Bosch Video IP

RCP+ over CGI

www.boschsecurity.com

BOSCH

1 Overview

Bosch IP Video products in video management system environments are mainly controlled and managed using an enhanced version of **R**emote **C**ontrol **P**rotocol, RCP+, pronounced »RCP plus«. This protocol defines commands and messages that allow to configure the units and to establish communication between units, or units and management systems.

Since firmware version 3.0, RCP+ commands can be encapsulated into CGI requests. The command structure of RCP+ must be translated into parameters and values handed over to the unit's XML-based CGI interpreter as a query string. Using HTTP as transportation protocol, e.g. with a Web browser, an URL may look like:

```
HTTP://160.10.0.1/rcp.xml?<<query_string>>
```

Each RCP+ command is a fixed predefined 16 bit value followed by a structured set of parameters, some mandatory, some optional. The RCP+ command and the required structure can be taken from the RCP+ documentation. All message elements in the query string must resemble an appropriate pair of parameter and value. The following examples shall give a principle overview of how to use RCP+ commands with CGI.

The following examples assume having installed a Dinion IP camera at IP address 160.10.0.1.

The **<<query_string>>** syntax and valid parameters are described in section 2.

1.1 Reading parameters

Various parameters can be read from a Bosch IP video product, e.g. the hardware or firmware version, the name of the camera(s), or the CPU performance values. To check our Dinion IP's encoder load, part of the values displayed with the performance bar in the web browser, the URL must look like:

```
HTTP://160.10.0.1/rcp.xml?command=0x0a07
&type=T_DWORD&direction=READ&num=1
```

The return value provides a recent snapshot value of the CPU performance required for encoding video. The web browser is able to display the returned XML message in a readable format.

The return value is presented in the **<result>** section and in this example shows a value of **51** %.

The XML return message is displayed in the web browser like this:

```
<rcp>
  <command>
    <hex>0x0a07</hex>
    <dec>2567</dec>
  </command>
  <type>T_DWORD</type>
  <direction>READ</direction>
  <num>1</num>
  <idstring />
  <payload />
  <cltid>0xd2f9</cltid>
  <auth>2</auth>
  <result>
    <hex>0x00000033</hex>
    <dec>51</dec>
  </result>
</rcp>
```

The return message displayed in the web browser also contains unused and optional parameters that are represented by an »empty« XML tag in the form of

```
<xmltag />
```

(e.g. <idstring /> or <payload/> in the example above). These may be disregarded.

To check our Dinion IP's camera name the URL must look like:

```
HTTP://160.10.0.1/rcp.xml?command=0x0019
&type=P_UNICODE&direction=READ&num=1
```

The XML return message is displayed in the web browser like this:

```
<rcp>
   <command>
      <hex>0x0019</hex>
      <dec>25</dec>
   </command>
   <type>P_UNICODE</type>
   <direction>READ</direction>
   <num>1</num>
   <idstring />
   <payload />
   <cltid>0xd82d</cltid>
   <auth>2</auth>
   <result>
      <str>00 43 00 61 00 6d 00 65 00 72 00 61
            00 20 00 31 00 00</str>
   </result>
</rcp>
```

The camera name as return value is coded in Unicode which is not clearly visible to humans but easily interpreted by a computer. The Unicode string in this example resembles the string **Camera 1**.

1.2 Writing parameters

RCP+ over CGI can also be used to change the various parameters of a Bosch IP Video product, e.g. the device name or the relay state. Parameter writing or setting is done in a similar way like reading them. Except the direction tag is set to WRITE instead of READ and a payload has to be specified. To set the state of the first (**num=1**) relay of our Dinion IP to 0 (**payload=0x0**; = off) the URL must look like:

```
http://160.10.0.1/rcp.xml?command=0x01c1
&type=F_FLAG&direction=WRITE&num=1
&payload=0x0
```

The XML return message is displayed in the web browser like this:

```
<rcp>
   <command>
      <hex>0x01c1</hex>
      <dec>449</dec>
   </command>
   <type>F_FLAG</type>
   <direction>WRITE</direction>
   <num>1</num>
   <idstring/>
   <payload/>
   <cltid>0x4d2d</cltid>
   <sessionid>0x00000000</sessionid>
   <auth>2</auth>
   <protocol>TCP</protocol>
   <result>
      <hex>0x00</hex>
      <dec>0</dec>
   </result>
</rcp>
```

2 CGI <<query_string>> reference

As described and shown in section 1, the CGI query string consist of different parameter, value pairs. The parameters and their possible values are described in the following sections. Some of them are mandatory, some of them are optional. First the mandatory parameters are described, followed by the optional ones.

