Standard Integration Package

Bosch Video IP

BOSCH

en Developer’s Guide

Table of Contents

1 Network

1.1 Network scan

Network scan offers you an easy way to detect Bosch video over IP devices on your network automatically. Devices generate and transmit their replies to the request within two seconds.

Integration methods:

☑ VideoSDK

☐ RCP+ over CGI

☐ RTSP

VideoSDK

Steps and codes	
1	Initiate the device finder.
	`m_deviceFinder = new Bosch.VideoSDK.Device.DeviceFinder();`
2	Add event handlers for the device finder's events.
	`m_deviceFinder.DeviceDetected += new Bosch.VideoSDK.GCALib._IDeviceFinderEvents_DeviceDetectedEventHandler(OnDeviceDetected);` `m_deviceFinder.DeviceRemoved += new Bosch.VideoSDK.GCALib._IDeviceFinderEvents_DeviceRemovedEventHandler(OnDeviceRemoved);`
3	Select the discovery mode.
	`m_deviceFinder.DiscoveryMode = Bosch.VideoSDK.DeviceDiscoveryModeEnum discoveryMode`
4	Start the device scan.
	`m_deviceFinder.StartDetect(0);`
5	Collect the results via event handlers.
	`private void OnDeviceDetected(Bosch.VideoSDK.Device.DeviceInfo deviceInfo) {//do something with the results}` `private void OnDeviceRemoved(Bosch.VideoSDK.Device.DeviceInfo deviceInfo) {//do something with the results}`

Further information

Video SDK Interface Definition.chm:

– Concepts > **Device Detection**

– Class List > **IDeviceFinder**

– Examples > **VSDKDeviceFinder.cs**

Partner Area on ipp.boschsecurity.com:

– Downloads > Code Samples > **IP_03_VSDK**

2 Video

2.1 Video Settings

2.1.1 Change Video Profile

There are two methods to change the video profile on a device, depending on the product group where the device belongs to. All device product groups can be found at

http://ipp.boschsecurty.com > Tools > Bosch firmware-product matrix

To change the streaming behavior of a BVIP device, several aspects need to be taken into account. This simplified explanation covers the most common use cases.

First, the video input format determines the frame rate. It is used to switch between 60 fps and 30 fps format. Next, you select the basic operating mode, which sets the resolution. All other settings like bit rate, reduced fps or expert settings (GOP structure, I-frame distance, or quantizer) are handled by profiles. Once a profile is set to the customer‘s needs, it can then be assigned to a video stream.

Integration methods:

- ☐ VideoSDK
- ☑ RCP+ over CGI
- ☐ RTSP
- ☑ RCP+ SDK

RCP+ over CGI

Change H.264 profile for CPP 3 and CPP 4 devices

Steps and codes	
1	Set the base frame rate with the RCP+ command CONF_VIDEO_INPUT_FORMAT_EX. Supported input formats can be retrieved with the RCP+ command CONF_VIDEO_INPUT_FORMAT_EX_OPTIONS. In the code below, the base frame-rate is set to 60 fps (0x0a = 720P**60**) for the first video input (num=1).
	`http://<Device IP>/rcp.xml?command=0x0b10&type=P_OCTET&direction=WRITE&num=1&payload=0x000a00000000000000000000000000000000000000`
2	Choose a stream 1 and stream 2 mode with the RCP+ command CONF_VIDEO_H264_ENC_BASE_OPERATION_MODE. In the code below, the base operation mode of the H.264 encoder for the first video input is changed. Stream 1 is set to H.264 MP fixed 1080p and stream 2 to 0: copy stream (0x6).
	`http://<Device IP>/rcp.xml?command=0x0ad3&type=P_OCTET&direction=WRITE&num=1&payload=0x0000000600000000`

Steps and codes	
3	Settings like bit rate, reduced fps or expert settings (GOP-structure, I-frame distance or quantizer) are handled by the profiles. Get the current profile for the first stream on video input 1 with the RCP+ command CONF_MPEG4_CURRENT_PARAMS_REL_CODER. The other payload parameters can be ignored on read request.
	`http://<Device IP>/rcp.xml?command=0x061c&type=P_OCTET&direction=READ&payload=0x0101000000000000`
4	Change the target and maximum data rate in the profile with the RCP+ commands CONF_MPEG4_BANDWIDTH_KBPS and CONF_MPEG4_BANDWIDTH_KBPS_SOFT_LIMIT. In the code below, the target bit rate of the selected profile is set. The profile is 1 and the target bit rate is 5000 kbps.
	`http://<Device IP>/rcp.xml?command=0x0607&type=T_DWORD&direction=WRITE&num=1&payload=5000`
	In the code below, the maximum bit rate of the selected profile is set. The profile is 1, the bit rate is 10000 kbps.
	`http://<Device IP>/rcp.xml?command=0x0612&type=T_DWORD&direction=WRITE&num=1&payload=10000`
5	Change the frame rate with the RCP+ command CONF_MPEG4_FRAME_SKIP_RATIO. In example below, the skip ratio is set to 3, which means the frame rate is 10 fps on a device where the max frame rate is 30 fps.
	`http://<Device IP>/rcp.xml?command=0x0606&type=T_DWORD&direction=WRITE&num=1&payload=3`
6	Assign the profile to the encoder with the RCP+ command CONF_MPEG4_CURRENT_PARAMS_REL_CODER. In the code below, the preset 1 is set to the encoder 1.
	`http://<Device IP>/rcp.xml?command=0x061c&type=P_OCTET&direction=WRITE&payload=0x0101000001000000`

