Axis Video API, HTTP - Interface Specification

Revision: 1.14
Date: 2004-April-22

TABLE OF CONTENTS

Enter your E-mail:

OK

Other e-newsletter options

Search

DOCUMENT HISTORY

Version	**Date**	**Comment**
1.00	2000-nov-01	Initial version

1.01	2000-nov-13	Added info about dataout and altered description of wait and timeout in "serial port control".
1.02	2000-nov-28	Added entry for controlling image buffers.
1.03	2000-nov-30	Added entries for closing PPP connection and resetting PPP connection maxtimer.
1.04	2000-dec-11	Added examples for setting motion detection parameters.
1.05	2001-jan-11	Corrected multipart boundary examples.
1.06	2001-jan-16	Removed check of output on IO-ports (until it is supported).
1.07	2001-sep-07	Added Audio support.
1.08	2001-nov-22	Changed audio MIME type and audio parameters.
1.09	2002-apr-15	Added frame rate control.
1.10	2002-jul-04	Added MPEG-2, recording, event and firewall support. Grouped the API requests into different sections.
1.11	2003-may-07	Updated and corrected Recording interface.
1.12	2003-dec-04	Added MPEG-2 multicast info request and response. Removed delay parameter and added predelay and postdelay parameters to JPEG buffer request.
1.13	2004-jan-22	Corrected description for JPEG buffer request, do=reset.
1.14	2004-apr-22	Added better description and an example for the info parameter to the ptz.cgi. Corrected some response descriptions.

1 OVERVIEW

This document specifies the external HTTP based application programming interface of the Axis camera and video servers.

The HTTP-based video interface provides the functionality for requesting single and multi-part images, for controlling camera functions (PTZ, output relay etc.) and for getting and setting internal parameter values. The image and CGI-requests are handled by the built-in Web server in the camera and video server.

1.1 Product and firmware versions

The support for this stated HTTP API is very dependent on the product and firmware release. Please refer to the Release Notes for the actual product for compliance information.

2 REFERENCES

HTTP protocol

Hypertext Transfer Protocol -- HTTP/1.0

External application programming interfaces (Client side)

Axis Video API, HTTP

Axis Video Product specific API Notes

3 DEFINITIONS

This section contains information on general usage of this document.

3.1 General notations

3.1.1 General abbreviations

The following abbreviations are used throughout this document

CGI	*Common Gateway Interface* - a standardized method of communication between a client (e.g., a web browser) and a server (e.g., a web server).
TBD	*To be done/designed* - notifies the reader that the referenced section/subsection/entity is intended to be specified, but has not reached a level of maturity to be public at this time.
N/A	*Not applicable* - a feature/parameter/value is of no use in a specific task.

3.1.2 Style convention

In URL syntax and in descriptions of CGI parameters, text in italic within angle brackets denotes content that should be replaced with either a value or a string. When replacing the text string, the angle brackets must also be replaced. An example of this is the description of the name for the server, denoted with `<`*`servername`*`>` in the URL syntax description below, which is replaced with the string `myserver` in the URL syntax example, also shown below.

URL syntax is written with the word "Syntax:" shown in bold face, followed by a box with the referred syntax, as shown below. The name of the server is written as `<`*`servername`*`>`. This is intended to be replaced with the name of the actual server. This can either be a name, e.g., "thecam" or "thecam.adomain.net" or the associated IP number for the server, e.g., `10.10.2.139`.

Syntax:

```
http://<servername>/jpg/image.jpg
```

A description of returned data is written with "Return:" in bold face, followed by the returned data in a box. All data returned as HTTP formatted, i.e., starting with the string `HTTP`, is line-separated with a Carriage Return and Line Feed (CRLF) printed as `\r\n`.

Return:

```
HTTP/1.0 <HTTP code> <HTTP text>\r\n
```

URL syntax examples are written with "Example:" in bold face, followed by a short description and a light grey box with the example.

Example: request default image

```
http://myserver/jpg/image.jpg
```

Examples of what can be returned by the server from a request is written with "Example:" in bold face, followed by a short description and a light grey box with an example of the returned data.

Example: returned data after a successful request

```
HTTP/1.0 200 Ok\r\n
```

3.1.3 General CGI URL syntax and parameters

CGI URLs are written in lower-case. CGI parameters are written in lower-case and as one word, with no underscores or other separators. When the CGI request includes internal camera parameters, the internal parameters must be written exactly as named in the camera or video server. The CGIs are organized in function related directories under the *axis-cgi* directory. The file extension of the CGI is required.

Syntax:

```
http://<servername>/axis-cgi/<subdir>[/<subdir>...]/<cgi>.<ext>
[?<parameter>=<value>[&<parameter>=<value>...]]
```

Example: setting PTZ parameters

```
http://myserver/axis-cgi/com/ptz.cgi?camera=1&move=home
```

3.1.4 Parameter value convention

In tables defining CGI parameters and supported parameter values, the default value for optional parameters is system configured.

4 INTERFACE SPECIFICATION

4.1 Naming conventions and URL syntax

4.1.1 Obsolete CGI parameters

Some CGI parameters and values in this specification are obsolete and are provided for backward compatibility. These might not be supported in the future.

Obsolete parameters and values are stated in the request descriptions.

4.2 Server responses

4.2.1 HTTP status codes

The built-in Web server uses the standard HTTP status codes.

Return:

```
HTTP/1.0 <HTTP code> <HTTP text>\r\n
```

with the following HTTP code and meanings

HTTP code	HTTP text	Description
200	OK	The request has succeeded, but an application error can still occur, which will be returned as an application error code.
204	No Content	The server has fulfilled the request, but there is no new information to send back.
400	Bad Request	The request had bad syntax or was inherently impossible to be satisfied.
401	Unauthorized	The request requires user authentication or the authorization has been refused.
404	Not Found	The server has not found anything matching the request.
409	Conflict	The request could not be completed due to a conflict with the current state of the resource.
500	Internal Error	The server encountered an unexpected condition that prevented it from fulfilling the request.
503	Service Unavailable	The server is unable to handle the request due to temporary overload.

Example: request includes invalid file names.

```
HTTP/1.0 404 Not Found\r\n
```

5 API GROUPS

To make it easier for developers to get an idea of which API requests are supported for different products, the requests have been grouped together. Information about which groups are supported can be found in the product-specific release notes document, available for download from the Axis web site.

5.1 General

The requests specified in the General section are supported by all video products.

5.1.1 Get camera parameter values

Get a camera's parameter values.

Note: This request requires administrator access (administrator authorization).

Method: GET/POST

Syntax:

```
http://<servername>/axis-cgi/admin/getparam.cgi?<parameter>[&<parameter>...]
```

<parameter>= <value>	**Values**	**Description**
<string> [1]	*<group>*[.*<name>*]	Returns the value of the camera parameter named *<group>*.*<name>* as described in 5.1.2. If *<name>* is omitted, all the parameters of the *<group>* are returned The camera parameters must be entered exactly as named in the camera or the video server.

[1] The supported parameters are product/release-dependent.

Example: Get the IP address

```
http://myserver/axis-cgi/admin/getparam.cgi?Network.IPAddress
```

Example: Get all of the network parameters

```
http://myserver/axis-cgi/admin/getparam.cgi?Network
```

5.1.2 Camera parameter values response

When querying parameter values, the current parameter values are returned.

Successful control requests returns parameter pairs, as follows.

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
<parameter pair>
```

where *<parameter pair>* is

```
<parameter>=<value>\n
[ <parameter pair> ]
```

Example: PTZ position query response

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
pan=-50\n
tilt=0\n
zoom=500\n
```

If the CGI request includes an invalid parameter value, the server returns an error message.

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
Error: <error text (or code)>\n
<description>\n
```

5.1.3 Set camera parameter values

Set a camera's parameter values.

Note: This request requires administrator access (administrator authorization).

Method: GET/POST

Syntax:

```
http://<servername>/axis-cgi/admin/setparam.cgi?<parameter>=<value>
[&<parameter>=<value>...]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
<string>=<string>	*<group.name>* = <value>	Assigns *<value>* to the parameter *<group.name>* The *<value>* must be URL-encoded when it contains non-alphanumeric characters. The camera parameters must be entered exactly as named in the camera or the video server.
nosync=<string> [1]	yes	Specifies that there should be no sync (write) of the corresponding configuration file to flash. If parameter is omitted, a sync will occur.

Example: Set default image resolution to 320x240 pixels

```
http://myserver/axis-cgi/admin/setparam.cgi?Image.Resolution=320x240
```

Example: Set the bandwidth limitation to 500, without writing the corresponding configuration file to flash

```
http://myserver/axis-cgi/admin/setparam.cgi?Network.Bandwidth=500&nosync=yes
```

5.1.4 Factory default

Reload factory default

Note: This request requires administrator access (administrator authorization).

Method: GET

Syntax:

```
http://<servername>/axis-cgi/admin/factorydefault.cgi
```

5.1.5 Restart server

Restart server

Note: This request requires administrator access (administrator authorization).

Method: GET

Syntax:

```
http://<servername>/axis-cgi/admin/restart.cgi
```

5.1.6 Server report

This CGI request generates and returns a server report. This report is useful as an input when requesting support. The report includes product information, parameter settings and system logs.

