VAPIX®, HTTP API Specification

Revision: 2.14
Date: 2007-October-17

TABLE OF CONTENTS

- DOCUMENT HISTORY
- 1 OVERVIEW
  - 1.1 Product and firmware versions
- 2 REFERENCES
- 3 DEFINITIONS
  - 3.1 General notations
    - 3.1.1 General abbreviations
    - 3.1.2 Style convention
    - 3.1.3 General CGI URL syntax and parameters
    - 3.1.4 Parameter value convention
- 4 INTERFACE SPECIFICATION
  - 4.1 Naming conventions and URL syntax
    - 4.1.1 Obsolete CGI parameters
  - 4.2 Server responses
    - 4.2.1 HTTP status codes
- 5 API GROUPS
  - 5.1 General
    - 5.1.1 Add, update, remove and list parameters and their values
      - 5.1.1.1 List parameters
      - 5.1.1.2 List output format
      - 5.1.1.3 Update parameters
      - 5.1.1.4 Add parameters
      - 5.1.1.5 Remove parameters
      - 5.1.1.6 Add/Remove server responses
    - 5.1.2 Add, modify and delete users
    - 5.1.3 Factory default
    - 5.1.4 Hard factory default
    - 5.1.5 Backup
    - 5.1.6 Restore
    - 5.1.7 Firmware upgrade
    - 5.1.8 Restart server

    - 5.1.9 Server report
    - 5.1.10 System logs
    - 5.1.11 Access logs
    - 5.1.12 System date and time
        - 5.1.12.1 Get system date and time
        - 5.1.12.2 Set system date and time
- 5.2 Image and Video
    - 5.2.1 Image size
    - 5.2.2 Video status
    - 5.2.3 Bitmap
        - 5.2.3.1 Bitmap image request
        - 5.2.3.2 Bitmap image (snapshot) CGI request
        - 5.2.3.3 Bitmap image response
    - 5.2.4 JPEG/MJPG
        - 5.2.4.1 JPEG image request
        - 5.2.4.2 JPEG image (snapshot) CGI request
        - 5.2.4.3 JPEG image response
        - 5.2.4.4 JPEG buffer request
        - 5.2.4.5 MJPG video request
        - 5.2.4.6 MJPG video CGI request
        - 5.2.4.7 MJPG video response
    - 5.2.5 MPEG-4
        - 5.2.5.1 MPEG-4 SDP description request
        - 5.2.5.2 Restart the MPEG-4 stream
        - 5.2.5.3 MPEG-4 statistics
    - 5.2.6 Dynamic text overlay
- 5.3 PTZ
    - 5.3.1 PTZ driver update
    - 5.3.2 PTZ administration
    - 5.3.3 PTZ control
    - 5.3.4 PTZ configuration
    - 5.3.5 Set PTZ parameters
    - 5.3.6 PTZ control queue
    - 5.3.7 PTZ control queue response
- 5.4 Motion Detection
    - 5.4.1 Add a Motion Detection window
    - 5.4.2 Remove a Motion Detection window
    - 5.4.3 Update the Motion Detection parameters
    - 5.4.4 List the Motion detection parameters

    - 5.4.5 Get the Motion Detection level
- 5.5 I/O
    - 5.5.1 I/O control
        - 5.5.1.1 Input
        - 5.5.1.2 Output
    - 5.5.2 Virtual I/O control
        - 5.5.2.1 Input
- 5.6 Serial port
    - 5.6.1 Serial port control
    - 5.6.2 Open serial port
- 5.7 IP filter
    - 5.7.1 IP filter administration
    - 5.7.2 Server responses
- 5.8 Audio
    - 5.8.1 Audio MIME types
    - 5.8.2 Audio data request
    - 5.8.3 Singlepart audio data response
    - 5.8.4 Multipart audio data response
    - 5.8.5 Audio data transmit
- 5.9 AXIS 292 Network Video Decoder
    - 5.9.1 Alarm
    - 5.9.2 Video control
        - 5.9.2.1 Connect
        - 5.9.2.2 Disconnect
        - 5.9.2.3 Invalidate Cache
        - 5.9.2.4 Select Source