Change H.264 profile for CPP ENC devices

Steps and codes	
1	Get the current stream configuration for the first stream on video input 1 with the RCP+ command CONF_MPEG4_CURRENT_PARAMS_REL_CODER. The other payload parameters can be ignored on read request.
	`http://<Device IP>/rcp.xml?command=0x061c&type=P_OCTET&direction=READ&payload=0x0101000000000000`
2	Change the resolution with the RCP+ command CONF_MPEG4_RESOLUTION.
	`http://<Device IP>/rcp.xml?command=0x0608&type=T_DWORD&direction=WRITE&num=1&payload=7`
3	Change the frame rate with the RCP+ command CONF_MPEG4_FRAME_SKIP_RATIO. In the example below, the skip ratio is set to 3, which means the frame rate is 10 fps on a device where the max frame rate is 30 fps.
	`http://<Device IP>/rcp.xml?command=0x0606&type=T_DWORD&direction=WRITE&num=1&payload=3`
4	Change the target and maximum data rate in the profile with the RCP+ commands CONF_MPEG4_BANDWIDTH_KBPS and CONF_MPEG4_BANDWIDTH_KBPS_SOFT_LIMIT. In the code below, the target bit rate of the selected profile is set. The profile is 1 and the target bit rate is 5000 kbps.
	`http://<Device IP>rcp.xml?command=0x0607&type=T_DWORD&direction=WRITE&num=1&payload=5000`
	In the code below, the maximum bit rate of the selected profile is set. The profile is 1, the bit rate is 10000 kbps.
	`http://<Device IP>/rcp.xml?command=0x0612&type=T_DWORD&direction=WRITE&num=1&payload=10000`
5	Assign the profile to the encoder with the RCP+ command CONF_MPEG4_CURRENT_PARAMS_REL_CODER. In the code below, the preset 1 is set to the encoder 1 on line 1.
	`http://<Device IP>/rcp.xml?command=0x061c&type=P_OCTET&direction=WRITE&payload=0x0101000001000000`

Further information

Partner Area on ipp.boschsecurity.com:

– Downloads > Code Samples > **IP_05_CGI**

Change JPEG profile for CPP 3 and CPP 4 devices

Steps and codes	
1	On CPP3 and CPP4 devices, the JPEG stream settings such as resolution, frame rate and quality are adjusted with the RCP+ command CONF_JPEG_STREAM_SETUP. In the code below, the resolution is set to 720p (0x12) with full 30 fps (0x7530) and medium quality settings (0x32).
	`http://<Device IP>/rcp.xml?command=0x0ad5 &type=T_OCTET&direction=WRITE&payload=0x000000120000753000000032`

Change JPEG profile for CPP ENC devices

Steps and codes	
1	On CPP-ENC devices, the JPEG settings are adjusted with video profiles. Get the current profile for the third stream, which is the JPEG stream, on video input 1. The other payload parameters can be ignored on read request.
	`http://<Device IP>/rcp.xml?command=0x061c &type=P_OCTET&direction=READ&payload=0x0103000000000000`
2	Change the resolution with the RCP+ command CONF_MPEG4_RESOLUTION. In the example below, VGA resolution (7) is assigned for profile 5.
	`http://<Device IP>/rcp.xml?command=0x0608 &type=T_DWORD&direction=WRITE&num=5&payload=7`
3	Change the frame rate with the RCP+ command CONF_MPEG4_FRAME_SKIP_RATIO. In the example below, the frame rate is set to 10 fps (the device base frame rate is 30 fps) with frame skip value 3 on the 5th profile.
	`http://<Device IP>/rcp.xml?command=0x0606 &type=T_DWORD&direction=WRITE&num=5&payload=3`
4	Change the JPEG bit rate with the RCP+ command CONF_JPEG_BANDWIDTH_KBPS for the 5th video profile to the value 1500.
	`http://<Device IP>/rcp.xml?command=0x061d &type=T_DWORD&direction=WRITE&num=5&payload=1500`
5	Assign the 5th profile to the JPEG encoder.
	`http://<Device IP>/rcp.xml?command=0x061c1 &type=P_OCTET&direction=WRITE&payload=0x0103000005000000`

2.1.2 Camera stamping

This section describes how to enable/disable the camera and alarm stamping video overlays in the device.

Integration methods:

☐ VideoSDK

☑ RCP+ over CGI

☐ RTSP

RCP+ over CGI

Steps and codes	
1	The camera name stamping can be changed with the RCP+ command CONF_NAME_STAMP_VAL. The following options are available for the camera name stamping: 0 = name stamping off 1 = name stamping on bottom 2 = name stamping on top 3 = name stamping with custom attributes In the code below, the camera stamping is set to custom stamping position.
	`http://<Device IP>/rcp.xml?command=0x0084 &type=T_OCTET&direction=WRITE&protocol=TCP&payload=3&num=1`
2	Optional: Only necessary if option 3 (name stamping with custom attributes) is chosen. The position can be set with the RCP+ command CONF_STAMP_ATTR_NAME. First byte sets the x position (0 – 255) and the second byte sets the y position (0 – 255). The following 10 bytes are reserved and not used right now. In the code below, the camera stamping is set to left (x = 5) bottom (y = 245) corner.
	`http://<Device IP>/rcp.xml?command=0x0936 &type=P_OCTET&direction=WRITE&protocol=TCP &payload=0x05F500000000000000000000&num=1`
3	The alarm text stamping can be changed with the RCP+ command CONF_ALARM_DISP_VAL. The following options are available for the alarm stamping: 1 = alarm display off 2 = alarm display on 3 = alarm display with custom attributes In the code below, the alarm stamping is set to custom stamping position.
	`http://<Device IP>/rcp.xml?command=0x008e &type=T_OCTET&direction=WRITE&num=1&payload=3`