Note: This request requires administrator access (administrator authorization).

Method: GET

Syntax:

```
http://<servername>/axis-cgi/admin/serverreport.cgi
```

5.1.7 System logs

Get system log information

Note: This request requires administrator access (administrator authorization).

Note: The response is product/release-dependent.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/admin/systemlog.cgi
```

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
<system log information>
```

5.2 JPEG/MJPG

The requests specified in the JPEG/MJPG section are supported by those video products that use JPEG/MJPG encoding.

5.2.1 JPEG image request

Request JPEG image

Method: GET

Syntax:

```
http://<servername>/jpg[/<camera>]/<name>.jpg
```

with the following parameters

Parameter	Values	Description
<camera>	1, ... [1]	Select input source. Applies only to servers with more than one input source. *Default:* default camera
<name>	image, quad[1], halfsize[2], fullsize[2], hugesize[2]	"image" returns an image with the default resolution and compression, as defined in the system configuration. The *<camera>* option is not allowed in quad request.

[1] Product-dependent. Check the product specification.

[2] Obsolete.

Example: request JPEG image from default camera with default resolution and compression

```
http://myserver/jpg/image.jpg
```

5.2.2 JPEG image (snapshot) CGI request

Request a JPEG image (snapshot) with specified properties.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/jpg/image.cgi
[?<parameter>=<value>[&<parameter>=<value>...]]
```

<parameter>=<value>	**Values**	**Description**
resolution=<int>x<int>	*<width>*[1],*<height>*[1]	Specify the resolution as *<width>* times *<height>* number of pixels of the returned image.
camera=<int>	1, ... [1]	Applies only to video servers with more than one video input. Selects the source camera.
compression=<int>	0 - 100 [1]	Adjusts the compression level of the image. Higher values correspond to higher compression, i.e. lower quality and smaller image size. **Note:** This value is internally mapped and is therefore product-dependent.
colorlevel=<int>	0 - 100 [1]	Sets level of color or grayscale. 0 = grayscale, 100 = full color. **Note:** This value is internally mapped and is therefore product-dependent.
clock=<int>	0, 1	Shows/hides the time stamp. 0 = hide, 1 = show
date=<int>	0, 1	Shows/hides the date. 0 = hide, 1 = show
quad=<int>	0, 1 [1]	Generate a quad image. 0 = normal, 1 = quad
text=<int>	0, 1	Shows/hides the text. 0 = hide, 1 = show
rotation=<int>	0, 90, 180, 270 [1]	Rotates the image clockwise.

showlength=<int>	0, 1	Content-Length is added to the HTTP-header and in the boundary section, between the images. 0 = hidden, 1 = shown.
timeout=<int>	> 0[1]	Set the timeout value (seconds) for the session. If a connection is blocked for this amount of time, the session will be closed by the server.

[1] Product-dependent. Check the product specification.

Example: request a JPEG image from camera 1 with a resolution of 320x240 and compression of 25

```
http://myserver/axis-cgi/jpg/image.cgi?resolution=320x240&camera=1&compression=25
```

5.2.3 JPEG image response

When a JPEG image is requested, the server either returns the specified JPEG file, or an error.

An optional field "Content-Length" header entry specifying the image size in bytes <*image size*> may also be included if the camera or video server is configured to include it. The other optional field "Content-Auth" is followed by authorization-specific data <*authorization information*>, e.g., the encryption method being used.

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: image/jpeg\r\n
[ Content-Length: <image size>\r\n ]
[ Content-Auth: <authorization information>\r\n ]
\r\n
<JPEG image data>\r\n
```

Example: requested JPEG image

```
HTTP/1.0 200 OK\r\n
Content-Type: image/jpeg\r\n
Content-Length: 15656\r\n
\r\n
<JPEG image data>\r\n
```

5.2.4 JPEG buffer request

Request for controlling image buffers via HTTP.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/buffer/command.cgi?<parameter>=<value>
[&<parameter>=<value>...]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
do=<string>	start, stop, get, reset	"start" will create a new image buffer. "stop" will make the image buffer store post alarm images. Buffer will stop after all post alarm images have been taken. "get" is used to fetch an image from the image buffer. "reset" will remove the image buffer, including any buffered images.
buffername=<string>	*<any string>*	Name used for identifying the buffer.
uri=<string>	*<any string>*	Corresponding image URI to be used by the image buffer. **Note:** Must be URI-encoded. The URI should also begin with "ftp://", otherwise an HTTP header is added to the image.
prealarm=<int>	0, ...	Number of images to be stored in the pre alarm buffer.

postalarm=<int>	0, ...	Number of images to be saved after an alarm occurs.
predelay=<int>	*<milliseconds>*	The preferred time between the pre alarm images in milliseconds. Default is one second, 1000 milliseconds.
postdelay=<int>	*<milliseconds>*	The preferred time between the post alarm images in milliseconds. Default value is equal to the value of predelay. I.e. 1000 milliseconds if not specified, and for example 3000 milliseconds, if predelay is set to that.
index=<int>	*<image number>*	The index of image in buffer.

Example 1: Create an image buffer, named DOOR1, with 10 pre alarm images and 15 post alarm images.

```
http://myserver/axis-cgi/buffer/command.cgi?do=start&buffername=DOOR1
&prealarm=10&postalarm=15&&uri=ftp://jpg/1/image.jpg
```

Example 2: Stop a buffer.

```
http://myserver/axis-cgi/buffer/command.cgi?do=stop&buffername=DOOR1
```

Example 3: Get images from a buffer.

```
http://myserver/axis-cgi/buffer/command.cgi?do=get&buffername=DOOR1&index=1
```

5.2.5 MJPG video request

Request Multipart JPEG image.

Method: GET

Syntax:

```
http://<servername>/mjpg[/<camera>]/<name>.mjpg
```

with the following parameters

Parameter	Values	Description
<camera>	1, ... [1]	Select input source. Applies only to servers with more than one input source. *Default:* default camera
<name>	video, quad[1], halfsize[2], fullsize[2], hugesize[2]	"video" returns a multipart image stream with the default resolution and compression, as defined in the system configuration. The *<camera>* option is not allowed in quad request.

[1] Product-dependent. Check the product specification.

[2] Obsolete.

Example: request JPEG image stream from the 2nd camera with default resolution and compression

```
http://myserver/mjpg/2/video.mjpg
```

5.2.6 MJPG video CGI request

Request a Multipart-JPEG image stream (video) with specified properties.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/mjpg/video.cgi
[?<parameter>=<value>[&<parameter>=<value>...]]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
resolution=<int>x<int>	*<width>*[1],*<height>*[1]	Specify the resolution as *<width>* times *<height>* number of pixels of the returned image.
camera=<int>	1, ... [1]	Applies only to video servers with more than one video input. Selects the source camera. If omitted, the default camera is used.
compression=<int>	0 - 100 [1]	Adjusts the compression level of the image. Higher values correspond to higher compression, i.e. lower quality and smaller image size. **Note:** This value is internally mapped and is therefore product-dependent.
colorlevel=<int>	0 - 100 [1]	Sets level of color or grayscale. 0 = grayscale, 100 = full color. **Note:** This value is internally mapped and is therefore product-dependent.
clock=<int>	0, 1	Shows/hides the time stamp. 0 = hide, 1 = show
date=<int>	0, 1	Shows/hides the date. 0 = hide, 1 = show
quad=<int>	0, 1 [1]	Generate a quad image. 0 = normal, 1 = quad
text=<int>	0, 1	Shows/hides the text. 0 = hide, 1 = show
rotation=<int>	0, 90, 180, 270 [1]	Rotates the image clockwise.

showlength=<int>	0, 1	Content-Length is added to the HTTP-header and in the boundary section, between the images. 0 = hidden, 1 = shown.
duration=<int>	0, ...	Specifies for how many seconds the video will be generated and pushed to the client.
nbrofframes=<int>	1, ...	Specifies how many frames the server will generate and push.
req_fps|des_fps=<int>	1, ...	Using either req_fps or des_fps (these cannot be used at the same time) it is possible to specify the frame rate from the server. req_fps = required FPS des_fps = desired FPS Required FPS has higher priority than desired FPS if these are used in two different requests simultaneously. Use req_fps for streams with high priority and des_fps for less important streams.
deltatime=<int>	0, 1	Timediff, specifying the time between the images in ms, is added to the boundary section between the images. 0 = hidden, 1 = shown.
timeout=<int>	> 0[1]	Set the timeout value (seconds) of the session. If a connection is blocked for this length of time, the session will be closed by the server.

[1] Product-dependent. Check the product specification.

Example: a Multipart JPEG image stream from camera 1 with a resolution of 320x240 and compression of 25

```
http://myserver/axis-cgi/mjpg/video.cgi?resolution=320x240&camera=1
```

```
&compression=25
```

Example: a Multipart JPEG image stream from camera 1 with a required frame rate of 5

```
http://myserver/axis-cgi/mjpg/video.cgi?req_fps=5
```

5.2.7 MJPG video response

When MJPG video is requested, the server returns a continuous flow of JPEG files. The content type is "multipart/x-mixed-replace" and each image ends with a boundary string <*boundary*>. The returned image and HTTP data is equal to the request for a single JPEG image.

