.55.4 RCP+ SDK example

Read request

In the example below, the Subnet mask is received.

```
RcpInputStream response;
client.sendRequest(0, 0x007d1030, response);
response.wait();
response.readString(subnetMask, response.available());
```

Write request

Set the Subnet mask on the device.

```
RcpOutputStream payload;
payload.writeString(subnetMask);
RcpInputStream response;
client.sendRequest(0, 0x007d1031, payload, response);
```

2.56 CONF_UNIT_NAME

Requires firmware version 2.53 or higher.

2.56.1 General description

This command enable you to get and set the unit name of the device.

Tag code: 0x0024

Numeric descriptor: Not used

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	P_UNICODE	0x14	P_UNICODE	0x14

2.56.2 Payload options

It contains the unit name of the device as an unicode string with the maximum length of 32 characters.

2.56.3 RCP+ over CGI example

Read request

In the example below, the unit name is received. In this case it is **Office Camera**.

```
http://<Device IP>/rcp.xml?command=0x0024&type=P_UNICODE&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0024</hex>
      <dec>36</dec>
   </command>
   <type>P_UNICODE</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x0282</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>28</len>
      <str> 00 4f 00 66 00 66 00 69  00 63 00 65 00 20 00 43  00 61 00 6d 00
65 00 72  00 61 00 00 </str>
   </result>
</rcp>
```

Write request

In the example below, the unit name is changed to **Tower Camera**.

```
http://<Device IP>/rcp.xml?command=0x0024
&type=P_UNICODE&direction=WRITE
&payload=0x0054006f00770065007200200043006100 6d0065007200610000
```

2.56.4 RCP+ SDK example

Read request

In the example below, the unit name is received.

```
RcpInputStream response;
Wchar_t UnitName[32];
client.sendRequest(0, 0x00241430, response);
response.wait();
response.readWString(UnitName, response.available());
```

Write request

In the example below, the unit name is changed.

```
payload.writeWString(UnitName);
RcpInputStream response;
client.sendRequest(0, 0x00241431, payload, response);
response.wait();
```

2.57 CONF_VID_IN_BRIGHTNESS

Requires firmware version 5.70 or higher.

2.57.1 General description

This command enables you to get and set the brightness value of the camera.

Tag code: 0x092a

Numeric descriptor: Video line

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	T_OCTET	0x01	T_OCTET	0x01

2.57.2 Payload options

The value of the brightness is between 0 and 255, it is returned in hexadecimal and decimal.

2.57.3 RCP+ over CGI example

Read request

In the example below, the brightness value for the first video line is retrieved. In this case brightness is set to 128.

```
http://<Device IP>/rcp.xml?command=0x092a
&type=T_OCTET&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x092a</hex>
      <dec>2345</dec>
   </command>
   <type>T_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x0019</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x80</hex>
      <dec></dec>
   </result>
</rcp>
```

Write request

In the example below, the brightness value for the first video line is set to 10.

```
http://<Device IP>/rcp.xml?command=0x092a
&type=T_OCTET&num=1&direction=WRITE&payload=0x000A
```

2.57.4 RCP+ SDK example

Read request

In the example below, the brightness value for the first video line is retrieved.

```
unsigned char brightness;
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x092a0130, response);
response.wait();
brightness = response.readOctet();
```

Write request

In the example below, the brightness value for the first video line is set.

```
RcpOutputStream payload;
payload.writeOctet(brightness);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x092a0131, payload, response);
response.wait();
```

2.58 CONF_VID_IN_CONTRAST

Requires firmware version 5.70 or higher.

2.58.1 General description

This command enables you to get and set the contrast value of the camera.

Tag code: 0x092b

Numeric descriptor: Video line

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	T_OCTET	0x01	T_OCTET	0x01

Payload options

The value of the contrast is between 0 and 255, it is returned in hexadecimal and decimal.