Steps and codes	
4	Optional: Only necessary if option 3 (alarm display with custom attributes) is chosen. The position can be set with the RCP+ command CONF_STAMP_ATTR_ALARM. First byte sets the x position (0 – 255) and the second byte sets the y position (0 – 255). The following 10 bytes are reserved and not used right now. In the code below, the alarm stamping is set to center (x = 120) top (y = 10) position.
	`http://<Device IP>/rcp.xml?command=0x0938&type=P_OCTET&direction=WRITE&protocol=TCP&payload=0x780A00000000000000000000&num=1`

2.2 Video Streaming

2.2.1 Display Video

Displaying video covers:

- Receiving video from the device
- Decoding the video
- Rendering and displaying the video

Integration methods:

☑ VideoSDK

☐ RCP+ over CGI

☐ RTSP

☐ RCP+ SDK

VideoSDK

Steps and codes	
1	Create a device connector.
	`private Bosch.VideoSDK.Device.DeviceConnector m_deviceConnector = new Bosch.VideoSDK.Device.DeviceConnector();`
2	Create a cameo to display the video.

	`/*creation of the Cameo ActiveX control*/`
	`private Bosch.VideoSDK.AxCameoLib.AxCameo m_axCameo = new Bosch.VideoSDK.AxCameoLib.AxCameo();`
	
	`/*configuration of the created control and addition to the child controls list of a GUI component*/`
	`PanelCameo.Controls.Add(m_axCameo);`
	`m_axCameo.Dock = DockStyle.Fill;`
	
	`/*retrieve the ICameo interface once the Cameo ActiveX control was visible for the first time*/`
	`private Bosch.VideoSDK.CameoLib.Cameo m_cameo = (Bosch.VideoSDK.CameoLib.Cameo)m_axCameo.GetOcx();`

3	Register to the connection event.
	`m_deviceConnector.ConnectResult += new Bosch.VideoSDK.GCALib._IDeviceConnectorEvents_ConnectResultEventHandler (DeviceConnector_ConnectResult);`
4	Connect to the BVIP device.
	`m_deviceConnector.ConnectAsync(string URL, string ProgID);`

Steps and codes	
5	When the connection is successfully established, the video stream can be assigned to the cameo to display the video.

```
private void
DeviceConnector_ConnectResult(Bosch.VideoSDK.Device.ConnectResultEnum
connectResult, string url, Bosch.VideoSDK.Device.DeviceProxy
deviceProxy)
{
   if (connectResult ==
      Bosch.VideoSDK.Device.ConnectResultEnum.creInitialized)
   {
      m_cameo.SetVideoStream(deviceProxy.VideoInputs[1].Stream);
   }
}
```

Further information

Video SDK Interface Definition.chm:

- Concepts > **Device Connection**
- Concepts > **Cameo ActiveX Control**
- Concepts > **Live Video**
- Class List > **ICameo**
- Class List > **IDeviceConnector**

Sample applications in the Video SDK installation directory:

- Sample Applications – Simple – **CSharpDesignerCameo**
- Sample Applications – Simple – **CSharpRuntimeCameo**
- Sample Applications – Simple – **JavaScript**

Partner Area on ipp.boschsecurity.com:

- Downloads > Code Samples > **IP_07_VSDK**

Related sections in this document:

- **Get Stream**, see page 15

2.2.2 Get Stream

This section describes how to get the raw video stream. The stream can then be post processed (adding some overlay) or decoded using the customers own decoder.

Integration methods:

☑ VideoSDK

☐ RCP+ over CGI

☑ RTSP

☑ RCP+ SDK

VideoSDK

Steps and codes	
1	Implement the virtual StreamItemProcessor class, mainly the ProcessFrame method, which is called when a new frame is delivered from the device.

	`//Very simple implementation of the virtual StreamItemProcessor class.`
	`//It counts the I frames.`
	`public partial class VsdkStreamItemProcessor`
	`: Bosch.VideoSDK.StreamItemsLib.IStreamItemProcessor`
	`{`
	`public int m_pframes, m_iframes, m_bframes;`
	`public VsdkStreamItemProcessor()`
	`{`
	`m_iframes =0;`
	`}`
	`public bool AcceptFrame(Guid logicalType){return true;}`
	`public void ProcessFrame(Guid logicalType,`
	`Bosch.VideoSDK.StreamItemsLib.IStreamItem streamItem)`
	`{`
	`//do something with the raw stream here`
	`if (streamItem.Flags==0x00000001)`
	`m_iframes++;`
	`}`
	`public void Reset(Guid logicalType){}`
	`}`

2	Create a new streamItemProvider and a new streamItemProcessor, connect the things together and start streaming. The PlaybackController is used to control replay sessions, which is not needed for live video.

