BOSCH
Invented for life

Tech Note
Working the JPEG way with BVIP

- Easy integration into 3rd party software application
- Simple Web browser control
- OS platform independent
- Encapsulated in CGI commands

Bosch IP Video products have the ability to provide multiple and different kinds of video streams. Besides the MPEG-4 video streams it is also possible to base video from the units on JPEG images.
This might be especially useful if the video management system is not able to have the MPEG-4 video decoder integrated, or the units need to provide feeds to another system, like Access Control or Alarm Verification systems, where often a high frame rate is not required.

The unit's Web browser interface that can be used to configure the units, view live or playback video via the Web browser also has a view of a JPEG image.

This tech note shall give a few hints of how JPEG images can be derived from the units via HTTP and the newly introduced Common Gateway Interface (CGI), e.g. with a Web browser, a Web server-based application, or in a "foreign" environment, meaning third party software.

The following examples assume having installed

- a Dinion IP camera at IP address 160.10.0.1,
- an Autodome IP camera at IP address 160.10.0.4,
- a VideoJet X40 at IP address 160.10.0.40 with 4 cameras connected.

Taking a JPEG snapshot

JPEG from an IP camera

To get a snapshot image from the Dinion IP the URL must look like:
HTTP://160.10.0.1/snap.jpg

This will show a snapshot of the camera in the default image size which is CIF and a default quality setting.

(CIF, 23 kB)

Similarly, if the unit is password-protected, user name and password must be provided, making e.g. the URL look like:
HTTP://user:myuserpwd@160.10.0.1/snap.jpg

This is further not assumed in our examples.

Sometimes it is preferred or necessary to have a different image size. For that, the units can deliver different sizes by providing the case-sensitive parameter "JpegSize" with one of the possible values.

Here is the list:

S	(small)	176 × 144/120 pixels (QCIF)
M	(medium)	352 × 288/240 pixels (CIF)
L	(large)	704 × 288/240 pixels (2CIF)
XL	(maximum)	704 × 576/480 pixels (4CIF)

The parameter is added to the URL in a query string appended after a question mark.
For receiving a JPEG image in maximum resolution the URL must look like:
HTTP://160.10.0.1/snap.jpg?JpegSize=XL

JPEGs from multi-channel units

The video inputs and the respective cameras of multi-channel units can also be selected by handing over a parameter with the snap.jpg command to the VideoJet X40, this time the also case-sensitive parameter “JpegCam”.

To connect to video from the second camera of the VideoJet X40, the URL must look like:
HTTP://160.10.0.40/snap.jpg?JpegCam=2

Multiple parameters can be handed over equally to standard query string notation in an URL, separated by “&”. The order of the parameters in the query string is not relevant.

Expecting that we also want to see the highest resolution image from the second camera on our VideoJet X40, using 4CIF resolution, the URL must look like:
HTTP://160.10.0.40/snap.jpg?JpegCam=2&JpegSize=XL

Tweaking the JPEG quality

The units use a pre-selected quality setting for all JPEG requests, which defaults to 4 and resembles a fairly good quality.

The required case-sensitive parameter for tweaking this is called “JpegQuality” and has a range from 1 (best) to 32 (modest).

In some applications, e.g. when requiring a small-sized QCIF image only, it could be possible to decrease the quality without loosing details but save bandwidth and thus transmission time.
In such a case, taking a low-quality QCIF snapshot from the Dinion IP, the URL could look like:
HTTP://160.10.0.1/snap.jpg?JpegSize=S&JpegQuality=32


(QCIF, 9 kB)

In another usage, e.g. for a large-scaled 4CIF display, it could be required to increase the quality to make use of the maximum details available.

In this case, taking a highest-quality 4CIF snapshot from the Autodome IP, the URL could look like:
HTTP://160.10.0.4/snap.jpg?JpegSize=XL&JpegQuality=1

(4CIF, 119 kB)

JPEGs in multi-view

From multi-channel units a JPEG containing more than only one video source can be derived.

Each of the maximum four video sources is identified by a single bit. Camera 1 is represented by bit 0, camera 2 by bit 1, camera 3 by bit 2 and camera 4 by bit 3.
These bits are combined into a decimal value handed over with the case-sensitive parameter “JpegCamBits”.