DOCUMENT HISTORY

Version	**Date**	**Comment**
2.00	2003-Sep-16	Initial version
2.01	2004-Feb-03	Added requests for Video status, Bitmap, JPEG buffer, PTZ, Motion Detection, I/O and Serial port. Added new parameters to JPEG image CGI request and MJPG video CGI request.
2.02	2004-Feb-16	Added requests for Image overlay.
2.03	2004-May-27	Added more valid values for the parameter resolution, to the Bitmap image, JPEG image and MJPG video CGI requests. Added requests for Backup, Restore, Firmware upgrade and IP address

		filtering.
2.04	2004-July-09	Added more valid values for the parameter resolution, to the JPEG image and MJPG video CGI requests.
2.05	2004-July-23	Added requests for MPEG-4.
2.06	2004-Nov-03	Added requests for Audio. Added responseformat parameter to param.cgi?action=list.
2.07	2005-Feb-25	Removed the nosync parameter from param.cgi. Added reference to the RTSP API for controlling MPEG-4 streams. The MPEG-4 SDP description request have been changed to media.sdp. The previous video.sdp will still be supported.
2.08	2005-May-26	Added a policy parameter for the ipfilter.cgi and examples. Added new HTTP APIs for the AXIS 292 Network Video Decoder.
2.09	2005-Oct-26	Added example to list parameters using wildcards. Added the parameter "range" to the JPEG buffer request. Added example to pulse the virtual input. Moved PTZ preset position configuration from ptz.cgi to ptzconfig.cgi. Added ptzconfig.cgi for OSD menu control. Updated information about the PTZ control queue response.
2.10	2007-Jan-11	Added force parameter to param.cgi. Added explanations for digital PTZ. Added areazoom parameter to ptz.cgi. Added timezone parameter to date.cgi. Corrected some examples and added audio transmit examples.
2.11	2007-Jan-31	Removed Image overlay APIs. Removed areazoom parameter for ptz.cgi, new API will soon be available.
2.12	2007-Feb-28	Added areazoom parameter for ptz.cgi.
2.13	2007-June-29	Added Dynamic text overlay.
2.14	2007-Oct-17	Added access logs.

1 OVERVIEW

This document specifies the external HTTP-based application programming interface of the Axis camera and video servers with firmware version 4.00 and above.

The HTTP-based video interface provides the functionality for requesting single and multi-part images 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 servers.

1.1 Product and firmware versions

The support for the HTTP API is product and firmware dependent. 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)

- VAPIX®, HTTP API Specification
- VAPIX®, RTSP API Specification
- Axis Video Product Release Notes
- VAPIX®, Parameter specification

3 DEFINITIONS

This section contains information on general usage of this document.

3.1 General notation

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* - signifies 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

URL RFC 1738 describes the syntax and semantics for a compact string representation for a resource available via the Internet. These strings are called "Uniform Resource Locators" (URLs).

URI A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource. RFC 2396 describes the generic syntax of URI.

3.1.2 Style convention

In URL syntax and in descriptions of CGI parameters, text in italics 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 are 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. When the CGI request includes internal camera parameters, the internal parameters must be written exactly as named in the camera or video server. For the POST method, the parameters must be included in the body of the HTTP request. 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: List the Network parameters.

```
http://<servername>/axis-cgi/operator/param.cgi?action=list&group=Network
```

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 may be 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.

Some CGI requests described in the Axis Video HTTP API version 1.xx are still supported by Axis products with firmware version 4.00 and above, and are provided for backward compatibility. They are, however, obsolete and should not be used. For more information, please read the document describing the differences between HTTP API version 1 and version 2.

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.
302	Moved Temporarily	The server redirects the request to the URI given in the Location header.
400	Bad Request	The request had bad syntax or was impossible to fulfill.
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 with firmware version 4.00 and above.

5.1.1 Add, update, remove and list parameters and their values

Note:

- These requests have different security levels. The security level for each parameter is specified in the parameter document.
- The URL must follow the standard way of writing a URL, (RFC 2396: Uniform Resource Identifiers (URI) Generic Syntax); that is, spaces and other reserved characters (";", "/", "?", ":", "@", "&", "=", "+", "," and "$") within a <parameter> or a <value> must be replaced with %<ASCII hex>. For example, in the string My camera, the space will have to be replaced with %20, My%20camera.

Method: GET/POST

Syntax:

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

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

with the following parameter and values

<parameter>=<value>	**Values**	**Description**
action=*<string>*	add, remove, update or list	Specifies the action to take. Depending on this parameter, various parameters may be set, as described in the following sections.