	`m_streamItemProvider = new Bosch.VideoSDK.StreamItemsLib.`
	`StreamItemProvider();`
	`m_streamItemProvider.SelectStream(m_dataStream,`
	`VsdkGuids.LogicalChannelType_Video, 0);`
	`if (m_playbackController != null)`
	`m_streamItemProvider.PlaybackController = m_playbackController;`
	`m_streamItemProcessor = new VsdkStreamItemProcessor();`
	`m_streamItemProvider.StreamItemProcessor = m_streamItemProcessor;`
	`m_streamItemProvider.StartStreaming();`

Further information

Video SDK Interface Definition.chm:

- Concepts > **Access to Encoded Media Data**
- Class List > **IStreamItemProcessor**
- Class List > **IStreamItemProvider**
- Class List > **IStreamItem**
- Examples > **VSDKStreamItemProcessor.cs**

Partner Area on ipp.boschsecurity.com:

- Downloads > Code Samples > **GW_01_VSDK**

Related sections in this document:

- **Display Video**, see page 13

RTSP

All RTSP options are described in the document **RTSP usage with Bosch VIP Devices** which can be found in the download area.

Further information

Partner Area on ipp.boschsecurity.com:

- Downloads > Protocols > **RTSP**

3 PTZ

3.1 Pan Tilt Zoom (PTZ) and Presets

This section describes how to control a PTZ device. The most commonly used functions are pan, tilt, zoom, get presets and set presets. Newer models also support absolute positioning, so the position is given/retrieved via coordinates.

Integration methods:

☑ VideoSDK

☑ RCP+ over CGI

☐ RTSP

☑ RCP+ SDK

VideoSDK

Steps and codes	
1	Set the video controller.
	`videoInput.CameraController.SetController(string ModelName, string XmlConfig, Int32 ComPortNumber);`
2	Pan to the left with maximum speed.
	`videoInput.CameraController.PTZ(-100, 0, 0, PTZModeEnum.ptzNormalized);`
3	Tilt up with normal speed.
	`videoInput.CameraController.PTZ(0, 50, 0, PTZModeEnum.ptzNormalized);`
4	Zoom in slow.
	`videoInput.CameraController.PTZ(0, 0, 1, PTZModeEnum.ptzNormalized);`

Further information

Video SDK Interface Definition.chm:

- Concepts > **Camera Control**
- Class List > **ICameraController**
- Examples > **VSDKCameraController.cs**

Partner Area on ipp.boschsecurity.com:

- Downloads > Code Samples > **IP_08_VSDK**

RCP+ over CGI

There are two ways to control PTZ devices: Over the BICOM (preferred) or over the OSRD protocol. For analog cameras connected to a PTZ unit, the OSRD protocol must be chosen.

With the CONF_RCP_TRANSFER_TRANSPARENT_DATA (0xffdd) RCP+ command, the payload is redirected to the serial interface of the encoder. OSRD offers the basic functions like PTZ and get/set presets.

IP devices are still compatible with the protocol. For pure IP devices, the communication between the encoder and the analog camera part is done via the BICOM protocol.

With the CONF_BICOM_COMMAND (0x09a5) RCP+ command, the payload is redirected to the BICOM interface of the analog camera part. Together with basic PTZ and get/set presets functions BICOM offers some advanced settings which might be Intelligent Tracking, tour replay/ record, etc.

Next to PTZ related commands, you can change picture settings, activate night mode or even control the wiper (depending on the device) via this interface.

OSRD

The examples below show the basic functions only using the Opcode 5 for variable-speed PTZ commands. This Opcode is supported by all Bosch devices.

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

Steps and codes	
1	Stop all PTZ operations.
	`http://<Device IP>/rcp.xml?command=0xFFDD&type=P_OCTET&direction=WRITE&num=1&payload=0x000000058700000500000000C`
2	Continuous pan to the left with minimum and maximum speed.

	`http://<Device IP>/rcp.xml?command=0xFFDD&type=P_OCTET&direction=WRITE&num=1&payload=0x0000000587000005000802160`
	
	`http://<Device IP>/rcp.xml?command=0xFFDD&type=P_OCTET&direction=WRITE&num=1&payload=0x0000000587000005007802060`

3	Continuous pan to the right with minimum and maximum speed.

	`http://<Device IP>/rcp.xml?command=0xFFDD&type=P_OCTET&direction=WRITE&num=1&payload=0x0000000587000005000801150`
	
	`http://<Device IP>/rcp.xml?command=0xFFDD&type=P_OCTET&direction=WRITE&num=1&payload=0x0000000587000005007801050`

4	Continuous tilt up with minimum and maximum speed.

	`http://<Device IP>/rcp.xml?command=0xFFDD&type=P_OCTET&direction=WRITE&num=1&payload=0x0000000587000005010008150`
	
	`http://<Device IP>/rcp.xml?command=0xFFDD&type=P_OCTET&direction=WRITE&num=1&payload=0x000000058700000050F000823`