2.58.2 RCP+ over CGI example

Read request

In the example below, the contrast value for the first video line is retrieved. In this case the contrast is set to 128.

```
http://<Device IP>/rcp.xml?command=0x092b
&type=T_OCTET&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x092b</hex>
      <dec>2347</dec>
   </command>
   <type>T_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x0037</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x80</hex>
      <dec>128</dec>
   </result>
</rcp>
```

Write request

In the example below, the contrast value for the first video line is set to 10.

```
http://<Device IP>/rcp.xml?command=0x092b
&type=T_OCTET&num=1&direction=WRITE&payload=0x0A
```

2.58.3 RCP+ SDK example

Read request

In the example below, the contrast value for the first video line is retrieved.

```
unsigned char contrast;
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x092b0130, response);
response.wait();
contrast = response.readOctet();
```

Write request

In the example below, the contrast value for the first video line is set.

```
RcpOutputStream payload;
payload.writeOctet(contrast);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x092b0131, payload, response);
response.wait();
```

2.59 CONF_VID_IN_SATURATION

Requires firmware version 4.50 or higher.

2.59.1 General description

This command enables you to get and set the saturation value of the camera.

Tag code: 0x092c

Numeric descriptor: Video line

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	T_OCTET	0x01	T_OCTET	0x01

2.59.2 Payload options

The value of the saturation is between 0 and 255, it is returned in hexadecimal and decimal.

2.59.3 RCP+ over CGI example

Read request

In the example below, the saturation value for the first video line is retrieved. In this case the saturation is set to 128.

```
http://<Device IP>/rcp.xml?command=0x092c
&type=T_OCTET&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x092c</hex>
      <dec>2348</dec>
   </command>
   <type>T_OCTET</type>
   <direction>READ</direction>
   <num>0</num>
   <idstring/>
   <payload/>
   <cltid>0x0265</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x80</hex>
      <dec>128</dec>
   </result>
</rcp>
```

Write request

In the example below, the saturation value for the first video line is set to 10.

```
http://<Device IP>/rcp.xml?command=0x092c
&type=T_OCTET&num=1&direction=WRITE&payload=0x00A
```

2.59.4 RCP+ SDK example

Read request

In the example below, the saturation value for the first video line is retrieved.

```
unsigned char saturation;
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x092c0130, response);
response.wait();
saturation = response.readOctet();
```

Write request

In the example below, the saturation value for the first video line is set.

```
RcpOutputStream payload;
payload.writeOctet(saturation);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x092c0131, payload, response);
response.wait();
```

2.60 CONF_VIDEO_ALARM_STATE

Requires firmware version 2.52 or higher.

2.60.1 General description

This command enables you to check the current video connection. It applies to encoders, where the video input is connected to an analog camera. In case this connection is broken, the video alarm is raised.

Note:

This command is also sent as a message.

Tag code: 0x01c2

Numeric descriptor: Video line

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	F_FLAG	0x00	

2.60.2 Payload options

0 = Video alarm is off, which means video signal is available

1 = Video alarm is on, no video signal detected at the video input

2.60.3 RCP+ over CGI example

Read request

In the example below, the video alarm status on the second video input is retrieved.

```
http://<Device IP>/rcp.xml?command=0x01c2&type=F_FLAG&direction=READ&num=2
```

Message

In the example below, status changes are retrieved as a message.

```
http://<Device IP>/rcp.xml?message=0x01c2&collectms=5000
```

2.60.4 RCP+ SDK example

Read request

In the example below, the video alarm status on the second video input is retrieved.

```
static bool getVideoAlarmState(RcpClient m_RcpClient, byte VideoLine)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   m_RcpClient.setNum(VideoLine);
   m_RcpClient.sendRequest(2000, (OpCode)0x01c20030, m_RcpInputStream);
   m_RcpInputStream.wait();
```

```
   bool state = m_RcpInputStream.readFlag();
   Console.WriteLine("Video Alarm state is " + state);
   return state;
}
```

Message

In the example below, status changes are retrieved as a message.

```
//Register to messages
OpCode[] p = new OpCode[1];
p[0] = unchecked((OpCode)0x01c20030);
m_RcpClient.subscribeMessageCallback(MessageCallback, m_RcpClient, p);

//Receive the messages
static private void MessageCallback(Object obj, RcpHeader hdr,
RcpInputStream ris)
{
   switch (hdr.opCode)
   {
      case (OpCode)0x01c20030: //Video Alarm state
         Console.WriteLine("Video Alarm state is " + ris.readFlag());
   }
}
```

2.61 CONF_VIDEO_H264_ENC_BASE_OPERATION_MODE

Requires firmware version 4.50 or higher.