5.1.1.1 List parameters

Syntax:

```
http://<servername>/axis-cgi/view/param.cgi?action=list
[&<parameter>=<value>...]
http://<servername>/axis-cgi/operator/param.cgi?action=list
[&<parameter>=<value>...]
http://<servername>/axis-cgi/admin/param.cgi?action=list
[&<parameter>=<value>...]
```

with the following parameter and values

<parameter>=<value>	**Values**	**Description**
group=*<string>*[,*<string>*...]	*<group*[*.name*]*>*[,*<group*[*.name*]*>*...]	Returns the value of the camera parameter named *<group>*.*<name>*. If *<name>* is omitted, all the parameters of the *<group>* are returned. The camera parameters must be entered exactly as they are named in the camera or video server. Wildcard (*) can be used when listing parameters. See example below. If this parameter is omitted, all parameters in the device are returned.
responseformat	rfc	Get the HTTP response format according to standard. Response format: HTTP/1.0 200 OK\r\n Content-Type: text/plain\r\n \r\n *<parameter pair>*

Example: List the Network parameters.

```
http://myserver/axis-cgi/admin/param.cgi?action=list&group=Network
```

Example: List the names of all Event parameters.

```
http://myserver/axis-cgi/admin/param.cgi?action=list&group=Event.*.Name
```

5.1.1.2 List output format

```
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: Network query response.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
root.Network.IPAddress=10.13.12.36\n
root.Network.SubnetMask=255.255.255.0\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: <description>\n
```

5.1.1.3 Update parameters

Syntax:

```
http://<servername>/axis-cgi/operator/param.cgi?action=update
[&<parameter>=<value>...]
http://<servername>/axis-cgi/admin/param.cgi?action=update
[&<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.

Example: Set the default image resolution to 320x240 pixels.

```
http://myserver/axis-cgi/operator/param.cgi?
action=update&Image.I0.Resolution=320x240
```

Example: Set the maximum number of viewers to 5.

```
http://myserver/axis-cgi/operator/param.cgi?
action=update&Image.MaxViewers=5
```

5.1.1.4 Add parameters

Note: Only applicable for dynamic parameter groups such as the event parameters.

Syntax:

```
http://<servername>/axis-
cgi/operator/param.cgi?action=add[&<parameter>=<value>...]
http://<servername>/axis-
cgi/admin/param.cgi?action=add[&<parameter>=<value>...]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
template=<*string*>	<*template*>	Use the specified <*template*> when creating the new group. The template is a file, which describes all parameters for this group. Depending on the product, different templates are available, please check the Release notes of the firmware version. See examples below.
group=<*string*>	<*group*>	Specifies the parent group. The parent group defines where in the parameter structure the new group will be created. For example, if adding an event (template=event) and specify group=Event the new group will be available as Event.E<number>. Where <number> is the unique number for the group (see return values below). The character before <number> is generated from the last section of the group name. E.g. Event will generate the character E and Event.Notification will generate the character N.
<*string*>=<*string*>	<*group.name*>=<*value*>	Set a parameter in the newly created group. As the group number is not known before the group is created, the id-number is simply left out, see the examples below. The new group number is created dynamically and can be any number. This is why all parameters are specified to set without any group number. The base path to the parameter is specified as <group>.<uppercase first letter of group>.<parameter name>.
force	yes	The force parameter can be used to exceed limits set for adding

		dynamic parameter groups. Axis products can for example be configured for up to 10 event types. The force parameter can be used to exceed this maximum of events.

Example: Create a new event under the group Event and set the name to MyEvent and the Enabled parameter to yes.

```
http://myserver/axis-cgi/operator/param.cgi?action=add&group=Event
&template=event&Event.E.Name=MyEvent&Event.E.Enabled=yes
```

Example: A listing of the new group will output the following.

```
root.Event.E0.Name=MyEvent
root.Event.E0.Type=T
root.Event.E0.Enabled=yes
root.Event.E0.Active=no
root.Event.E0.Priority=1
root.Event.E0.Image=1
root.Event.E0.HWInputs=xxxx
root.Event.E0.SWInput=
root.Event.E0.Weekdays=1111111
root.Event.E0.Starttime=00:00
root.Event.E0.Duration=0
root.Event.E0.ImageURLSettingsEnabled=no
root.Event.E0.ImageURLSettings=
root.Event.E0.IncludePreTrigger=no
root.Event.E0.PreTriggerSize=0
root.Event.E0.PreTriggerInterval=1000
root.Event.E0.PreTriggerIntervalUnit=s
root.Event.E0.PreTriggerUnit=s
root.Event.E0.IncludePostTrigger=no
root.Event.E0.PostTriggerSize=0
root.Event.E0.PostTriggerInterval=1000
root.Event.E0.PostTriggerIntervalUnit=s
root.Event.E0.PostTriggerUnit=s
root.Event.E0.IncludeBestEffort=no
root.Event.E0.BestEffortInterval=1000
root.Event.E0.BestEffortDuration=0
root.Event.E0.BestEffortIntervalUnit=s
root.Event.E0.BestEffortDurationUnit=s
root.Event.E0.FileName=image.jpg
root.Event.E0.Suffix=%y-%m-%d_%H-%M-%S-%f
root.Event.E0.MaxSequenceNumber=-100
```