Steps and codes	
5	Continuous tilt down with minimum and maximum speed.
	http://<Device IP>/rcp.xml?command=0xFFDD &type=P_OCTET&direction=WRITE&num=1&payload=0x000000058700000501000411 http://<Device IP>/rcp.xml?command=0xFFDD &type=P_OCTET&direction=WRITE&num=1&payload=0x00000058700000050F00041F
6	Continuous zoom in with minimum and maximum speed.
	http://<Device IP>/rcp.xml?command=0xFFDD &type=P_OCTET&direction=WRITE&num=1&payload=0x000000058700000051000203C http://<Device IP>/rcp.xml?command=0xFFDD &type=P_OCTET&direction=WRITE&num=1&payload=0x000000058700000057000201C
7	Continuous zoom out with minimum and maximum speed.
	http://<Device IP>/rcp.xml?command=0xFFDD &type=P_OCTET&direction=WRITE&num=1&payload=0x000000058700000051000102C http://<Device IP>/rcp.xml?command=0xFFDD &type=P_OCTET&direction=WRITE&num=1&payload=0x000000058700000057000100C
8	Get the prepositions 1 and 99.
	http://<Device IP>/rcp.xml?command=0xFFDD &type=P_OCTET&direction=WRITE&num=1&payload=0x000000008600000070501 13 http://<Device IP>/rcp.xml?command=0xFFDD &type=P_OCTET&direction=WRITE&num=1&payload=0x000000008600000070563 75
9	Set the prepositions 1 and 99.
	http://<Device IP>/rcp.xml?command=0xFFDD &type=P_OCTET&direction=WRITE&num=1&payload=0x000000008600000070401 12 http://<Device IP>/rcp.xml?command=0xFFDD &type=P_OCTET&direction=WRITE&num=1&payload=0x000000008600000070463 74

BICOM

Please check the BICOM documentation for the command syntax.

The examples below shows the basic functions only.

Continuous Move

Steps and codes	
1	Stop all PTZ operations.
	http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085000000
2	Continuous pan to the left with minimum and maximum speed.
	http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085010000 http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x8000060110850F0000
3	Continuous pan to the right with minimum and maximum speed.
	http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085810000 http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x8000060110858F0000
4	Continuous tilt up with minimum and maximum speed.
	http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085008100 http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085008F00
5	Continuous tilt down with minimum and maximum speed.
	http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085000100 http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085000F00
6	Continuous zoom in with minimum and maximum speed.
	http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085000081 http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085000087

Steps and codes	
7	Continuous zoom out with minimum and maximum speed.

	`http://<Device IP>/rcp.xml?command=0x09A5&type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085000001`
	
	`http://<Device IP>/rcp.xml?command=0x09A5&type=P_OCTET&direction=WRITE&num=1&payload=0x800006011085000007`

8	Get the prepositions 1 and 99.

	`http://<Device IP>/rcp.xml?command=0x09A5&type=P_OCTET&direction=WRITE&num=1&payload=0x80000201B080070501`
	
	`http://<Device IP>/rcp.xml?command=0x09A5&type=P_OCTET&direction=WRITE&num=1&payload=0x80000201B080070563`

9	Set the prepositions 1 and 99.

	`http://<Device IP>/rcp.xml?command=0x09A5&type=P_OCTET&direction=WRITE&num=1&payload=0x80000201B080070401`
	
	`http://<Device IP>/rcp.xml?command=0x09A5&type=P_OCTET&direction=WRITE&num=1&payload=0x80000201B080070463`

Absolute Positioning

Steps and codes	
1	Get the pan position. The result is in radians × 10000. To get the degrees, you need to convert it first. The response from the code below is 0xc12d, which is 49453 decimal. Using the formula ((response/10000)/(2 × PI)) × 360, the pan position is 283,34 degrees.
	`http://<Device IP>/rcp.xml?command=0x09A5&type=P_OCTET&direction=WRITE&num=1&payload=0x810006011201`
2	Get the tilt position, using the same formula as above to get the value in degrees.
	`http://<Device IP>/rcp.xml?command=0x09A5&type=P_OCTET&direction=WRITE&num=1&payload=0x810006011301`
3	Get the zoom position. The zoom range is 0 to 255, where 0 means zoomed out, and 255 maximum optical zoom.
	`http://<Device IP>/rcp.xml?command=0x09A5&type=P_OCTET&direction=WRITE&num=1&payload=0x810006011401`
4	Set the pan position to 30 degrees, using the formula (Position/360) × 2 × PI × 10000. The position in radians × 10000 is 5236 (0x1474).
	`http://<Device IP>/rcp.xml?command=0x09A5&type=P_OCTET&direction=WRITE&num=1&payload=0x8100060112031474`

Steps and codes	
5	Set the tilt position to 180 degrees, using the formula above.
	`http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=W RITE&num=1&payload=0x8100060113037AB7`
6	Set a custom zoom position (0x7F).
	`http://<Device IP>/rcp.xml?command=0x09A5 &type=P_OCTET&direction=WR ITE&num=1&payload=0x810006011403007F`

Further information

Partner Area on ipp.boschsecurity.com:

- Downloads > Code Samples > **GW_02_Exe**
- Downloads > Code Samples > **GW_03_Exe**
- Downloads > Protocols > **BICOM**
- Downloads > Protocols > **OSRD**

3.2 ePTZ with ROI

This section describes how to use ROI/ePTZ.

Prerequisites:

– The seconds stream needs to be configured as ROI.
– SessionID is recommended to handle dual ROI configurations correctly.

When the region of interest (ROI) is configured electronic PTZ (ePTZ) can be used to display only part of the video.

Integration methods:

☑ VideoSDK

☑ RCP+ over CGI

☐ RTSP

☑ RCP+ SDK

All ePTZ features can be used via the same methods/interfaces as described in the Section **Pan Tilt Zoom (PTZ) and Presets**, see page 17.