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=<boundary>\r\n
\r\n
--<boundary>\r\n
<image>
```

where the proposed <*boundary*> is

```
myboundary
```

and the returned <*image*> field is

```
Content-Type: image/jpeg\r\n
[ Content-Length: <image size>\r\n ]
[ Content-Auth: <authorization information>\r\n ]
\r\n
<JPEG image data>\r\n
--<boundary>\r\n
<image>
```

Example: requested Multipart JPEG image

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=myboundary\r\n
\r\n
```

```
--myboundary\r\n
Content-Type: image/jpeg\r\n
Content-Length: 15656\r\n
\r\n
<JPEG image data>\r\n
--myboundary\r\n
Content-Type: image/jpeg\r\n
Content-Length: 14978\r\n
\r\n
<JPEG image data>\r\n
--myboundary\r\n
Content-Type: image/jpeg\r\n
Content-Length: 15136\r\n
\r\n
<JPEG image data>\r\n
--myboundary\r\n
  .
  .
  .
```

5.3 MPEG-2

The requests specified in the MPEG-2 section are supported by those products that use MPEG-2 encoding.

5.3.1 MPEG-2 video request

Requests an MPEG-2 video stream with specified properties.
Method: GET

Syntax:

```
http://<servername>/axis-cgi/mpeg2/video.cgi[?<parameter>=<value>
[&<parameter>=<value> ...]]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
duration=<int>	0, ...	Specifies for how many seconds the video shall be generated and pushed to the client. If the duration parameter is omitted, an infinite video stream will be provided to the client.
camera=<int>	1, ... [1]	Searched recording should match this camera id.

maxlag=<int>	500,... [1]	A number specifying at which lag (in ms) the server should eliminate the lag by cutting the stream. A lag could occur at the client due to clock differences between the client computer and the server. **Note:** This will be the maximum lag. The real lag may be lower than the specified value. Values less than 500 will be interpreted as 500. If the maxlag parameter is left out, the maximum lag will be determined by the available space in the server.

[1] Product-dependent. Check the product specification.

5.3.2 MPEG-2 video response

When MPEG-2 video is requested, the server either returns the specified video stream, or an error message.

A field "X-Prebuffer-Length" is included in the HTTP header when responding to an MPEG-2 buffer request. This entry specifies the length (in seconds) of the pre-buffer video that is included in the video stream response. This length depends on what is specified in the request, and on a maximum value set by the server. If the requested length is greater than or equal to the maximum value set by the server, the returned pre-buffer length will be equal to the maximum value. If the requested length is shorter than the maximum value, the returned length will be equal to the requested length.

```
HTTP/1.0 200 OK \r\n
Content-type: video/mpeg\r\n
[X-Prebuffer-Length: <floating number in seconds> \r\n]
\r\n
<MPEG stream>\r\n
```

Example:

```
HTTP/1.0 200 OK\r\n
Content-Type: video/mpeg\r\n
\r\n
```

```
<MPEG stream>\r\n
```

5.3.3 MPEG-2 buffer request

Request for pre-buffered MPEG-2 video.
Method: GET

Syntax:

```
http://<servername>/axis-cgi/mpeg2/buffer.cgi?
[?<parameter>=<value>[&<parameter>=<value>...]]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
prebuffer=<int>	0, ...	Number of seconds of video, buffered before the request, to be provided to the client.
duration=<int>	0, ...	Specifies for how many seconds the video will be generated and pushed to the client. If the duration parameter is omitted, an infinite video stream will be provided to the client.

5.3.4 MPEG-2 multicast info request

The requests specified in this section can be used with MPEG-2 products that support multicasting and which have this support enabled.

Method: GET

Syntax:

```
http://<servername>/view/sdp/guide.sdp

http://<servername>/view/sdp/guide.axsdp
```

```
http://<servername>/view/sdp/guide.sdpx
```

5.3.5 MPEG-2 multicast info response

The response to an MPEG-2 multicast info request is a text file describing the characteristics of the stream elements and where they can be retrieved.
The description text is in SDP format and follows RFC 2327.

All requests return the same description file. The responses from guide.sdp and guide.axsdp have "application/sdp" as content type, but using guide.sdp is preferred practice. The response from guide.sdpx has "text/plain" as content type.

Return: A successful guide.sdp or guide.axsdp request

```
HTTP/1.0 200 OK\r\n
Date: <date>
Server: <web server>
Content-Length: <length>
Last-Modified: <date>
Content-Type: application/sdp\r\n
\r\n
\r\n
<text according to RFC 2327>
```

Return: A successful guide.sdpx request

```
HTTP/1.0 200 OK\r\n
Date: <date>
Server: <web server>
Content-Length: <length>
Last-Modified: <date>
Content-Type: text/plain\r\n
\r\n
\r\n
<text according to RFC 2327>
```

Example: Requested MPEG-2 media stream session info with guide.sdp from an MPEG-2 video product

```
HTTP/1.0 200 OK
Date: Thu, 28 Nov 2002 13:16:13 GMT
Server: Boa/0.92o
Content-Length: 777
```

```
Last-Modified: Thu, 28 Nov 2002 13:12:24 GMT
Content-Type: application/sdp

v=0
o=root 1038489142 1038489142 IN IP4 10.13.9.123
s=AXIS 250S MPEG-2 Video Server RTP Stream from 10.13.9.123
b=AS:4000
t=0 0
a=type:broadcast
a=tool:AXIS 250S MPEG-2 Video Server 3.10
m=video 5000 RTP/AVP 32
c=IN IP4 239.192.182.161/5
a=x-dimensions:720,576
a=framerate:25
a=x-iptv-svr:video 10.13.9.123 file 1 loop
m=audio 5000 RTP/AVP 14
c=IN IP4 239.192.182.94/5
a=x-bitrate:128
a=x-samplerate:44100
a=x-iptv-svr:audio 10.13.9.123 file 1 loop
v=0
o=root 1038489143 1038489143 IN IP4 10.13.9.123
s=AXIS 250S MPEG-2 Video Server RTP Stream No Audio from 10.13.9.123
b=AS:4000
t=0 0
a=type:broadcast
a=tool:AXIS 250S MPEG-2 Video Server 3.10
m=video 5000 RTP/AVP 32
c=IN IP4 239.192.182.161/5
a=x-dimensions:720,576
a=framerate:25
a=x-iptv-svr:video 10.13.9.123 file 1 loop
```

5.4 PTZ

The requests specified in the PTZ section are supported by those video products that have support for Pan/Tilt/Zoom devices.

5.4.1 PTZ control

To control the Pan, Tilt, and Zoom behavior of a PTZ unit, the following PTZ control URL is used.

Important: The PTZ control is device-dependent. Please check the specification of the Axis PTZ driver you intend using, for information about supported parameters and actual parameter values. The following table is only an overview.