Note that in this example the id is E0. This can be any number, depending on if other events were added before. Parameters that are not specified in the request will have their default values, as specified in the configuration file.

5.1.1.5 Remove parameters

Note: Only applicable for dynamic parameter groups such as the event parameters.

Syntax:

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

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
group=<*string*>[,<*string*>...]	<*group*>[,<*group*>]	Deletes the specified <*group*>

Example: Delete event group E2 and E4.

```
http://myserver/axis-
cgi/operator/param.cgi?action=remove&group=Event.E2,Event.E4
```

5.1.1.6 Add/Remove server responses

These actions produce one of the following server responses:

Return: A successful *add.*

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

Return: A successful *remove.*

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

Return: Add new group failed. The group was not created, due to errors in the parameters to add command.

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

Return: Add new group failed. The group was created, but the specified parameters could not be set.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
<additional error information>
# Error: <error message>\r\n
```

Return: Remove failed.

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

Example: Add new event entry and set the specified name.

```
http://myserver/axis-
cgi/operator/param.cgi?action=add&group=Event&template=event
&Event.E.Name=MyEvent
```

Response:

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

Example: Delete an event entry.

```
http://myserver/axis-cgi/operator/param.cgi?action=remove&group=Event.E7
```

Response:

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

5.1.2 Add, modify and delete users

Add a new user with password and group membership, modify the information and remove a user.

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

Method: GET/POST

Syntax:

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
action=<*string*>	add, update, remove or get	add = create a new user account. update = change account information of specified parameters if the account exists. remove = remove an existing account if it exists. get = get a list of the users which belong to each group

		defined.
user=*<string>*	*<string>*	The user account name, a non-existing user name. Valid characters are a thru z, A thru Z and 0 thru 9.
pwd=*<string>*	*<string>*	The unencrypted password of the account. Valid characters are a thru z, A thru Z and 0 thru 9.
grp=*<string>*	*<string>*	An existing primary group name of the account.
sgrp=*<string>*:[*<string>*...]	*<string>*[,*<string>*...]	Colon separated existing secondary group names of the account.
comment=*<string>*	*<string>*	The comment field of the account.

Example: Create a new administrator account.

```
http://myserver/axis-
cgi/admin/pwdgrp.cgi?action=add&user=joe&pwd=foo&grp=axuser
&sgrp=axadmin:axoper:axview&comment=Joe
```

Example: Change the password of an existing account.

```
http://myserver/axis-cgi/admin/pwdgrp.cgi?action=update&user=joe&pwd=bar
```

Example: Remove an account.

```
http://myserver/axis-cgi/admin/pwdgrp.cgi?action=remove&user=joe
```

Example: List groups and users.

```
http://myserver/axis-cgi/admin/pwdgrp.cgi?action=get
```

5.1.3 Factory default

Reload factory default. All parameters except Network.BootProto, Network.IPAddress, Network.SubnetMask, Network.Broadcast and Network.DefaultRouter are set to their factory default values.

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

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

5.1.4 Hard factory default

Reload factory default. All parameters are set to their factory default value.

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

Method: GET

Syntax:

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

5.1.5 Backup

Download a unit specific backup of all files in the folder /etc in tar format.

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

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

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: application/x-tar\r\n
Content-Disposition: attachment; filename=backup_<MAC address>.tar\r\n
\r\n
<file content of backup_<MAC address>.tar>
```

5.1.6 Restore

Upload a unit specific backup previously created by the backup.cgi.

Note: This requires administrator access (administrator authorization).

Method: POST

Syntax:

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

The file content is provided in the HTTP body according to the format given in RFC 1867. The body is created automatically by the browser if using HTML form with input type "file".