4 Recording

4.1 Search and Replay

4.1.1 Search Video

In order to play back recordings, a search needs to be performed first to locate the desired recording.

With VideoSDK, the connection is established with the iSCSI medium itself, which means the BVIP device is not needed.

With RCP+ SDK, the connection is established to the device itself, no matter if the recordings are local, locally managed or centrally managed.

Integration methods:

☑ VideoSDK

☑ RCP+ over CGI

☐ RTSP

☑ RCP+ SDK

VideoSDK

For locally managed recordings, a connection to the device has to be established. If the recordings are centrally managed, a connection to the VRM has to be established.

First a track search is made to find out how many recording tracks are available. A camera can have either one track if single recording is active or two tracks if dual recording is active.

The track concept is very important for centrally managed recordings. In this case, the VRM has a unique TrackID for each recorded camera. This means that a camera has (up to) two tracks while a VRM can have multiple tracks, depending on how many cameras are recorded.

Once a track is selected, additional searches can be made on that track. The most common search is the **VideoRecorded** search, which shows recordings and gaps within a track.

Steps and codes	
1	Create a track search session.
	`m_trackSearchSession = m_deviceProxy.MediaDatabase.CreateSearchSession(Bosch.VideoSDK .MediaDatabase.SearchTypeEnum.steTrack);`
2	Set event handlers.
	`m_trackSearchSession.TrackAvailable += new Bosch.VideoSDK.GCALib. _ISearchSessionEvents_TrackAvailableEventHandler (m_searchSession_TrackAvailable); m_trackSearchSession.Progress += new Bosch.VideoSDK.GCALib. _ISearchSessionEvents_ProgressEventHandler (m_trackSearchSession_Progress);`

Steps and codes	
3	Start the track search.
	`m_trackSearchSession.Start();`
4	Collect the results.

	`void m_searchSession_TrackAvailable(`
	`Bosch.VideoSDK.MediaDatabase.SearchTypeEnum Type,`
	`Bosch.VideoSDK.MediaDatabase.Track pResult,`
	`Bosch.VideoSDK.MediaDatabase.SearchSession pSearchSession)`
	`{`
	`  /*do something here */`
	`}`

5	Wait until the track search has finished.

	`void m_trackSearchSession_Progress(int Progress,`
	`Bosch.VideoSDK.MediaDatabase.SeachSession pSearchSession)`
	`{`
	`  if (Progress == 100)`
	`  {/*do something here*/}`
	`}`

6	Create an event search session.

	`m_eventSearchSession =`
	`m_deviceProxy.MediaDatabase.CreateSearchSession(Bosch.VideoSDK`
	`.MediaDatabase.SearchTypeEnum.steEvent);`

7	Define the event type and track ID.
	`m_eventSearchSession.AddEventFilter(EventTypeEnum EventType);` `m_eventSearchSession.AddIdentifierFilter(Int32 TrackID);`
8	Set event handlers.

	`m_eventSearchSession.ResultAvailable += new Bosch.VideoSDK.GCALib.`
	`_ISearchSessionEvents_ResultAvailableEventHandler`
	`(m_eventSearchSession_ResultAvailable);`
	`m_eventSearchSession.Progress += new Bosch.VideoSDK.GCALib.`
	`_ISearchSessionEvents_ProgressEventHandler`
	`(m_eventSearchSession_Progress);`

9	Start the event search.
	`m_eventSearchSession.Start();`
10	Collect the results.

	`void m_eventSearchSession_ResultAvailable(`
	`Bosch.VideoSDK.MediaDatabase.SearchTypeEnum Type,`
	`Bosch.VideoSDK.MediaDatabase.SearchResult pResult,`
	`Bosch.VideoSDK.MediaDatabase.SearchSession pSearchSession)`
	`{/*do something here*/}`

Steps and codes	
11	Wait until the event search has finished.

	`void m_eventSearchSession_Progress`
	`(int Progress, Bosch.VideoSDK.MediaDatabase.SearchSession`
	`pSearchSession)`
	`{`
	`  if (Progress == 100)`
	`  {/*do something here*/}`
	`}`

Further information

Video SDK Interface Definition.chm:

- Concepts > **Media Database Search**
- Class List > **IMediaDatabase**
- Class List > **ISearchSession**

Partner Area on ipp.boschsecurity.com:

- Downloads > Code Samples > **IP_09_VSDK**

Related sections in this document:

- **Replay**, see page 29
- **Export**, see page 32

RCP+ over CGI

Note:

Search of recorded video over RTSP only works for locally managed recordings, not for centrally managed (by VRM) recordings and requires firmware version 5.70 or higher.

To enable searching on a recorded video stream a correct setup of the RTSP replay connection is preconditioned.

The track, the search is performed on, is specified in the setup of the RTSP replay connection.

Steps and codes	
1	Set up a RTSP connection for replay.
	`rtsp://<Device IP>/?rtsp_tunnel?line=1&inst=1&rec=1&rnd=718`
2	Get the RTSP session ID with the help of the random number specified in the URL used to request the recorded video stream via RTSP with the command CONF_GET_RTSP_-SESSION_ID.
	`http://<Device IP>/rcp.xml?command=0x0ae8 &type=T_DWORD&direction=READ&protocol=TCP&num=718` The session ID is for example: 154861654
3	Start the search with the command CONF_HD_PARTITION_FILE_INFO. Set the start and end time of the search in seconds since 01.01.2000 00:00h and the max entry number. In the code below 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.
	`http://<Device IP>rcp.xml?command=0x0901 &type=P_OCTET&direction=READ&protocol=TCP &payload=0x1849B660184A0AC00000000400000000&num=1&sessionid=154861654`
	The reply contains all slices up to the max entry number specified in the search command.