Method: GET/POST

Syntax:

```
http://<servername>/axis-cgi/com/ptz.cgi?<parameter>=<value>[&
<parameter>=<value>... ]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
camera=<int>	1, ... [1]	Applies only to video servers with more than one video input. Selects the source camera. If omitted, the default camera is used.
whoami=<string>	*<any value>*	Returns the name of the system-configured device driver.

center=<int>,<int>
extrem[3]=<int>,<int>
	<x>,*<y>*	**Absolute:** Used to send the coordinates for the point in the image where the user clicked. This information is then used by the server to calculate the pan/tilt move required to (approximately) center the clicked point.
		Relative: Used to send the coordinates for the point in the image where the user clicked. This information is then used by the server to calculate the direction and number of degrees to move. The number of degrees increases with the distance from the center of the image to the point clicked.

imagewidth=<int>	1, ...[1]	Required in conjunction with *center* if the image width is displayed different from the default size of the image, which is product-specific.
imageheight=<int>	1, ...[1]	Needed in conjunction with *center* if the image height is different from the default size, which is product-specific.
move=<string>	home, up, down, left, right, upleft, upright, downleft, downright	**Absolute:** Moves the device 5 degrees in the specified direction. **Relative:** Moves the device approx. 50-90 degrees[2] in the specified direction.

pan=<float>	-180.0 - 180.0	**Absolute:** Pans the device relative to the (0,0) position. **Relative:** N/A
tilt=<float>	-180.0 - 180.0	**Absolute:** Tilts the device relative to the (0,0) position. **Relative:** n/a
zoom=<int>	1 - 9999	**Absolute:** Zooms the device *n* steps. **Relative:** n/a
focus=<int>	1 - 9999	**Absolute:** Move Focus *n* steps. **Relative:** n/a
iris=<int>	1 - 9999	**Absolute:** Move iris to *n* steps. **Relative:** n/a
rpan=<float>	-360.0 - 360.0	**Absolute:** Pans the device *n* degrees relative to the current position. **Relative:** Pans the device approx. *n* degrees relative to the current position.
rtilt=<float>	-360.0 - 360.0	**Absolute:** Tilts the device *n* degrees relative to the current position. **Relative:** Tilts the device approx. *n* degrees relative to the current position.
rzoom=<int>	-9999 - 9999	**Absolute:** Zooms the device *n* steps relative to the current position. Positive values mean zoom in, negative values mean zoom out. **Relative:** Zooms the device approx. *n* steps relative to the current position. Positive values mean zoom in, negative values mean zoom out.
rfocus=<int>	-9999 - 9999	**Absolute:** Move Focus *n* steps relative to the current position. Positive values mean focus near, negative values mean focus far.

		Relative: Move Focus approx. *n* steps relative to the current position. Positive values mean focus near, negative values mean focus far.
riris=<int>	-9999 - 9999	**Absolute:** Move iris *n* steps relative to the current position. Positive values mean open iris, negative values mean close iris. **Relative:** Move iris approx. *n* steps relative to the current position. Positive values mean open iris, negative values mean close iris.
zoomrel[3]=<string>	tele, wide, telemax, telemin	Adjusts the zoom gradually.
focusrel[3]=<string>	far, farmore, near, nearmore	Adjusts the focus gradually.
irisrel[3]=<string>	open, close, openmore, closemore	Adjusts the iris gradually.
autofocus=<string>	on, off	Autofocus On/Off.
autoiris=<string>	on, off	Autoiris On/Off.
continuouspantiltmove= <int>,<int>	-100 - 100, -100 - 100	Continuous pan/tilt motion. Positive values mean right (pan) and up (tilt), negative values mean left (pan) and down (tilt). "0,0" means stop. Values as *<pan speed>*,*<tilt speed>*
continuouszoommove= <int>	-100 - 100	Continuous zoom motion. Positive values mean zoom in and negative values mean zoom out. "0" means stop.
continuousfocusmove= <int>	-100 - 100	Continuous focus motion. Positive values mean focus near and negative values mean focus far. "0" means stop.

continuousirismove=<int>	-100 - 100	Continuous iris motion. Positive values mean iris open and negative values mean iris close. "0" means stop.
auxiliary=<string>	*<function name>*	Activates/deactivates auxiliary functions of the device where <*function name*> is the name of the device-specific function.
setserverpresetname=<string>	*<preset name>*[4]	Associates the current position to <*preset name*> as a preset position in the server.
setserverpresetno=<int>	1, ...	Saves the current position as a preset position number in the server.
removeserverpresetname=<string>	*<preset name>*[4]	Removes the specified preset position associated with <*preset name*>.
removeserverpresetno=<int>	1, ...	Removes the specified preset position.
gotoserverpresetname=<string>	*<preset name>*[4]	Move to the position associated with the <*preset name*>.
gotoserverpresetno=<int>	1, ...	Move to the position associated with the specified preset position number.
setdevicepreset=<int>	*<preset pos>*	Bypasses the presetpos interface and tells the device to save its current position as preset position <*preset pos*> directly in the device, where <*preset pos*> is a device-specific preset position number.
gotodevicepreset=<int>	*<preset pos>*	Bypasses the presetpos interface and tells the device to go directly to the preset position number <*preset pos*> stored in the device, where the <*preset pos*> is a device-specific preset position number.
barcoord=<int>,<int>	*<x>*,*<y>*	Used in conjunction with panbar, tiltbar, zoombar, focusbar or irisbar, to send coordinates to the server.
panbar=<int>,<string>	*<length>*, *<alignment>*	<*length*> is the length of the bar in pixels, which is needed in order to calculate the center of the bar.

		<alignment> is one of the strings "*horisontal*" or "*vertical*". The alignment string determines if the x (horizontal) or the y (vertical) coordinate from *barcoord* is used, i.e. if the bar is horizontal; use "*horizontal*" and if the bar is vertical; use *"vertical*" as alignment.
tiltbar=<int>,<string>	*<length>*, *<alignment>*	*<length>* is the length of the bar in pixels, which is needed in order to calculate the center of the bar. *<alignment>* is one of the strings "*horisontal*" or "*vertical*". The alignment string determines if the x (horizontal) or the y (vertical) coordinate from *barcoord* is used, i.e. if the bar is horizontal; use "*horizontal*" and if the bar is vertical; use *"vertical*" as alignment.
zoombar=<int>,<string>	*<length>*, *<alignment>*	*<length>* is the length of the bar in pixels, which is needed in order to calculate the center of the bar. *<alignment>* is one of the strings "*horisontal*" or "*vertical*". The alignment string determines if the x (horizontal) or the y (vertical) coordinate from *barcoord* is used, i.e. if the bar is horizontal; use "*horizontal*" and if the bar is vertical; use *"vertical*" as alignment.
focusbar=<int>,<string>	*<length>*, *<alignment>*	*<length>* is the length of the bar in pixels, which is needed in order to calculate the center of the bar. *<alignment>* is one of the strings "*horisontal*" or "*vertical*". The alignment string determines if the x (horizontal) or the y (vertical) coordinate from *barcoord* is used, i.e. if the bar is horizontal; use "*horizontal*" and if the bar is vertical; use

		"vertical" as alignment.
irisbar=<int>,<string>	*<length>*, *<alignment>*	*<length>* is the length of the bar in pixels, which is needed in order to calculate the center of the bar. *<alignment>* is one of the strings "*horisontal*" or "*vertical*". The alignment string determines if the x (horizontal) or the y (vertical) coordinate from *barcoord* is used, i.e. if the bar is horizontal; use "*horizontal*" and if the bar is vertical; use *"vertical"* as alignment.
speed=<int>	1 - 100	Sets the head speed of the device connected to the specified camera.
query=<string>	speed, position, presetposcam, presetposall	Returns the current parameter values. See 5.1.2 for more information about server responses.
info=<int>	1	Returns a description of available PTZ commands. No PTZ control is performed.

[1] Product-dependent. Check the product specification.

[2] Actual values are device driver-specific.

[3] Obsolete.

[4] *<preset name>* is a string with a maximum of 31 characters, ~ is not allowed.

Example: Request information about which PTZ commands are available for camera 3.

```
http://myserver/axis-cgi/com/ptz.cgi?info=1&camera=3
```

5.5 Audio

The requests specified in the Audio section are supported by those video products that have audio capability.

5.5.1 Multipart audio data request

Request a Multipart Audio stream.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/audio/receive.cgi
```

There are no valid parameters and values

Example: a Multipart Audio stream

```
http://myserver/axis-cgi/audio/receive.cgi
```

5.5.2 Multipart audio data response

When an audio stream is requested/transmitted, the server returns/receives a continuous flow of audio packets. The content type is "multipart/x-mixed-replace" and each audio packet ends with a boundary string *<boundary>*. The MIME type used for the audio transmitted is audio/32KADPCM. The message body contains a block of binary data.

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=<boundary>\r\n
\r\n
--<boundary>\r\n
<audio>
```

where the proposed *<boundary>* is

```
myboundary
```

and the *<audio>* field is

```
Content-Type: audio/32KADPCM\r\n
\r\n
<Audio data>\r\n
--<boundary>\r\n
<audio>
```

Example: Multipart Audio data

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=myboundary\r\n
\r\n
--myboundary\r\n
Content-Type: audio/32KADPCM\r\n
\r\n
<Audio data>\r\n
--myboundary\r\n
Content-Type: audio/32KADPCM\r\n
\r\n
<Audio data>\r\n
--myboundary\r\n
Content-Type: audio/32KADPCM\r\n
\r\n
<Audio data>\r\n
--myboundary\r\n
```

5.5.3 Multipart audio data transmit
Transmits a Multipart Audio data stream.

Method: POST

Syntax:

```
http://<servername>/axis-cgi/audio/transmit.cgi
```
There are no valid parameters and values.

Example: transmit a Multipart Audio stream

```
http://myserver/axis-cgi/audio/transmit.cgi
```

5.5.4 Get audio configuration parameters
Get information about audio configuration parameters.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/audio/getparam.cgi
```
Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
<audio parameters>
```

and *<audio parameters>* is

```
root.Audiod.duplexMode=<duplexMode>\n
root.Audiod.maxNumberOfClients=<maxNumberOfClients>\n
root.Audiod.forwardOnPost=<forwardOnPost>\n
root.Audiod.buttonMode=<buttonMode>\n
root.Audiod.acousticEcho=<acousticEcho>\n
root.Audiod.lineEcho=<lineEcho>\n
root.Audiod.slopeFilter=<slopeFilter>\n
root.Audiod.noisecancelIOgain=<noicecancelIOgain>\n
root.Audiod.noisecancelAttenuation=<noisecancelAttenuation>\n
root.Audiod.audioEncoding=<audioEncoding>\n
root.Audiod.httpMessageType=<httpMessageType>\n
root.Audiod.microphoneMode=<microphoneMode>\n
root.Audiod.speakerMode=<speakerMode>\n
root.Audiod.forceMicrophoneMode=<forceMicrophoneMode>\n
root.Audiod.forceSpeakerMode=<forceSpeakerMode>\n
root.Audiod.bufferLevel=<bufferLevel>\n
root.Audiod.forceBufferLevel=<forceBufferLevel>\n
\n
```