Resulting from this, a quad-view of all 4 cameras would be resembled by all four bits set, creating a value of hexadecimal 0xF, or decimal 15.

Multi-view JPEG images are always scaled to 4CIF width (704 pixels), so the parameter JpegSize is not required. A single camera displayed in one row is scaled to 2CIF.

Following our example, a quad-view JPEG image in 4CIF resolution from our VideoJet X40 would result in the following URL:
HTTP://160.10.0.40/snap.jpg?JpegCamBits=15

This will produce an image like this:

We might want to have our four video sources separated a bit for better analysis or a nicer display in an array of multiple JPEGs.

This can be achieved by setting a border around the camera images. The width and color of this border can be defined. Please note that the borders add to the image size.

The color is defined as hexadecimal values for Y'UV.
Y represents the luminance value (the black-and-white component) and basically defines the brightness of the border color, where 0x00 is dark and 0xff is bright.
U and V represent the so-called chrominance (color) values where U defines the blue-luminance difference value, and V the red-luminance difference value.

For those who want to gain deeper knowledge about Y'UV and the color space definitions there is a pretty good explanation in Wikipedia.

These 3 bytes are followed by a two-digit value for the border width in steps of 16 pixels, resulting in values of 0x10, 0x20, ... 0xf0. The border is drawn at the top and the right side of each of the single camera images. This allows to seamlessly fit multiple JPEGs from various units on one larger screen.

To set the border color properly you either need some understanding of the Y'UV colour scheme – or you simply play around with it a bit.

For example, if you want to draw a large red border you should set the Y value to a modest brightness, let's say 0x40, put in a balanced value for blue against green, let's say also 0x40, put the V value to a red-saturated 0xff, and set the border width to 0x80.

The resulting URL should look like:
HTTP://160.10.0.40/snap.jpg?JpegCamBits=15&JpegBorder=0x4040ff80

This will produce an ugly image that no one would like to look at:

So we will define our border color as a medium-grey as this is least annoying. The values shall be modest 0x80 for all Y, U and V. The border shall be 16 pixels.

Based on this example, a quad-view JPEG image with the defined border settings would result in the following URL:
HTTP://160.10.0.40/snap.jpg?JpegCamBits=15&JpegBorder=0x80808010

This will produce an image like this:

"Streaming" JPEG

JPEGs continuously updated

Video management systems typically provide live viewing of cameras. In many cases a frame rate lower than 25/30 images per second is quite sufficient. These cases could be covered with a "quick" update of the JPEG image.

If this update is a kind of automated or scheduled, so that the impression is tending towards moving pictures, we call this "Motion JPEG".

There is no public standard yet that defines the transmission of a "Motion JPEG" stream, so you find many different implementations to achieve a "quasi"-video impression.

BVIP units running firmware 3.0 or later can be used in a mode where almost all the computing performance can be used solely for JPEG creation. As continuous JPEG encoding requires much more performance than MPEG-4 encoding with comparable quality no MPEG-4 encoding would then be possible and also no video content analysis.

Doing so the units can provide up to 25/30 JPEG images per second, which is – from the Motion JPEG view – equal to the maximum frame rate of MPEG-4 video.
But please note that much more data needs to be transmitted than with video as JPEG images can not benefit from the more advanced MPEG-4 compression algorithms.

A management system can implement continuous requesting of updated JPEG images from BVIP units. Some measures must be taken to avoid too many requests that could possibly flood the network if not taken care of.
So there is a mechanism available to avoid transmission of identical JPEGs that already have been sent, e.g. due to too fast repetition of requests.

A viewing system – we will call it "client" – can add an ID to the URL. This ID is defined by the client itself and should be unique throughout the total system, meaning every client shall have its own different ID.

The ID is handed over with the case-sensitive parameter "JpegDomain" and can be any number or alphanumeric string.

The BVIP unit checks this ID and only transmits a JPEG image after an update is available.

If the client is a Web browser, or the viewing system is browser-based, there could be a problem with the Web browser caching an already requested URL.
That is nothing we want.