Further information

Partner Area on ipp.boschsecurity.com:

– Downloads > Protocols > **RTSP**

Related sections in this document:

– **Replay**, see page 29

4.1.2 Replay

Replay lets you view recorded video material. The recordings can be centrally managed (by VRM) or locally managed by the device itself.

Integration methods:

☑ VideoSDK

☑ RCP+ over CGI

☐ RTSP

☑ RCP+ SDK

VideoSDK

For locally managed recordings, a connection to the device has to be established. If the recordings are centrally managed, a connection to the VRM has to be established.

In order to replay video, the desired recording must first be located using a search. This can be done using the **Search video** function of the VideoSDK.

Steps and codes	
1	Create a playbackController to navigate in the recording.
	`m_playbackController = new Bosch.VideoSDK.MediaDatabase.PlaybackController();`
2	Connect the playbackController to a specific track which has been received with the TrackSearch session before.
	`m_mediaSession = m_mediaDatabase.GetMediaSession(m_trackInfo.TrackID, m_playbackController);`
3	Seek to the time you are interested in.
	`m_playbackController.Seek(Bosch.VideoSDK.MediaDatabase.Time64 seektime);`
4	Set the ReplaySpeed. VideoSDK supports forward (value >0), backward replay (value <0) and pause (value = 0) modes.
	`m_playbackController.Play(Int32 Rate);`
5	Assign the replay stream to a cameo.
	`m_cameo.SetVideoStream(m_mediaSession.GetVideoStream());`

Further information

Video SDK Interface Definition.chm:

- Concepts > **Media Database Replay**
- Concepts > **Cameo ActiveX Control**
- Class List > **IMediaDatabase**
- Class List > **IPlaybackController**

Sample applications in the Video SDK installation directory:

- Sample Applications – Simple – **CSharpDesignerCameo**
- Sample Applications – Simple – **CSharpRuntimeCameo**

Partner Area on ipp.boschsecurity.com:

- Downloads > Code Samples > **IP_10_VSDK**

Related sections in this document:

- **Search Video**, see page 25
- **Export**, see page 32

RCP+ over CGI

Note:

For centrally managed (by VRM) recordings firmware version 5.70 and VRM version 3.0 or higher is required.

In case of controlling the replay of a recorded video stream a correct setup of the RTSP replay connection is preconditioned.

Steps and codes	
1	Set up a RTSP connection for replay with a random choosen rnd number.
	`rtsp://<Device IP>/?rtsp_tunnel?line=1&inst=1&rec=1&rnd=718`
2	Get the RTSP session ID with the help of the random number specified in the URL used to request the recorded video stream via RTSP with the RCP+ command CONF_GET_RTSP_SESSION_ID. In the example below, the sessionID is 154861654.
	`http://<Device IP>/rcp.xml?command=0x0ae8 &type=T_DWORD&direction=READ&protocol=TCP&num=718`
3	Stop/pause the HD replay at the current position (default value is play with speed 100 %) with the RCP+ command CONF_HD_REPLAY_START.
	`http://<Device IP>/rcp.xml?command=0x0902 &type=T_INT&direction=WRITE&num=1&sessionid=154861654 &payload=0x00000000`
4	Seek to the time in seconds since 01.01.2000 00:00h you are interested in with the RCP+ command CONF_HD_REPLAY_SEEK_TIME. Example below: 29.11.2012 11:38:20 (local time).
	`http://<Device IP>/rcp.xml?command=0x0905 &type=P_OCTET&direction=WRITE&num=1&sessionid=154861654 &payload=0x184A05AC`
5	Start the playback of the recorded video stream with normal speed (100%) with the RCP+ command CONF_HD_REPLAY_START. Forward (values >0), backward (values <0)and stop/pause (value = 0) replay modes are supported.
	`http://<Device IP>/rcp.xml?command=0x0902 &type=T_INT&direction=WRITE&num=1&sessionid=154861654 &payload=0x00000064`

Further information

Partner Area on ipp.boschsecurity.com:

– Downloads > Protocols > **RTSP**

Related sections in this document:

– **Search Video**, see page 25

4.2 Export

Recorded video and audio can be exported to a file on the local HDD or to FTP/cloud storage.

Integration methods:

☑ VideoSDK

☐ RCP+ over CGI

☐ RTSP

☑ RCP+ SDK

VideoSDK

The export file is saved on local or network attached storage.

It can be in VideoSDK's native format which means that no post processing is done when the re-cording is retrieved from the storage. It can be played back either with your own solution or via Bosch BVC or BVMS.

The file can also be stored in .asf format, which can be opened with Windows Media Player or other media players. In this case, the video data is transcoded in real time as it is received from storage, which may cause heavy CPU load.

In order to export video, a search needs to be executed (see page 25).

Steps and codes	
1	Create PlaybackController.
	`m_playbackController = Bosch.VideoSDK.MediaDatabase.PlaybackController();`
2	Init MediaFileWriter and set the export properties.