Example:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
root.Audiod.duplexMode=<duplexMode>\n
root.Audiod.maxNumberOfClients=<maxNumberOfClients>\n
root.Audiod.forwardOnPost=<forwardOnPost>\n
root.Audiod.buttonMode=<buttonMode>\n
root.Audiod.acousticEcho=<acousticEcho>\n
root.Audiod.lineEcho=<lineEcho>\n
root.Audiod.slopeFilter=<slopeFilter>\n
root.Audiod.noisecancelIOgain=<noicecancelIOgain>\n
root.Audiod.noisecancelAttenuation=<noisecancelAttenuation>\n
root.Audiod.audioEncoding=<audioEncoding>\n
root.Audiod.httpMessageType=<httpMessageType>\n
root.Audiod.microphoneMode=<microphoneMode>\n
root.Audiod.speakerMode=<speakerMode>\n
root.Audiod.forceMicrophoneMode=<forceMicrophoneMode>\n
root.Audiod.forceSpeakerMode=<forceSpeakerMode>\n
root.Audiod.bufferLevel=<bufferLevel>\n
root.Audiod.forceBufferLevel=<forceBufferLevel>\n
\n
```

5.6 Motion Detection

The requests specified in the Motion Detection section are supported by those video products that have built-in motion detection.

5.6.1 Set motion detection parameters

Adding new motion detection window, removing or updating any existing window.

The "key" entry is a unique window identity communicated in all actions concerning the window, i.e., when adding, updating or removing it, or when reading information about it. The "name" entry is a window identification useful when e.g. communicating with a user in an application or an alarm. The three parameters "size", "sensitivity" and "history" are as generic as possible, *i.e.* they do not depend on the actual motion detection algorithm.

Method: GET/POST

Syntax:

```
http://<servername>/axis-cgi/admin/mdsetparam.cgi?<parameter>=<value>[&
<parameter>=<value>...]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
action=<string>	add, remove, update	"add" creates a new motion detection window and requires both the "key" and "name" entries to be defined. "update" make changes to an existing window and only requires the "key" entry to be defined. "remove" deletes an existing window and only requires the "key" entry to be defined. Any other additional entry is ignored. *Default:* no default value provided.
key=<int>	0, ...[1]	A unique identity to the motion detection window. *Default:* no default value provided.
name=<string>	*<any string>*	A user's window identification, e.g., "Door" or "Window". *Default:* no default value provided.

method=<char>	w	Motion detection method to use for this window. *Default:* w
left=<int>	0 ... 9999	The coordinate for the left boundary of the rectangular motion detection window. The full value range is related to the full image width and *0* is counted from the left hand side of the image. *Default:* 0
right=<int>	0 ... 9999	The coordinate for the right boundary of the rectangular motion detection window. The full value range is related to the full image width and *0* is counted from the left hand side of the image. *Default:* 9999
top=<int>	0 ... 9999	The coordinate for the upper boundary of the rectangular motion detection window. The full value range is related to the full image height and *0* is counted from the upper side of the image. *Default:* 0
bottom=<int>	0 ... 9999	The coordinate for the bottom boundary of the rectangular motion detection window. The full value range is related to the full image height and *0* is counted from the upper side of the image. *Default:* 9999
sensitivity=<int>	0 ... 100	This tunes the "object difference from the background"-sensitivity, *i.e.* difference in terms of

		color and/or structure. A low value detects even very small changes and can e.g. trigger on image noise if set too low. A very high value on the other hand, requires a very dramatic change, with e.g. a dark object appearing in an almost white scene (or vice versa). *Default: method specific*
history=<int>	0 ... 100	This adjusts the motion detection to be more or less sensitive for very fast motion. At very low values motion is detected only when a scene has changed, but rapidly adopts to this changed scene as the new normal scene. Higher values are more conservative and will detect motion in a greater number of pictures before considering this as being a normal object in the scene. *Default: method specific*
size=<int>	0 ... 100	Defines as a percentage the size of the object that will result in motion detection. At a low value, even very small changes will trigger detection, whilst a very large value requires a very large object to trigger detection. *Default: method specific*

Return: all parameters are set correct

```
HTTP/1.0 204 No Content\r\n
```

Return: one or more critical errors occurred

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Error: <error description>\r\n
```

Example: Add a new motion detection window

```
http://myserver/axis-cgi/admin/mdsetparam.cgi?action=add&key=2
&name=Entrance&top=500&bottom=7000&left=5000&right=8500
```

Example: Update parameters for an existing motion detection window

```
http://myserver/axis-cgi/admin/mdsetparam.cgi?action=update&key=2
&top=1500&bottom=8000
```

Example: Remove a motion detection window

```
http://myserver/axis-cgi/admin/mdsetparam.cgi?action=remove&key=2
```

5.6.2 Get motion detection parameters

Read information about current defined motion detection windows.

Method: GET/POST

Syntax:

```
http://<servername>/axis-cgi/motion/mdgetparam.cgi
```

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
<window data>
```

and <*window data*> is

```
key=<key>\r\n
name=<name>\r\n
method=<method>\r\n
left=<left>\r\n
top=<top>\r\n
```

```
right=<right>\r\n
bottom=<bottom>\r\n
sensitivity=<sensitivity>\r\n
history=<history>\r\n
size=<size>\r\n
\r\n
[ <window data> ]
```

Example: two motion detection windows "Backdoor" and "Window"

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
key=0\r\n
name=Backdoor\r\n
method=W\r\n
left=200\r\n
top=800\r\n
right=3600\r\n
bottom=9400\r\n
sensitivity=90\r\n
history=85\r\n
size=10\r\n
\r\n
key=1\r\n
name=Window\r\n
method=W\r\n
left=7000\r\n
top=1000\r\n
right=9500\r\n
bottom=4000\r\n
sensitivity=95\r\n
history=85\r\n
size=25\r\n
\r\n
```

5.6.3 Get motion detection level

To read the current motion detection levels of all windows, the URL stated below is used.

Method: GET/POST

Syntax:

```
http://<servername>/axis-cgi/motion/motiondetect.cgi
```

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=<boundary>\r\n
\r\n
--<boundary>\r\n
<motion levels>
```

where the proposed boundary *<boundary>* is

```
axismdb
```

and the *<motion levels>* part is

```
Content-Type: text/plain\r\n
\r\n
<motion level for window n>
--<boundary>\r\n
```

and *<motion level for window* n>" is

```
key=<key n>;level=<motion level for n>;

threshold=<threshold level for n>;\r\n
[ <motion level for window n+1> ]
```

Example: two motion detection windows with key "0" and key "1"

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=axismdb\r\n
\r\n
--axismdb\r\n
Content-Type: text/plain\r\n
\r\n
key=0;level=28;threshold=45;\r\n
key=1;level=43;threshold=25;\r\n
--axismdb\r\n
Content-Type: text/plain\r\n
\r\n
key=0;level=54;threshold=45;\r\n
key=1;level=38;threshold=25;\r\n
--axismdb\r\n
Content-Type: text/plain\r\n
\r\n
key=0;level=49;threshold=45;\r\n
key=1;level=19;threshold=25;\r\n
--axismdb\r\n
 .
 .
 .
```

5.7 Recording

The requests specified in the Recording section are supported by those products that provide storage of recordings.

5.7.1 List recordings

Choose to list all recordings, one specific recording, or all recordings matching a criteria.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/view/reclist.cgi[?<parameter>=<value>
[&<parameter>=<value> ...]]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
recordingid=<int>	> 0	Recording id, unique number for each recording. Only this recording will be listed.
camera=<int>	1, ... [1]	List only recordings that match this camera id.
event=<int>	> 0	List only recordings that match this event id. Id of an event can be found in the event parameters.
time=<int>	Seconds since 1970: 1/100 seconds	List only recordings made during this time.
timespan=<int>	> 0	List only recordings made during the time span from "time" until "time" + "timespan". *Requires:* parameter "time"
info=<string>	count	Begin the server response with a count of the number of listed recordings.

[1] Product-dependent. Check the product specification.

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
<info data>1
<recording data>
```

[1] Only if parameter "info=count" is used.

and *<info data>* is

```
numberofrecordings=<count>\r\n
\r\n
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
recordingid=<int>	> 1	Recording id, unique number for every recording.
camera=<int>	1, ... [1]	ID of camera recorded from.

cameraname=<string>	user specified	Name of the camera recorded from.
eventid=<int>	user specified	Event ID
eventpath=<string>	parameter path	Parameter path to event parameters. *Example:* root.Event.E7
eventname=<string>	user specified	Name of recorded event.
prebufferstartframe= <int>:<int>	frame id	Startframe is the first image/frame of the recording. If the beginning is deleted by reduction the startframe is moved to the new start. *Example:* 785421:57
prebufferstarttime= <int>:<int>	Seconds since 1970: 1/100 seconds	Time of the startframe. *Example:* 983453412:10
eventstartframe= <int>:<int>	frame id	The image/frame when the event was trigged. *Example:* 785421:57
eventstarttime= <int>:<int>	Seconds since 1970: 1/100 seconds	Time of the eventstartframe. *Example:* 983453412:10
eventstopframe= <int>:<int>	frame id	The image/frame when the event ended. *Example:* 785421:57
eventstoptime= <int>:<int>	Seconds since 1970: 1/100 seconds	Time of the eventstopframe. *Example:* 983453412:10
postbufferstopframe= <int>:<int>	frame id	The last image/frame of recording. *Example:* 785421:57

postbufferstoptime=<int>:<int>	Seconds since 1970: 1/100 seconds	Time of the postbufferstopframe. *Example:* 983453412:10
priority=<int>	0 - 100 [1]	Priority of the event. 100 = highest priority. **Note**: This value is internally mapped and is therefore product-dependent.
ongoing=<string>	yes, no	Recording in progress or not.
secured=<string>	yes, no	A secured (preserved) recording will not be reduced or deleted. See Preserve and release recordings
startquality=<int>	0 - 100 [1]	Quality of the beginning of the recording, due to reduction. 100 = highest quality. **Note**: This value is internally mapped and is therefore product-dependent.
stopquality=<int>	0 - 100 [1]	Quality of the beginning of the recording, due to reduction. 100 = highest quality. **Note**: This value is internally mapped and is therefore product-dependent.

[1] Product-dependent. Check the product specification.

Example 1: List recording id 12:

Request:

```
http://myserver/axis-cgi/view/reclist.cgi?recordingid=12
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
recordingid=12\r\n
cameraid=1\r\n
cameraname=backdoor\r\n
eventid=2\r\n
eventpath=root.Event.E2\r\n
eventname=alarm\r\n
prebufferstartframe=456213:00\r\n
prebufferstarttime=98992000:00\r\n
eventstartframe=456345:00\r\n
eventstarttime=98992501:00\r\n
eventstopframe=456287:34\r\n
eventstoptime=98992500:34\r\n
postbufferstopframe=456288:34\r\n
postbufferstoptime=98992501:34\r\n
priority=2\r\n
ongoing=no\r\n
secured=no\r\n
startquality=3\r\n
stopquality=4\r\n
\r\n
```

Example 2: List recordings made during timespan Jul 4 2002 12:00:00 - 13:00:00 CET with count info:

Request:

```
http://myserver/axis-cgi/view/reclist.cgi?time
=1025776800&timespan=3600&info=count
```

Response: (edited)

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
numberofrecordings=3\r\n
\r\n
recordingid=245\r\n
...
eventstarttime=1025754065:43\r\n
...
eventstoptime=1025779423:32\r\n
...
\r\n
recordingid=356\r\n
...
eventstarttime=1025776780:32\r\n
...
eventstoptime=1025776850:02\r\n
...
```

```
\r\n
recordingid=357\r\n
...
eventstarttime=1025778305:00\r\n
...
eventstoptime=1025778401:34\r\n
...
\r\n
```

Example 3: Search and list recordings of event 6 from camera 1:

```
http://myserver/axis-cgi/view/reclist.cgi?camera=1&event=6
```

Example 4: List all available recordings with count info:

```
http://myserver/axis-cgi/view/reclist.cgi?info=count
```

5.7.2 Play recordings

With this interface, recorded images can be replayed from the server, either single JPG images or MJPG video streams.
Method: GET

Syntax:

```
http://<servername>/axis-cgi/view/player.cgi[?<parameter>=<value>

[&<parameter>=<value> ...]]
```

Parameters and values listed below.
For greater efficiency when playing back and forward, stepping etc., the player uses *recording id* and *session id*.
Recording id is received by list recording.
Use the recording id in the first player call, and then the session id returned in every player response.

Example:

AXIS Server ->	<- Client
List all recordings	

http://myserver/axis-cgi/view/reclist.cgi	
	Server response containing info about all recordings
Play recording 24	
http://myserver/axis-cgi/view/player.cgi?recordingid=24	
	Server response containing sessionid=3 and images.
Play backward	
http://myserver/axis-cgi/view/player.cgi?sessionid=3& direction=backward	
	Server response containing sessionid=3 and images.

with the following parameters and values

Parameter/Syntax	**Values**	**Description**
recordingid=<int>	> 0 [1]	The id of the recording to play.
sessionid=<int>	0 if no session, else > 0 [1]	Id of this session. Returned in previous server response. Used for faster database search.
camera	1, ... [2]	Play from this camera.
info=<int>	1	Returns a description of this CGI-request. No image generated.
starttime=<int>:<int>	Seconds since 1970: 1/100 seconds [3]	Play only a part of the recording. Start at this time. *Example:* 983453412:10
stoptime=<int>:<int>	Seconds since 1970: 1/100 seconds [3]	Play only a part of the recording. Stop at this time. *Example:* 983453412:10
startframe=<int>:<int>	frame id [3]	Play only a part of the recording. Start at this frame.

		Example: 785421:57
stopframe=<int>:<int>	frame id[3]	Play only a part of the recording. Stop at this frame. *Example:* 785421:57
time=<int>:<int>	Seconds since 1970: 1/100 seconds [4]	Get a single image closest to this time. *Example:* 983453412:10
frame=<int>:<int>	frame id [4]	Get a single image closest to this frame id. *Example:* 785421:57
direction=<string>	forward, backward	Play recording forwards or backwards. *Default:* forward
step=<int>	*MJPG video stream* > 0	*MJPG video stream* Play only every *n*:th frame. Can be used to "fast forward" a recording. *Default:* 1 (play every frame)
	JPG single image >= 0	*JPG single image* Get the *n*:th frame in the given direction from the specified frame. *Default:* 0 (get the specified frame)
session=<string>	close	Closes the session and frees server resources for a new session. If a session is not closed it will eventually timeout and be automatically closed. *Requires:* sessionid

timeout=<int>	> 0	Set the timeout value (in seconds) of the player session. The session will be automatically closed if an open stream is blocked for this amount of time the client has not made a new request in this amount of time after the last stream was closed. *Default:* 150
keepalive=<string>	yes, no	Push search progress information in the stream prior to the first frame. This is useful as user feedback and also prevents client software or connection from timing out if the recording lookup (when using start/stop time/frame) takes a long time. *Default:* no

[1] Request must contain one of these. See example below.

[2] Product-dependent. Check the product specification.

[3] Use either *starttime*/*stoptime* or *startframe*/*stopframe*. Do not combine time and frame formats in same request.

[4] Use either *time* or *frame* to get a JPG single image.

If time is specified, the first subsection found that matches the time (within the specified recording, if such is specified) will be played. Since there can be several subsections that matches the same time (due to changes in server time) there is no way of knowing which was requested, hence the first match is always played.
Frame ID is used to directly identify a unique video frame from a specific camera. This is independent of different recordings and changes in server time. The scale of the Frame ID is linear to real time.

Example 1: Play recording 5

```
http://myserver/axis-cgi/view/player.cgi?recordingid=5
```

Example 2: Play part of recording backward - (sessionid=3 was returned in previous server response)

```
http://myserver/axis-cgi/view/player.cgi?sessionid=3&startframe=45347:00
&stopframe=47023:00&direction=backward&keepalive=yes
```

Example 3: Fast forward part of recording - (sessionid=3 was returned in previous server response)

```
http://myserver/axis-cgi/view/player.cgi?sessionid=3&startframe=45347:00
&stopframe=47023:00&step=4&keepalive=yes
```

Example 4: Get single JPG image of specific time - (sessionid=3 was returned in previous server response)

```
http://myserver/axis-cgi/view/player.cgi?sessionid=3&time=983433518:00&keepalive=yes
```

Example 5: Get next single JPG image from specified frame - (sessionid=3 was returned in previous server response)

```
http://myserver/axis-cgi/view/player.cgi?sessionid=3&frame=45692:14
&direction=forward&step=1&keepalive=yes
```

Example 6: Close session - (sessionid=3 was returned in previous server response)

```
http://myserver/axis-cgi/view/player.cgi?sessionid=3&session=close
```

Example 7: Play time interval - (The first found matching interval for the camera is played)

```
http://myserver/axis-cgi/view/player.cgi?camera=4&starttime=982813227:00
&stoptime=982813720:00&keepalive=yes
```

5.7.3 Recorded JPG/MJPG response

When recorded MJPG video is requested, the server returns a continuous flow of JPEG files. The content type is "multipart/x-mixed-replace" and each image is preceded of data about the image and ends with a boundary string *<boundary>*.

MJPG video stream response:

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=<boundary>\r\n
\r\n
--<boundary>\r\n
<progress>[1]
<image>
```

where the proposed *<boundary>* is

```
myboundary
```

and the *<progress>* field is

```
x-axis-dvr-search-progress-percent: <percent>\r\n
Content-Type: text/plain\r\n
\r\n
searchsourceframe=<frame id>\r\n
searchcurrentframe=<frame id>\r\n
searchdestinationframe=<frame id>\r\n
--<boundary>\r\n
<progress>
```

and the returned *<image>* field is

```
x-axis-dvr-session-id: <sessionid>\r\n
x-axis-dvr-frame-time: <time>\r\n
x-axis-dvr-frame-id: <frameid>\r\n
x-axis-dvr-frame-state: <state>\r\n
Content-Type: image/jpeg\r\n
Content-Length: <image size>\r\n
\r\n
<JPEG image data>\r\n
--<boundary>\r\n
<image>
```

[1] Only if parameter "keepalive=yes" is used and the search for the start frame takes more than a few seconds. Progress fields are pushed with a few seconds interval.

JPG singe image response:

```
HTTP/1.0 200 OK\r\n
\r\n
x-axis-dvr-session-id: <sessionid>\r\n
x-axis-dvr-frame-time: <time>\r\n
x-axis-dvr-frame-id: <frameid>\r\n
x-axis-dvr-frame-state: <state>\r\n
Content-Type: image/jpeg\r\n
```