2.61.1 General description

This command enables you to retrieve and set the base operation mode, consisting of resolution and corresponding frame rate of the h.264 encoders per video input, of the video over IP device.

Tag code: 0x0ad3

Numeric descriptor: Video line

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	P_OCTET	0x0c	P_OCTET	0x0c

2.61.2 Payload options

Payload structure

Stream 1 mode 2 Bytes	Stream 2 mode 2 Bytes

Get or set the base operation mode of the H.264 encoders per line. The first DWORD is stream 1 and the second stream 2.

Note:

To get the possible modes for setting see command **CONF_VIDEO_H264_ENC_BASE_OPERATION_MODE_CAPS**.

Encoder mode

Values	Types
0	Copy other stream
1	Compatibility mode to H.264 BP + (bit rate limited)
3	H264 MP SD
4	H264 MP fixed 720p
5	H264 MP fixed 720p full framerate
6	H264 MP fixed 1080p
7	H264 MP fixed 720p skip=3
8	H264 MP fixed 720p skip=4
9	H264 MP fixed 1080p skip=7
10	H264 MP SD ROI PTZ
11	H264 MP HD 2592 × 1944

Values	Types
12	H264 MP SD upright format (cropped)
13	H264 MP SD 4CIF resolution 4:3 format (cropped)
14	H264 MP SD (max. 288p) dual stream with independent ROI PTZ

2.61.3 RCP+ over CGI example

Read request

In the example below, the base operation mode of the encoder for stream 1 and 2 on video line 1 is retrieved. In this case, stream 1 is set to **H264 MP fixed 1080p** and stream 2 to **H264 MP fixed 720p skip=3**.

```
http://<Device IP>/rcp.xml?command=0x0ad3
&type=P_OCTET&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0ad3</hex>
      <dec>2771</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>8</len>
      <str>00 00 00 06 00 00 00 07 </str>
   </result>
</rcp>
```

Write request

In the example below, the base operation mode for video line 1 is set. For stream 1 to **H264 MP fixed 720p** and stream 2 to **copy stream**.

```
http://<Device IP>/rcp.xml?command=0x0ad3
&type=P_OCTET&num=1&direction=WRITE&payload=0x0000000400000000
```

2.61.4 RCP+ SDK example

Read request

In the example below, the base operation mode of the encoder for stream 1 and 2 is retrieved.

```
client.setNum(1);
RcpInputStream response;
```

```
client.sendRequest(0, 0x0ad30C30, response);
response.wait();
operationMode[0] = response.readULong();
operationMode[1] = response.readULong();
```

Write request

In the example below, the base operation mode is set for stream 1 to **H264 MP fixed 720p** and stream 2 to **copy stream**.

```
payload.writeULong(operationMode[0]);
payload.writeULong(operationMode[1]);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x0ad30C31, payload, response);
response.wait();
```

2.62 CONF_VIDEO_H264_ENC_BASE_OPERATION_MODE_CAPS

Requires firmware version 4.50 or higher.

2.62.1 General description

This command enables you to get the base operation mode list of the H.264 encoder.

Tag code: 0x0af9

Numeric descriptor: Video line

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.62.2 Payload options

Payload structure

Number of streams (S) 2 Bytes	Encoder mode pair 1 S × 2 Bytes
S × N × 2 Bytes	Encoder mode pair N S × 2 Bytes

Get the base operation mode capability list of the H.264 encoders per line. The list contains all possible mode combinations for all streams. The first DWORD is the number of streams. Then the combinations follows paired in n DWORDs.

Number of streams

Number of streams which are paired in the encoder mode.