Example: Upload of backup, where "\r\n" has been omitted in the HTTP body.

```
POST /axis-cgi/admin/restore.cgi? HTTP/1.0\r\n
Content-Type: multipart/form-data; boundary=AaBo3x\r\n
Content-Length: <content length>\r\n
\r\n
--AaBo3x\r\n
Content-Disposition: form-data; name="backup.tar";
filename="backup_<MAC address>.tar"\r\n
Content-Type: application/x-tar\r\n
\r\n
<file content of backup_<MAC address>.tar>
\r\n
--AaBo3x--\r\n
```

5.1.7 Firmware upgrade

Upgrade the firmware version.

Note: This requires administrator access (administrator authorization).

Method: POST

Syntax:

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

with the following parameters and values

<*parameter*>=<*value*>	**Values**	**Description**
type=<*string*>	normal, factorydefault	Specifies the type of firmware upgrade. normal = Upgrade and restore old settings. factorydefault = Upgrade and discard all settings. type is by default set to normal.

The file content is provided in the HTTP body according to the format given in RFC 1867. The body is created automatically by the browser if using HTML form with input type "file".

Example:

```
POST /axis-cgi/admin/firmwareupgrade.cgi?type=normal HTTP/1.0\r\n
Content-Type: multipart/form-data; boundary=AsCg5y\r\n
Content-Length: <content length>\r\n
\r\n
--AsCg5y\r\n
Content-Disposition: form-data; name="firmware.bin";
filename="firmware.bin"\r\n
Content-Type: application/octet-stream\r\n
\r\n
<firmware file content>
\r\n
--AsCg5y--\r\n
```

5.1.8 Restart server

Restart server.

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

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

5.1.9 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 requires administrator access (administrator authorization).

Method: GET

Syntax:

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

5.1.10 System logs

Get system log information.

Note: This 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\r\n
\r\n
<system log information>
```

5.1.11 Access logs

Get client access log information. The level of information included in the log is set in the Log.Access parameter group.

Note: This requires administrator access (administrator authorization).

Note: The response is product/release-dependent.

Method: GET

Syntax:

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

Return:

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

```
\r\n
<system log information>
```

5.1.12 System date and time

Get or set the system date and time.

Method: GET/POST

Syntax:

```
http://<servername>/axis-cgi/admin/date.cgi?<parameter>=<value>
```

with the following parameter and values

<parameter>=<value>	**Values**	**Description**
action=*<string>*	get or set	Specifies what to do. get = get the current date and time. set = set the current date and/or time.

5.1.12.1 Get system date and time

Syntax:

```
http://<servername>/axis-cgi/admin/date.cgi?action=get
```

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
<month> <day>, <year> <hour>:<minute>:<second>\r\n
```

Example:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Apr 03, 2003 15:16:04\r\n
```

5.1.12.2 Set system date and time

Syntax:

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

with the following parameters and values

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

year=<*int*>	1970 - 2031	Current year.
month=<*int*>	1 - 12	Current month.
day=<*int*>	1 - 31	Current day.
hour=<*int*>	0 - 23	Current hour.
minute=<*int*>	0 - 59	Current minute.
second=<*int*>	0 - 59	Current second.
timezone=<*string*>	GMT	Specifies the time zone that the new date and/or time is given in. The camera translates the time into local time using whichever time zone has been specified through the web configuration. If omitted the new date and/or time is assumed to be in local time. **Note:** Requires that daylight saving time (DST) is turned off, and that the time mode of the camera is not to synchronize with an NTP server or with the computer time. Currently only GMT is considered valid input. The rest of the time zones are subject to future expansion.

The set action produces one of the following server responses:

Return: A successful *set.*

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

Return: A failed *set*. Settings or syntax 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: Set the date.

```
http://myserver/axis-cgi/admin/date.cgi?action=set&year=2005&month=4&day=3
```

Response:

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

5.2 Image and video

5.2.1 Image size

Get the actual image size of default image settings, or with given parameters.

Method: GET/POST

Syntax:

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
camera=<*string*>	1, 2, 3, 4 or quad[1]	Returns image size of the camera. **Note:** This parameter is required.
Any parameter from jpeg-interface affecting image size.		See 5.2.4.2 for more information about image parameters.

[1] The number of video inputs may differ between different cameras and video servers. See the product's specification.

Example: Request image size of default settings from camera 1.