	`m_mediaFileWriter = new Bosch.VideoSDK.MediaDatabase.MediaFileWriter();`
	`m_mediaFileWriter.FileFormat = format;// asf or MPEGActiveX`
	`m_mediaFileWriter.FileSizeLimitKB = 0; // no limit`
	`m_mediaFileWriter.MaximumNumberOfFiles = 0; //no limit`
	`m_mediaFileWriter.RecordingStartTime = Bosch.VideoSDK.MediaDatabase.Time64 RecordingStartTime;`
	`m_mediaFileWriter.RecordingEndTime = Bosch.VideoSDK.MediaDatabase.Time64 RecordingEndTime;`

3	Set event handlers.

	`m_mediaFileWriter.Progress +=new Bosch.VideoSDK.GCALib._IMediaFileWriterEvents_ ProgressEventHandler(m_mediaFileWriter_Progress);`
	`m_mediaFileWriter.RecordingStopped +=new Bosch.VideoSDK.GCALib._IMediaFileWriterEvents_ RecordingStoppedEventHandler(m_mediaFileWriter_RecordingStopped);`
	`m_mediaFileWriter.NewFileCreated +=new Bosch.VideoSDK.GCALib._IMediaFileWriterEvents_ NewFileCreatedEventHandler(m_mediaFileWriter_NewFileCreated);`

Steps and codes	
4	Set stream options.
	`m_mediaFileWriter.AddStream(Bosch.VideoSDK.DataStream Stream, MediaTypeEnum MediaType, Int32 TrackID, string TrackName, string SourceURL, Int32 SourceID);`
5	Start export.
	`m_mediaFileWriter.StartRecording(string Filename, string fileComment);`

Further information

Video SDK Interface Definition.chm:

- Concepts > **Media Database Export**
- Class List > **IMediaDatabase**
- Class List > **IPlaybackController**
- Class List > **IMediaFileWriter**

Partner Area on ipp.boschsecurity.com:

- Downloads > Code Samples > **IP_11_VSDK**
- Downloads > Code Samples > **IP_20_VSDK**

Related sections in this document:

- **Search Video**, see page 25
- **Replay**, see page 29

5 Events

5.1 General Event Handling

Bosch video over IP devices send events which the user can choose to receive or not. They can be used to check device status like video loss, detect a broken/disconnected video cable or implement your own alarm scenarios when motion/inputs are triggered.

Integration methods:

☑ VideoSDK

☑ RCP+ over CGI

☐ RTSP

☑ RCP+ SDK

VideoSDK

Steps and codes	
1	Register to the alarm input event handler to handle input events.
	`m_alarmInput.StateChanged += new Bosch.VideoSDK.GCALib._IAlarmInputEvents_StateChangedEventHandler(AlarmInput_StateChanged);`
2	Get the input events.
	`private void AlarmInput_StateChanged(Bosch.VideoSDK.Live.AlarmInput eventSource, bool state) {/*do something here*/}`
3	Register to the output (relay) events.
	`m_relay.StateChanged += new Bosch.VideoSDK.GCALib._IRelayEvents_StateChangedEventHandler (Relay_StateChanged);`
4	Register to video loss and motion events.
	`m_videoInput.SignalStateChanged += new Bosch.VideoSDK.GCALib._IVideoInputEvents_SignalStateChangedEventHandler(VideoInput_SignalStateChanged);`
5	Get the video loss and motion events.
	`private void VideoInput_SignalStateChanged(Bosch.VideoSDK.Live.VideoInput event-Source, bool signalState, bool motion, bool alarm) {/*do something here*/}`
6	Register to recording status events.
	`m_videoInputRecordingEvents.RecordingStateChanged += new Bosch.VideoSDK.GCALib._IVideoInputRecordingEvents_RecordingStateChangedEventHandler (VideoInputRecordingEvents_RecordingStateChanged);`

Steps and codes	
7	Get the recording status events.
	`private void VideoInputRecordingEvents_RecordingStateChanged (Bosch.VideoSDK.Live.VideoInput eventSource, Bosch.VideoSDK.GCALib.RecordingStateEnum recordingState) {/*do something here*/}`

Further information

Video SDK Interface Definition.chm:

- Concepts > Fundamentals > **Event Handling in Managed Applications**
- Class List > **IAlarmInputEvents**
- Class List > **IRelayEvents**
- Class List > **IVideoInputStateEvents**
- Class List > **IVideoInputRecordingEvents**

Partner Area on ipp.boschsecurity.com:

- Downloads > Code Samples > **IP_04_VSDK**

RCP+ over CGI

RCP+ messages will be processed and received by the CGI client using a poll mechanism. A CGI command set with the requested messages command tag numbers need 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 'collect-ms' CGI value to the appropriate number of milliseconds.

In order to ensure correct assignment of messages in case several clients are polling for messages or if a polling client uses different socket connections, a unique message domain ID (CGI parameter 'msgdomainID') should be provided. Messages are then collected separately for each ID and can be uniquely assigned to the polling client even if requesting via different socket connections.

Steps and codes	
1	Subscribe to messages. In the example below, subscription is done for motion alarm and input pin state messages.
	`http://<Device IP>/rcp.xml?message=0x01c0$0x01c3&collectms=5000`

Further information

Partner Area on ipp.boschsecurity.com:

- Downloads > Protocols > **RCP+ over CGI**

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