Encoder mode

Values	Types
0	Copy other stream
1	Compatibility mode to H264 BP + (bit rate limited)
3	H264 MP SD
4	H264 MP fixed 720p
5	H264 MP fixed 720p full framerate
6	H264 MP fixed 1080p
7	H264 MP fixed 720p skip=3
8	H264 MP fixed 720p skip=4
9	H264 MP fixed 1080p skip=7
10	H264 MP SD ROI PTZ
11	H264 MP HD 2592 × 1944
12	H264 MP SD upright format (cropped)

Values	Types
13	H264 MP SD 4CIF resolution 4:3 format (cropped)
14	H264 MP SD (max 288p) dual stream with independent ROI PTZ

2.62.3 RCP+ over CGI example

Read request

In the example below, the base operation mode list is retrieved for the first video input. In this case, the number of streams is 2, with the following combinations.

```
http://<Device IP>/rcp.xml?command=0x0af9
&type=P_OCTET&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0af9</hex>
      <dec>2809</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>124</len>
      <str>00 00 00 02 00 00 00 04 00 00 00 04 00 00 00 04 00 00 00 03 00 00
00 04 00 00 00 0a 00 00 00 04 00 00 00 0c 00 00 00 04 00 00 00 0d 00 00 00
04 00 00 00 0e 00 00 00 03 00 00 00 03 00 00 00 06 00 00 00 03 00 00 00 06
00 00 00 0a 00 00 00 06 00 00 00 07 00 00 00 06 00 00 00 09 00 00 00 06 00
00 00 00 00 00 00 06 00 00 00 0c 00 00 00 06 00 00 00 0d 00 00 00 06 00 00
00 0e </str>
   </result>
</rcp>
```

According to the encoder mode **enum**, the response can be read as following:

```
Stream 1              stream 2
H264 MP fixed 720p    H264 MP fixed 720p
H264 MP fixed 720p    H264 MP SD
H264 MP fixed 720p    H264 MP SD ROI PTZ
H264 MP fixed 720p    H264 MP SD upright format (cropped)
H264 MP fixed 720p    H264 MP SD 4CIF resolution 4:3 format (cropped)
H264 MP fixed 720p    H264 MP SD (max 288p) dual stream with independent ROI PTZ
H264 MP SD            H264 MP SD
H264 MP fixed 1080p   H264 MP SD
H264 MP fixed 1080p   H264 MP SD ROI PTZ
```

```
H264 MP fixed 1080p   H264 MP fixed 720p skip=3
H264 MP fixed 1080p   H264 MP fixed 1080p skip=7
H264 MP fixed 1080p   Copy other stream
H264 MP fixed 1080p   H264 MP SD upright format (cropped)
H264 MP fixed 1080p   H264 MP SD 4CIF resolution 4:3 format (cropped)
H264 MP fixed 1080p   H264 MP SD (max 288p) dual stream with independent ROI PTZ
```

2.62.4 RCP+ SDK example

Read request

In the example below, the base operation mode list is retrieved for the first video input.

```
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x0af90C30, response);
response.wait();
NbrOfStreams = response.readULong();
while (ris.available())
{
   streamCombination++;
   for (int i=0; i<NbrOfStreams, i++)
   {
      operationMode[streamCombination][i] = response.readULong();
   }
}
```

2.63 CONF_VIDEO_INPUT_FORMAT_EX

Requires firmware version 5.50 or higher.

2.63.1 General description

This command enables you to get and set the video input format.

Tag code: 0x0b10

Numeric descriptor: Video line

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	P_OCTET	0x0c	P_OCTET	0x0c

2.63.2 Payload options

Payload structure

Video mode 1 Byte	Video format 2 Bytes	Reserved 18 Bytes

Video mode

Values	Modes
0	fixed
1	auto-detect

Video format

Values	Types
0	No
1	PAL
2	NTSC
3	VGA
4	720P
5	1080P
6	QVGA
7	720P25
8	720P30
9	720P50
10	720P60
11	1080P25
12	1080P30
13	2592x1944P12

Values	Types
14	1440x1080P25

2.63.3 RCP+ over CGI example

In the example below, the video input format is retrieved for video line one. In this case it is **1080P30**.

```
http://<Device IP>/rcp.xml?command=0x0b10
&type=P_OCTET&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0b10</hex>
      <dec>2832</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>20</len>
      <str>00 0c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 </str>
   </result>
</rcp>
```

Write request

In the example below, the video input format on line 1 is set to **1080P25**.

```
http://<Device IP>/rcp.xml?command=0x0b10
&type=P_OCTET&num=1&direction=WRITE&payload=0x000b
```

2.63.4 RCP+ SDK example

In the example below, the video input format is retrieved for video line 1.

```
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x0b100C30, response);
response.wait();
videoMode = response.readOctet();
videoFormat = response.readUShort();
```

Write request

In the example below, the video input format on line 1 is set to **1080P25**.

```
payload.writeOctet(videoMode);
payload.writeUShort(videoFormat);
payload.write(zeroBuffer, 18);
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x0b100C31, payload, response);
response.wait();
```

2.64 CONF_VIDEO_INPUT_FORMAT_EX_OPTIONS

Requires firmware version 5.80 or higher.