```
http://myserver/axis-cgi/view/imagesize.cgi?camera=1
```

Example: Returned data after a successful request.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
image width = 176\n
image height = 144\n
```

Example: Request image size with supplied parameters for camera 1.

```
http://myserver/axis-
cgi/view/imagesize.cgi?camera=1&resolution=QCIF&compression=60
```

Example: Returned data after a successful request.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
image width = 160\n
image height = 120\n
```

5.2.2 Video status

Check the status of one or more video sources.

Method: GET

Syntax:

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

with the following parameter and values

<parameter>=<value>	**Values**	**Description**
status=<*int*>[[,<*int*>],...]	1 ... 4[1]	Returns the status of one or more inputs.

[1] The number of video inputs may differ between different cameras and video servers. See the product's specification.

Example: Request video status from input 1, 2, 3 and 4.

```
http://myserver/axis-cgi/view/videostatus.cgi?status=1,2,3,4
```

Example: Returned data after a successful request.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Video 1 = video
Video 2 = no video
Video 3 = no video
Video 4 = video
```

5.2.3 Bitmap

The requests specified in the bitmap section are supported by those video products that have bitmap support. Supported image formats can be checked by reading the parameter Properties.Image.Format. The parameter can be listed using param.cgi.

5.2.3.1 Bitmap image request

Returns an image with the default resolution as defined in the system configuration.

Method: GET

Syntax: Request a bitmap image

```
http://<servername>/bitmap[/<camera>]/image.bmp
```

with the following parameter and values

<parameter>=<value>	**Values**	**Description**
<*camera*>	1 ... 4[1] quad[1]	Select the input source. Applies only to servers with more than one input source.

[1] Product-dependent. See the product's specification.

Example: Request a bitmap image from the default camera with the default resolution.

```
http://myserver/bitmap/image.bmp
```

Example: Request a bitmap image from camera 2 with the default resolution.

```
http://myserver/bitmap/2/image.bmp
```

Example: Request a bitmap image containing four video sources.

```
http://myserver/bitmap/quad/image.bmp
```

5.2.3.2 Bitmap image (snapshot) CGI request

Request a bitmap image with the specified properties.

Method: GET

Syntax:

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
resolution=<*string*>	768x576, 4CIF, 704x576, 704x480, VGA, 640x480, 2CIFEXP, 2CIF, 704x288, 704x240, 480x360, CIF, 384x288, 352x288, 352x240, 320x240, 240x180, QCIF, 192x144, 176x144, 176x120, 160x120 [1]	Specifies the resolution as <*width*> times <*height*> number of pixels of the returned image.
camera=<*string*>	1 ... 4[1] quad[1]	The camera source. Applies only to video servers with more than one video input.
squarepixel=<*int*>	0, 1	Enable/disable square pixel correction. Applies only to video servers.

[1] Product-dependent. See the product's specification.

Example: Request a bitmap image from camera 1 with a resolution of 320x240.

```
http://myserver/axis-cgi/bitmap/image.bmp?resolution=320x240&camera=1
```

5.2.3.3 Bitmap image response

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

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: image/bitmap\r\n
Content-Length: <image size>\r\n
\r\n
<bitmap image data>\r\n
```

Example: Requested bitmap image.

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

```
Content-Type: image/bitmap\r\n
Content-Length: 1216566\r\n
\r\n
<bitmap image data>\r\n
```

5.2.4 JPEG/MJPG

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

5.2.4.1 JPEG image request

Returns an image with the default resolution and compression as defined in the system configuration.

Method: GET

Syntax:

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

with the following parameter

Parameter	**Values**	**Description**
<camera>	1, 2, 3, 4 or quad[1]	Select input source. Applies only to servers with more than one input source. *Default:* default camera.

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

Example: Request JPEG image from default camera with default resolution and compression.

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

Example: Request JPEG image from camera number 2 with default resolution and compression.

```
http://myserver/jpg/2/image.jpg
```

Example: Request JPEG image containing four video sources.

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

5.2.4.2 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>...]]
```

with the following parameters and values

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

resolution=<*string*>	1280x1024, 1280x960, 1280x720, 768x576, 4CIF, 704x576, 704x480, VGA, 640x480, 640x360, 2CIFEXP, 2CIF, 704x288, 704x240, 480x360, CIF, 384x288, 352x288, 352x240, 320x240, 240x180, QCIF, 192x144, 176x144, 176x120, 160x120 [1]	Specify the resolution of the returned image.
camera=<*string*>	1, 2, 3, 4 or quad [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*>[2]	0 - 100 [1]	Sets level of color or grey-scale. 0 = grey-scale, 100 = full color. **Note:** This value is internally mapped and is therefore product-dependent.
color=<*int*>	0, 1	Enables/disables color. 0 = black and white, 1 = color.
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.
text=<*int*>	0, 1	Shows/hides the text. 0 = hide, 1 = show.
textstring=<*string*>[2]	A string	The text shown in the image, the string must be URL encoded.
textcolor=<*string*>[2]	black, white	The color of the text shown in the image.
textbackgroundcolor=<*string*>[2]	black, white, transparent, semitransparent	The color of the text background shown in the image.
rotation=<*int*>	0, 90, 180, 270 [1]	Rotates the image clockwise.
textpos=<*string*>	top, bottom	The position of the string shown in the image.