The solution is to add another parameter to the URL that "fools" the Web browser. The case-sensitive parameter "Counter" will not be evaluated by the BVIP unit but will force the Web browser to treat this as a brand-new request and leapfrog the browser cache.

A snapshot of a complete URL combined by the client to repeatedly retrieve JPEG images from our VideoJet X40 would then look like:
HTTP://160.10.0.40/snap.jpg?JpegDomain=myclient123&Counter=12345

The JpegDomain with "myclient123" must remain stable while Counter will e.g. be continuously incremented with every request.

JPEG Push with JavaScript

BVIP units provide a "Motion JPEG" tab on their livepage. As this page is accessed via a Web browser we can make advantage of scripting possibilities to get a kind of automation.

The most common scripting language in a Web browser is JavaScript. It is relatively easy to learn and understand for simple applications while providing some useful, sometimes really powerful features.

Our BVIP products' Web pages use JavaScript to gain maximum usability for configuration and operation of the units. They also use it for the Motion JPEG page.

The following example requires some experienced knowledge of JavaScript and HTML.

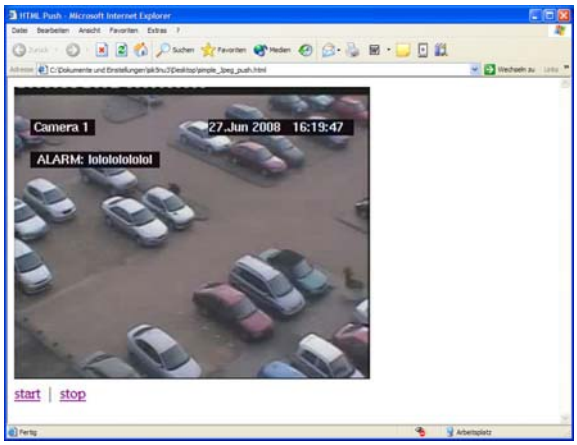
The script that handles the continuous "pushing" of JPEG images towards the browser is embedded in the BVIP unit and is called "pushimage.js". It provides a class of JavaScript functions that allow creation of a JPEG image element in the Web browser and – beside others – start/stop functions.

We can create a simple HTML page that uses just these basic functions to get a continuous stream from our Dinion IP.
This HTML page includes the JavaScript file pushimage.js, which is downloaded from the Dinion IP, and displays the JPEG images and start/stop control links.

Here is the source code of the HTML page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<HTML>
 <HEAD>
  <TITLE>
   HTML Push
  </TITLE>
    <script language="JavaScript"
        type="text/javascript"
        src="http://160.10.0.1/pushimage.js">
    </script>
    <script type="text/javascript">
        var pimg1;
        var debugarea;
        function init() {
            pimg1=createPushImage("Push1", 0);
            start();
        }
        function stop() {
            pimg1.stopPush();
        }
        function start() {
            pimg1.startPush();
        }
    </script>
 </HEAD>
 <BODY onLoad="init()">
  <IMG alt="" name="Push1" id="Push1"
      src="http://160.10.0.1/snap.jpg>
  <br>
  <a href="javascript:start()">start</a>
   | 
  <a href="javascript:stop()">stop</a>
 </BODY>
</HTML>
```

We open this file in our Web browser. As we can see, on loading the page an image element is created, the refresh of the JPEG is initiated, and the JPEG image from the Dinion IP camera is displayed at the maximum possible speed.
This will now run until we click "stop" which leaves the latest JPEG image on display.



Americas:
Bosch Security Systems
130 Perinton Parkway
Fairport, New York, 14450, USA
Phone:+1 585 223 4060
Fax:+1 800 289 0096
security.sales@us.bosch.com
http://www.boschsecurity.us

Europe, Middle East, Africa:
Bosch Security Systems B.V.
P.O. Box 80002
5600 JB Eindhoven, The Netherlands
Phone:+31 (0) 40 27 83955
Fax:+31 (0) 40 27 86668
emea.securitysystems@bosch.com
http://www.boschsecurity.com

Asia-Pacific:
Bosch Security Systems Pte Ltd
38C Jalan Pemimpin
Singapore 577180
Phone:+65 6319 3450
Fax:+65 6319 3499
apr.securitysystems@bosch.com
http://www.boschsecurity.com