```
Content-Length: <image size>\r\n
\r\n
<JPEG image data>\r\n
```

Meta tag	Description
x-axis-dvr-search-progress-percent: <int>	Percent (0-100) of search completed.
x-axis-dvr-session-id: <int>	ID of the session.
x-axis-dvr-frame-time: <int>:<int>	Timestamp in string format epoch:hh.
x-axis-dvr-frame-id: <int>:<int>	Frame ID.
x-axis-dvr-frame-state: <int>	Active alarm inputs when this picture was taken, numeric representation of 4-bit field. Alarm inputs 1 - 4 are represented by values 1 (0001), 2 (0010), 4 (0100), 8 (1000). *Example:* State 5 represents inputs 1 and 3 active (1+4). State 15 represents all inputs active (1+2+4+8).

<parameter>=<value>	**Values**	**Description**
searchsourceframe=<int>:<int>	frame id	Frame that search started at. *Example:* 785421:57
searchcurrentframe=<int>:<int>	frame id	Frame that search is currently at. *Example:* 785421:57
searchdestinationframe=<int>:<int>	frame id	Frame that is search for. *Example:* 785421:57

Example: Response to request for part of recording starting at frame 563234:23 with keepalive set.

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=myboundary\r\n
\r\n
--myboundary\r\n
x-axis-dvr-search-progress-percent: 67\r\n
Content-Type: text/plain\r\n
\r\n
searchsourceframe=436799:00\r\n
searchcurrentframe=521510:66\r\n
searchdestinationframe=563234:23\r\n
--myboundary\r\n
x-axis-dvr-search-progress-percent: 100\r\n
Content-Type: text/plain\r\n
\r\n
searchsourceframe=436799:00\r\n
searchcurrentframe=563234:23\r\n
searchdestinationframe=563234:23\r\n
--myboundary\r\n
x-axis-dvr-session-id: 132\r\n
x-axis-dvr-frame-time: 1003445674:23\r\n
x-axis-dvr-frame-id: 563234:23\r\n
x-axis-dvr-frame-state: 3\r\n
Content-Type: image/jpeg\r\n
Content-Length: 15656\r\n
\r\n
<JPEG image data>\r\n
--myboundary\r\n
x-axis-dvr-session-id: 132\r\n
x-axis-dvr-frame-time: 1003445674:54\r\n
x-axis-dvr-frame-id: 563234:54\r\n
x-axis-dvr-frame-state: 3\r\n
Content-Type: image/jpeg\r\n
Content-Length: 14978\r\n
\r\n
<JPEG image data>\r\n
--myboundary\r\n
x-axis-dvr-session-id: 132\r\n
x-axis-dvr-frame-time: 1003445675:21\r\n
x-axis-dvr-frame-id: 563235:21\r\n
x-axis-dvr-frame-state: 3\r\n
Content-Type: image/jpeg\r\n
Content-Length: 15136\r\n
\r\n
<JPEG image data>\r\n
--myboundary\r\n
 .
 .
 .
```

5.7.4 Preserve and release recordings

Note: This request requires Administrator or Operator access.

This request changes the status on a recording from released to preserved or vice versa.
A preserved recording is not affected by quality reduction, but maintains the quality it had when preserved, until released.

Syntax:

```
http://<servername>/axis-cgi/operator/recadmin.cgi?
camera=<camera id>&[preserve|release]=<recid>
```

Example 1: Preserve recording 4234 from camera 2:

```
http://myserver/axis-cgi/operator/recadmin.cgi?camera=2&preserve=4234
```

Example 2: Release recording 243592 from camera 1:

```
http://myserver/axis-cgi/operator/recadmin.cgi?camera=1&release=243592
```

5.7.5 Controlling recordings

This interface is used to start and stop recordings from a camera.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/view/record.cgi[?<parameter>=<value>
[&<parameter>=<value> ...]]
```

with the following parameters and values:

<parameter>=<value>	**Values**	**Description**

do=<string>[1]	start, stop	Start or stop recording.
camera=<int>[1]	1, ... [2]	Selects a camera.

[1] Request must contain these. See example below.

[2] Product-dependent. Check the product specification.

Example 1: Start recording from camera 1

```
http://myserver/axis-cgi/view/record.cgi?do=start&camera=1
```

Example 2: Stop recording from camera 2

```
http://myserver/axis-cgi/view/record.cgi?do=stop&camera=2
```

5.8 I/O

The requests specified in the I/O section are supported by those products that have Input/Output connectors.

5.8.1 I/O control

5.8.1.1 Input

Input

Method: GET

Syntax:

```
http://<servername>/axis-cgi/io/input.cgi?<parameter>=<value>
[&<parameter>=<value>...]
```

with the following parameters and values

***<parameter>*=**	**Values**	**Description**

<value>		
check=<int>[,<int>, ...]	*<id1>*[,*<id2>*, ...] [1]	Returns the status of one or more inputs numbered *id1 ,id2, ...*. See 5.2.1 for more information about server responses.
monitor=<int>[, <int>, ...] [2]	*<id1>*[,*<id2>*, ...] [1]	Returns a multipart stream of "check" inputs (see return description below).

[1] Number of inputs may differ for different camera and video servers. See product specification.

[2] Support for this parameter is product/release-dependent.

Return: "*monitor*", i.e., multipart "*check*" parameter

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=<boundary>\r\n
\r\n
--<boundary>\r\n
<monitor data>
```

where the proposed boundary *<boundary>* is

```
ioboundary
```

and the *<monitor data>* part is

```
Content-Type: text/plain\r\n
\r\n
<check data>
--<boundary>\r\n
```

and *<check data>* is

```
IO<n>:<action char>\r\n
```

and *<n>* is the I/O port number and *<action char>* is the action character described in the table above.
Note: The output can contain extra blank lines, i.e., extra `\r\n` within the sections.

Example: Monitor data on input ports 1, 2, 3, and 4

```
http://myserver/axis-cgi/io/input.cgi?monitor=1,2,3,4
```

Example: Monitor data on input port 1

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace; boundary=ioboundary\r\n
\r\n
\r\n
\r\n
\r\n
--ioboundary\r\n
Content-Type: text/plain\r\n
\r\n
IO0:/\n
\r\n
\r\n
--ioboundary\r\n
Content-Type: text/plain\r\n
\r\n
IO0:H\n
\r\n
--ioboundary\r\n
Content-Type: text/plain\r\n
\r\n
\r\n
IO0:\\n
\r\n
\r\n
--ioboundary\r\n
Content-Type: text/plain\r\n
\r\n
\r\n
\r\n
\r\n
--ioboundary\r\n
Content-Type: text/plain\r\n
\r\n
\r\n
  .
  .
  .
```

5.8.1.2 Output

Output

Method: GET

Syntax:

```
http://<servername>/axis-cgi/io/output.cgi?<parameter>=<value>
[&<parameter>=<value>...]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
check=<int>[,<int>, ...]	*<id1>*[,*<id2>*, ...] [1]	Returns the status of one or more outputs numbered *id1 ,id2, ...* See 5.2.1 for more information about server responses.
monitor=<int>[,<int>, ...] [2]	*<id1>*[,*<id2>*, ...] [1]	Returns a multipart stream of "check" outputs (see return description below).
action=<string>	[*<id>* [1]]:*<a>*[*<wait>* *<a>* ...]	Sets the output relay *<id>* on or off and waits *<wait>* milliseconds. *<id>* = Output number. If omitted, output *1* is selected. *<a>* = Action character: / or \\ / = on, \\ = off. *<wait>* = Delay in milliseconds.

[1] Number of outputs may differ for different camera and video servers. See product specification.

[2] Support for this parameter is product/release-dependent.

Example: Set output 1 on

```
http://myserver/axis-cgi/io/output.cgi?action=1:/
```

Example: Set two 300 ms pulses with 500 ms delay between the pulses on output 1

```
http://myserver/axis-cgi/io/output.cgi?action=1:/300\500/300\
```

Example: Wait 1 second before setting output 1 on

```
http://myserver/axis-cgi/io/output.cgi?action=1:1000/
```

5.8.2 Virtual I/O control

5.8.2.1 Input

Input

Method: GET

Syntax:

```
http://<servername>/axis-cgi/io/virtualinput.cgi?<parameter>=<value>
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
action=<string>	[<*id*>]:<*a*>	Sets the virtual input <*id*> on or off. <*id*> = Input number. If omitted, input *1* is selected. <*a*> = Action character: / or \ / = on, \ = off.

Example: Set virtual input 1 on

```
http://myserver/axis-cgi/io/virtualinput.cgi?action=1:/
```

5.9 Serial Port

The requests specified in the Serial Port section are supported by those video products that have implemented a Generic driver.

5.9.1 Serial port control

Control serial port

Method: GET/POST

Syntax:

```
http://<servername>/axis-cgi/com/serial.cgi?<parameter>=<value>
[&<parameter>=<value>... ]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
port=<int>	1, ... [1]	The COM port is selected with this parameter.

write=<string>
dataout[2]=<string>
	<bytestring>	*<bytestring>*: hex coded bytes with values of {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, a, b, c, d, e, f} Writes the specified data string to the selected serial port. Max string length: 128 bytes [1] .

writestring=<string>	*<url encoded string>*	Writes the URL-encoded string to the selected serial port. Max string length: 128 bytes [1].
read=<int>	1, ...	Reads *n* bytes from the selected serial port. The returned data will be hexadecimal coded and placed between #s (e.g. #3A#)
wait=<int>	1 - 9	Specified in seconds. Used together with the "read"

		parameter. A read is terminated when the specified number of bytes is read or when the wait period has ended.
timeout=<int>	1 - 9000	Specified in milliseconds. Used together with the "read" parameter. A read is terminated when the specified number of bytes is read or the timeout has expired.