overlayimage=<*int*>[2]	0, 1	Enable/disable overlay image.
overlaypos=<*int*>x<*int*>[2]	<*xoffset*>[1]x<*yoffset*>[1]	Set the position of the overlay image.
squarepixel=<*int*>[2]	0, 1	Enable/disable square pixel correction. Applies only to video servers.

[1] Product-dependent. Check the product's specification.
[2] Support for this parameter is product/release-dependent.

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
```

Example: Request a JPEG image from camera 2 with the text My Camera displayed.

```
http://myserver/axis-cgi/jpg/image.cgi?camera=2&text=1&textstring=My%20Camera
```

5.2.4.3 JPEG image response

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

Return:

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

5.2.4.4 JPEG buffer request

Requests 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	start = create a new image buffer. stop = this corresponds to an alarm occurring and it makes

	get reset remove	the image buffer store post-alarm images. The buffer will stop after all post-alarm images have been taken. get = used to fetch an image from the image buffer. reset = restarts the image buffer, removing any buffered images. remove = removes the image buffer, including any buffered images.
buffername=<*string*>	<*string*>	Name used for identifying the buffer.
uri=<*string*>	<*string*>	Corresponding image URI to be used by the image buffer. **Note:** Must be URI encoded. The URI should also begin with "ftp://".
prealarm=<*int*>	0, ...	Number of images to be saved 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.
postdelay=<*int*>	<*milliseconds*>	The preferred time between the post-alarm images in milliseconds.
index=<*int*>	<*image number*>	The index of an image in the buffer.
range=<*string*>	all, allpre, allpost, <*index*>-<*index*>	Retrieve a range of images from the image buffer. all = get all the images in the buffer. allpre = get all the pre alarm images. allpost = get all the post alarm images. <*index*>-<*index*> = get images in the specified range.

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 the buffer.

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

Example 4: Create an image buffer, named DOOR2. The images should be taken from camera 1 and have a resolution of 320x240 pixels. Five post-alarm images should be stored.

```
http://myserver/axis-cgi/buffer/command.cgi?do=start&buffername=DOOR2
&postalarm=5&uri=ftp://jpg/image.jpg?resolution=320x240&camera=1
```

5.2.4.5 MJPG video request

Returns a multipart image stream with the default resolution and compression as defined in the system configuration.

Method: GET

Syntax: Request Multipart JPEG image.

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

with the following parameter

Parameter	Values	Description
<*camera*>	1, 2, 3, 4 or quad[1]	Select input source. Applies only to servers with more than one input source. *Default:* default camera.

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

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

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

Example: Request quad composed JPEG image stream with default resolution and compression.

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

5.2.4.6 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=<*string*>	1280x1024, 1280x960, 1280x720, 768x576, 4CIF, 704x576, 704x480, VGA, 640x480, 640x360, 2CIFEXP, 2CIF, 704x288, 704x240, 480x360, CIF, 384x288, 352x288, 352x240, 320x240, 240x180, QCIF, 192x144, 176x144, 176x120, 160x120 [1]	Specify the resolution of the returned image.
camera=<*string*>	1, 2, 3, 4 or quad [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*>[2]	0 - 100 [1]	Sets level of color or grey-scale. 0 = grey-scale, 100 = full color. **Note:** This value is internally mapped and is therefore product-dependent.
color=<*int*>	0, 1	Enables/disables color. 0 = black and white, 1 = color.
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.
text=<*int*>	0, 1	Shows/hides the text. 0 = hide, 1 = show.
textstring=<*string*>[2]	A string	The text shown in the image, the string must be URL encoded.
textcolor=<*string*>[2]	black, white	The color of the text shown in the image.
textbackgroundcolor=<*string*>[2]	black, white, transparent, semitransparent	The color of the text background shown in the image.
textposition=<*string*>[2]	top, bottom	The position of the string shown in the image.
rotation=<*int*>	0, 90, 180, 270 [1]	Rotates the image clockwise.
textpos=<*int*>[2]	0, 1	Specify text position. 0 = top, 1 = bottom.
overlayimage=<*int*>[2]	0, 1	Enable/disable overlay image.
overlaypos=<*int*>x<*int*>[2]	<*xoffset*>[1]x<*yoffset*>[1]	Set the position of the overlay image.
squarepixel=<*int*>[2]	0, 1	Enable/disable square pixel correction. Applies only to video servers.
duration=<*int*>	0, ...	Specifies for how many seconds the video will be generated and pushed to the client. 0 = unlimited.

nbrofframes=<*int*>	0, ...	Specifies how many frames the server will generate and push. 0 = unlimited.
fps=<*int*>	0, ...	Using fps it is possible to specify the frame rate from the server. 0 = unlimited.

[1] Product-dependent. Check the product's specification.
[2] Support for this parameter is product/release-dependent.

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

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

5.2.4.7 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
\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
 .
 .
 .
```