2.64.1 General description

This command enables you to get the supported video formats.

Tag code: 0x0b9b

Numeric descriptor: Video line

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.64.2 Payload options

Payload structure

The response offers a list of supported video formats, which can be selected with the command **CONF_VIDEO_INPUT_FORMAT_EX**.

No of supported video formats 1 Byte	Video format 1 1 Byte	... N × 1 Byte	Video format N 1 Bytes

2.64.3 RCP+ over CGI example

Read request

In the example below, the supported video formats are retrieved. In this case **NTSC**, **1080P25** and **1080P30**.

```
http://<Device IP>/rcp.xml?command=0x0b9b
&type=P_OCTET&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0b9b</hex>
      <dec>2971</dec>
   </command>
   <type>P_OCTET</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <len>3</len>
      <str>02 0c 0b </str>
```

```
   </result>
</rcp>
```

2.64.4 RCP+ SDK example

Read request

In the example below, the supported video formats are retrieved.

```
client.setNum(1);
RcpInputStream response;
client.sendRequest(0, 0x0b9b0C30, response);
response.wait();
nbrOfVideoFormats = response.readOctet();
for (int i=0; i<nbrOfVideoFormats, i++)
{
   videoFormat[i] = response.readOctet();
}
```

2.65 CONF_VIPROC_ALARM

Requires firmware version 4.10 or higher.

2.65.1 General description

This command enables you to retrieve the state of the Intelligent Video Analysis (IVA).

Note:

This command is also sent as a message.

The message is sent whenever any of the bit values changes. Additionally it is sent once per 10 seconds when any of the bits is set.

Tag code: 0x0807

Numeric descriptor: Video line

Direction	Read request		Write request
Access level	No protection		Not supported
Data type	P_OCTET	0x0c	

2.65.2 Payload options

Payload structure

Alarm flags 2 Bytes	Detector flags 2 Bytes	Config ID 1 Byte

Alarm flags

Values	
0x0080	Invalid configuration
0x0100	Reference image check failed
0x0200	Signal loss/video alarm
0x0400	Image too blurry
0x0800	Signal too noisy
0x1000	Signal too dark
0x2000	Signal too bright
0x4000	Global change
0x8000	Motion

Detector flags

The detector flags are mapped to the IVA rules, where the 0x0001 detector flag corresponds to the first IVA rule, 0x0002 detector flag to second IVA rule, and so on.

Config ID

Identification number of the IVA profile caused this alarm.

2.65.3 RCP+ over CGI example

Read request

In the example below, the IVA status from the first video input is retrieved.

```
http://<Device IP>/rcp.xml?command=0x0807&type=P_OCTET&direction=READ&num=1
```

Message

In the example below, status changes are retrieved as a message.

```
http://<Device IP>/rcp.xml?message=0x0807&collectms=5000
```

2.65.4 RCP+ SDK example

Read request

In the example below, the IVA status from the first video input is retrieved.

```
static void getIVAState(RcpClient m_RcpClient, byte VideoLine)
{
   RcpInputStream m_RcpInputStream = new RcpInputStream();
   m_RcpClient.setNum(VideoLine);
   m_RcpClient.sendRequest(2000, (OpCode)0x08070c30, m_RcpInputStream);
   m_RcpInputStream.wait();
   uint ivaFlags = m_RcpInputStream.readUShort();
   uint detectorField = m_RcpInputStream.readUShort();
   if ((uint)(ivaFlags & 0x100)>0)
      Console.WriteLine("Reference Image check failed");

   if ((uint)(ivaFlags & 0x200) > 0)
      Console.WriteLine("Signal loss");
   if ((uint)(ivaFlags & 0x400) > 0)
      Console.WriteLine("Image too blurry");
   if ((uint)(ivaFlags & 0x800) > 0)
      Console.WriteLine("Signal too noisy");
   if ((uint)(ivaFlags & 0x1000) > 0)
      Console.WriteLine("Signal too dark");
   if ((uint)(ivaFlags & 0x2000) > 0)
      Console.WriteLine("Signal too bright");
   if ((uint)(ivaFlags & 0x4000) > 0)
      Console.WriteLine("Global change");
   if ((uint)(ivaFlags & 0x8000) > 0)
      Console.WriteLine("TaskID: " + detectorField + "is triggered");
```

```
}
```