[1] Product-dependent. Check the product specification.

[2] Obsolete.

5.9.2 Open serial port

This CGI makes it possible to open the serial port using the HTTP protocol. Authentication is handled by the Web server.

After an initial connect command from the client, the connection is kept alive until the client closes it.

Several clients may be connected concurrently to the same serial port.

After the connection has been set up, data sent from the client to the Web server is forwarded to the serial port, and incoming serial data is returned to all currently connected clients.

Syntax:

```
http://<servername>/axis-cgi/com/serial.cgi?<parameter>=<value>

[&<parameter>=<value>...]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
port=<int>	1, ... [1]	Select COM port.

camera=<int>
unit=<int>
	1, ... [1]	Selects the source camera or external unit. If omitted, and "port=" command is also omitted, the default camera/unit is used to

		determine the serial port to use.
connect=<string>	yes	Makes the server keep the connection open, and start acting as a link between the client and the serial port.

[1] Product-dependent. Check the product specification.

5.10 PPP

The requests specified in the PPP section are supported by those products that have support for the PPP protocol.

5.10.1 Close PPP connection

Note: This request requires administrator access (administrator authorization).

Accessing this URL will force the Axis device to immediately close and disconnect any current PPP connection. When closing a PPP connection, all connections will be closed (dial-in, dial-out, callback).

Method: GET

Syntax:

```
http://<servername>/axis-cgi/admin/closeppp.cgi[?<wait>]
```

with the following parameter and value

<parameter>	**Value**	**Description**
<wait>	<int>	Number of seconds from request is received until the PPP connection is closed. *Default:* 0

Return: close PPP connection

```
HTTP/1.0 204 No Content\r\n
```

5.10.2 Reset PPP connection maxtimer

Note: This request requires administrator access (administrator authorization).

Accessing this URL will reset the timer counting down the maximum time a device is allowed to be connected over a PPP connection.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/admin/resetppptimer.cgi
```

Return: reset PPP connection maxtimer

```
HTTP/1.0 204 No Content\r\n
```

5.11 Event

The requests specified in the Event section are supported by those products that have support for event handling.

5.11.1 Event administration

Note: This request requires Administrator or Operator access.
Used to administer the list of event entries.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/operator/ev_admin.cgi?add
```

```
http://<servername>/axis-cgi/operator/ev_admin.cgi?delete=<entry>
```

5.11.1.1 Actions

These are the actions used to administer the events. Each event is given an entry identifier (*E0*, *E1 E2*, etc) which is later used to delete the entry. The identifier of a deleted entry may be reused by a new entry.

Action/Syntax	**Argument**	**Description**
add		Add a new empty entry to list of events.
delete=<*entry*>	E0, E1, E2, ...	Delete an entry from list of events.

5.11.1.2 Server responses

The actions produce one of the following server responses:

Return: A successful *delete*:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
OK\r\n
```

Return: A successful *add*:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
<entry> OK\r\n
```

Return: A failed request. Input values are probably incorrect:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
```

```
\r\n
Request failed: <error message>\r\n
```

Example: Adding new event entry

```
http://myserver/axis-cgi/operator/ev_admin.cgi?add
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
E7 OK\r\n
```

Example: Deleting event entry

```
http://myserver/axis-cgi/operator/ev_admin.cgi?delete=E7
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
OK\r\n
```

5.12 Firewall

The requests specified in the Firewall section are supported by those products that have built-in firewall functionality.

5.12.1 Firewall administration

Note: This request requires Administrator access.
Used to administer the built-in firewall, which restricts network access to the server. When enabled, only the hosts added through this interface will be able to access the server.

For each added entry, the services that are to be accessible must be specified. This is done by parameters, from where a specific service or a range of port numbers for custom services can be specified.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/admin/fwall_admin.cgi?list

http://<servername>/axis-cgi/admin/fwall_admin.cgi?enable[&<option> ...]

http://<servername>/axis-cgi/admin/fwall_admin.cgi?disable[&<option> ...]

http://<servername>/axis-cgi/admin/fwall_admin.cgi?add[&<parameter>[=<value>] ...]]
[&<option> ...]

http://<servername>/axis-cgi/admin/fwall_admin.cgi?update=<entry>[&<parameter>
[=<value>] ...]][&<option> ...]

http://<servername>/axis-cgi/admin/fwall_admin.cgi?delete=<entry>[&<option> ...]
```

5.12.1.1 Actions

These are the actions used to administer the firewall. Each added host is given an entry identifier (*F0*, *F1 F2*, etc) which is later used to modify or delete the entry. The identifier of a deleted entry may be reused by a new entry.

Action/Syntax	**Argument**	**Description**
list		Produce a list of all settings and entries.
add		Add a new entry to list of accepted hosts.
delete=<*entry*>	F0, F1, F2, ...	Delete an entry from list of accepted hosts.
update=<*entry*>	F0, F1, F2, ...	Update an entry in list of accepted hosts.
enable		Enable the firewall. Blocks access from all except the added entries.
disable		Disables the firewall. Enables access by everyone.

5.12.1.2 Parameters

Parameter/Syntax	**Values**	**Description**

host=<*address*>[1]	host name[2] or numerical Internet address	Host to add or modify.
http		Opens ports for HTTP access.
https		Opens ports for HTTPS access.
ftp		Opens ports for FTP access.
telnet		Opens ports for TELNET access.
tcp=<*int*>-<*int*>	1024 - 65535	Opens specified TCP port range.
udp=<*int*>-<*int*>	1024 - 65535	Opens specified UDP port range.

[1] Required by action *add*.

[2]Address specified by host name requires that the server has access to a Domain Name Server.

5.12.1.3 Options

Before modifying the settings of an enabled firewall, the script will verify that the host that requested the action will still have access to the server once the action is performed. If the client uses a proxy, proxy access will be verified instead.

If verification fails, i.e. the host would have been blocked from the server, the action is not carried out and an error is returned. This prevents accidental lock out by the administrator, etc.

By setting the *force* option this verification is skipped and all actions are carried out, regardless of their consequences.

Option/Syntax	**Description**
force	Do not verify host accessibility.

5.12.1.4 Server responses

The actions produce one of the following server responses:

Return: A successful *enable*, *disable*, *delete* or *update*:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
OK\r\n
```

Return: A successful *add*:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
<entry> OK\r\n
```

Return: A failed request. Input values are probably incorrect:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Request failed: <error message>\r\n
```

Return: A failed verification by an *enable*, *add*, *delete* or *update* request. No settings have been changed. Use the *force* option to override the verification.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Verification failed for IP <Numerical Internet address>\r\n
```

Return: A list request.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
[Firewall]\r\n
enabled=<value>\r\n
\r\n
<entry data>
<entry data>
...
```

where each <*entry data*> is:

```
[<entry>]\r\n
host=<value>\r\n
http=<value>\r\n
https=<value>\r\n
ftp=<value>\r\n
telnet=<value>\r\n
tcp=<value>\r\n
udp=<value>\r\n
\r\n
```

The parameter values has the following syntax:

Parameter	**Values**	**Description**
enabled	yes, no	States if the firewall is enabled or disabled.
host	host name[1] or numeric Internet address	The host for this entry.
http	yes, no	States if host has HTTP access.
https	yes, no	States if host has HTTPS access.
ftp	yes, no	States if host has FTP access.
telnet	yes, no	States if host has TELNET access.
tcp	[1024-65535]-[1024-65535]	Specific TCP port range accessible by host.
udp	[1024-65535]-[1024-65535]	Specific UDP port range accessible by host.

[1]Address specified by host name requires that the server has access to a Domain Name Server.

Example: Adding new host with HTTP and FTP access, with verification

```
http://myserver/axis-cgi/admin/fwall_admin.cgi?add&host=10.13.18.20&http&ftp
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
F1 OK\r\n
```

Example: Listing entries

```
http://myserver/axis-cgi/admin/fwall_admin.cgi?list
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
[Firewall]\r\n
enabled="yes"\r\n
\r\n
[F0]\r\n
\r\n
host=10.13.18.1\r\n
http=yes\r\n
https=no\r\n
ftp=no\r\n
telnet=no\r\n
tcp=4000-4001\r\n
udp=\r\n
\r\n
[F1]\r\n
host=10.13.18.20\r\n
http=yes\r\n
https=no\r\n
ftp=yes\r\n
telnet=no\r\n
tcp=\r\n
udp=\r\n
\r\n
```

Example: Deleting entry, with failed verification

```
http://myserver/axis-cgi/admin/fwall_admin.cgi?delete=F1
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
```

```
Verification failed for IP 10.13.18.20\r\n
```

Example: Deleting entry, overriding verification

```
http://myserver/axis-cgi/admin/fwall_admin.cgi?delete=F1&force
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
OK\r\n
```