2.1 Mandatory parameters

2.1.1 Parameter »command«

This is the most important part of the CGI request and represents the RCP+ command tag.

Values:

RCP+ command tag value.

2.1.2 Parameter »type«

The payload type for a specific RCP+ command tag.

Values:

F_FLAG

T_OCTET

T_WORD

T_INT

T_DWORD

P_OCTET

P_STRING

P_UNICODE

2.1.3 Parameter »direction«

Defines, whether the call is getting or setting the parameter in question.

Values:

READ

WRITE

2.1.4 Parameter »num«

The numeric descriptor parameter. This parameter is mandatory for certain RCP+ commands.

Values:

The numeric descriptor.

2.1.5 Parameter »payload«

The payload for a specific RCP+ command. The payload structure is depending on the command's payload type as described above.

Values:

The payload as ***readable string*** *for payload type* ***P_STRING***

The payload as ***octet array*** *with no spaces preceded with 0x for payload type* ***P_OCTET*** *and* ***P_UNICODE***

The payload ***value*** *in* ***hex*** *or* ***decimal*** *notion for all* ***other*** *payload types.*

2.2 Optional CGI parameters

2.2.1 Parameter »idstring«

Values:

User defined parameter which will not be processed by the unit; will return unchanged in the reply.

2.2.2 Parameter »sessionid«

Values:

Context specific session ID for this command

2.3 Message handling

RCP+ messages will be processed and received by the CGI client using a poll mechanism.

A CGI command with the requested messages command tag values needs to be issued. After issuing a request, the reply returns immediately after a message has been sent or the default timeout of 1000 ms has been expired. When the timeout expires, a message count of 0 is returned. Otherwise the number of received messages is signaled. The default timeout can be altered by setting the **collectms** CGI value to the appropriate number of milliseconds.

A parameter description can be found below.

2.3.1 Parameter »message«

Values:

One or a list of requested message RCP+ command tag values separated by ***$***

2.3.2 Parameter »collectms«

Values:

Time in milliseconds to collect messages before returning with ***No messages***

Default is 1000 ms.

The **<msgcnt>** section contains the number of valid **<msg>** sections inside the reply.

2.3.3 Message reply

A buffer mechanism (depth 64) ensures that no messages will be lost during two consecutive poll cycles. The **<over>** section represents the number of lost messages. The **<clip>** value that the internal buffers were too small; this should never occur. The **<poll>** section counts the number of issued poll requests.

A sample request to receive all connection related messages (**CONF_CONNECT_TO**) will look like:

```
HTTP://160.10.0.1/rcp.xml?message=0xffcc
&collectms=5000
```

The XML return message is displayed in the web browser like this:

```
<message_list>
   <stats>
      <msgcnt>2</msgcnt>
      <over>0</over>
      <clip>0</clip>
      <poll>1</poll>
   </stats>
   <cltid>0x002a</cltid>
   <msg>
      <no>1</no>
      <command>0xffcc</command>
      <num>0</num>
      <sessionID>0x7063001f</sessionID>
      <hex>0xa00a0034000000000
            101000001010001</hex>
   </msg>
   <msg>
      <no>2</no>
      <command>0xffcc</command>
      <num>0</num>
      <sessionID>0x70630020</sessionID>
      <hex>0xa00a0034000000000
            101000001010001</hex>
   </msg>
</message_list>
```

3 Authentication

The RCP+ server need proper authentication to process protected commands. The CGI interface provides three basic authentication schemes to be used.

3.1 Using http authentication

In this case, HTTP header authentication (basic or digest) must be present in the request. The internal HTTP server will pass the login information to the RCP+ server.

3.2 Using session cookie

For each successful HTTP authentication (for all available resources like html pages, images ...) a session cookie will be returned by the HTTP server. When this cookie is present in any further HTTP connections, the same authorization level will be granted as in the originating connection. The session cookie will remain active as long as at least one HTTP connection remains open. In this case the internal HTTP server will pass the login information to the RCP+ server.

3.3 CGI inline

By passing the CGI parameter **pwd** with a value of the identification string as defined in the CONF_RCP_CLIENT_REGISTRATION section for normal method. This information will be directly passed to the RCP+ server for validation.