Message

In the example below, status changes are retrieved as a message.

```
//Register to messages
OpCode[] p = new OpCode[1];
p[0] = unchecked((OpCode)0x01c00030);
m_RcpClient.subscribeMessageCallback(MessageCallback, m_RcpClient, p);

//Receive the messages
static private void MessageCallback(Object obj, RcpHeader hdr,
RcpInputStream ris)
{
   switch (hdr.opCode)
   {
      case (OpCode)0x08070c30: //IVA state
           uint ivaFlags = ris.readUShort();
        uint detectorField = ris.readUShort();
        uint res = ivaFlags & 0x100;
        if ((uint)(ivaFlags & 0x100) > 0)
           Console.WriteLine("Reference Image check failed");
        if ((uint)(ivaFlags & 0x200) > 0)
           Console.WriteLine("Signal loss");
        if ((uint)(ivaFlags & 0x400) > 0)
           Console.WriteLine("Image too blurry");
        if ((uint)(ivaFlags & 0x800) > 0)
           Console.WriteLine("Signal too noisy");
        if ((uint)(ivaFlags & 0x1000) > 0)
           Console.WriteLine("Signal too dark");
        if ((uint)(ivaFlags & 0x2000) > 0)
           Console.WriteLine("Signal too bright");
        if ((uint)(ivaFlags & 0x4000) > 0)
           Console.WriteLine("Global change");
        if ((uint)(ivaFlags & 0x8000) > 0)
           Console.WriteLine("TaskID: " + detectorField+ "is triggered");

        break;
   }
}
```

2.66 CONF_VIRTUAL_ALARM_STATE

Requires firmware version 4.00 or higher.

2.66.1 General description

This command enables you to get and set the virtual alarm state.

Note:
This command is also sent as a message.

Tag code: 0x0a8b

Numeric descriptor: Virtual alarm number

Direction	Read request		Write request	
Access level	No protection		Service	
Data type	F_FLAG	0x00	F_FLAG	0x00

2.66.2 Payload options

Values	
0	Virtual alarm off
1	Virtual alarm on

2.66.3 RCP+ over CGI example

Read request

In the example below, the virtual alarm state of the first virtual alarm is received. In this case, the status is set to **off**.

```
http://<Device IP>/rcp.xml?command=0x0a8b&type=F_FLAG&num=1&direction=READ
```

```
<rcp>
   <command>
      <hex>0x0a8b</hex>
      <dec>2699</dec>
   </command>
   <type>F_FLAG</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x038a</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
```

```
      <hex>0x00</hex>
      <dec>0</dec>
   </result>
</rcp>
```

Write request

In the example below, the virtual alarm state of the first virtual alarm is set to **on**.

```
http://<Device IP>/rcp.xml?command=0x0a8b
&type=F_FLAG&num=1&direction=WRITE&payload=1
```

2.66.4 RCP+ SDK example

Read request

In the example below, the virtual alarm state is received.

```
client.setNum(virtualAlarmNbr);
RcpInputStream response;
client.sendRequest(0, 0x0a8b0030, response);
response.wait();
VirualAlarmState = response.readOctet();
```

Write request

In the example below, the virtual alarm is set.

```
RcpOutputStream payload;
payload.writeOctet(VirualAlarmState);
client.setNum(virtualAlarmNbr);
RcpInputStream response;
client.sendRequest(0, 0x0a8b0031, payload, response);
response.wait();
```

Bosch Sicherheitssysteme GmbH
Robert-Bosch-Ring 5
85630 Grasbrunn
Germany
www.boschsecurity.com
© Bosch Sicherheitssysteme GmbH, 2013