5.2.5 MPEG-4

Due to the streaming properties of the MPEG-4 media format, the control of the MPEG-4 streams is handled via RTSP, which is an HTTP-like protocol designed for controlling real-time multi-media streams. Please check the RTSP API.

5.2.5.1 MPEG-4 SDP description request

Returns the SDP description for an MPEG-4 stream.

Method: GET

Syntax:

```
http://<servername>/mpeg4[/<camera>]/video.sdp [1]
http://<servername>/mpeg4[/<camera>]/media.sdp [1]
```

[1] Firmware dependent. media.sdp should be used for firmware 4.20 and above.

with the following parameter

<value>	**Values**	**Description**
<camera>	1, 2, 3, 4 or quad [1]	Select input source. Applies only to servers with more than one input source. *Default:* default camera.

[1] Product dependent. Check the product specification.

Example: Request the MPEG-4 SDP description for the first camera.

```
http://myserver/mpeg4/1/media.sdp
```

5.2.5.2 Restart the MPEG-4 stream

Restarts the MPEG-4 stream. This will force the server to break the current GOV and insert configuration headers followed by a new GOV. This can be used by a new client to make sure that it can start to show the stream as fast as possible.

Method: GET, POST

Syntax:

```
http://<servername>/axis-
cgi/mpeg4/restart_stream.cgi[?<parameter>=<value>&...]
```

with the following parameters and values

<value>	**Values**	**Description**
camera=<string>	1, 2, 3, 4 or quad [1]	Select input source. Applies only to servers with more than one input source. *Default:* default camera.

[1] Product dependent. Check the product specification.

Example: Restart the MPEG-4 stream from the first camera.

```
http://myserver/axis-cgi/mpeg4/restart_stream.cgi?camera=1
```

5.2.5.3 MPEG-4 statistics

It is possible to retrieve some statistics about the generated MPEG-4 streams. These statistics reflects the servers point-of-view of the stream, i.e., a slow network or client may result in a difference in appearance compared to these values.

Method: GET, POST

Syntax:

```
http://<servername>/axis-
cgi/operator/mpeg4_statistics.cgi[?<parameter>=<value>&...]
```

with the following parameters and values

<value>	**Values**	**Description**
camera=<string>	1, 2, 3, 4 or quad [1]	Select input source. Applies only to video servers with more than one input source. *Default:* default camera.
stats=<string>[,<string>...]	bitrate, minbitrate, maxbitrate, framerate, minframerate, maxframerate, compression	Which statistics to include. *Default:* all
reset=<int>	1	Reset the min and max values to the current values.

[1] Product dependent. Check the product specification.

Example: Retrieve all statistics for the first camera.

```
http://myserver/axis-cgi/operator/mpeg4_statistics.cgi?camera=1
```

Example: Retrieve the current bit rate and frame rate for the default camera.

```
http://myserver/axis-
cgi/operator/mpeg4_statistics.cgi?stats=bitrate,framerate
```

Example: Reset the min and max values for the first camera.

```
http://myserver/axis-cgi/operator/mpeg4_statistics.cgi?camera=1&reset=1
```

5.2.6 Dynamic text overlay

Request to set or get dynamic text overlay in the image.
Note: To use this functionality the text overlay must be enabled (Image.I#.Text.TextEnabled) and the text string (Image.I#.Text.String) must be set to contain the value *#D.*

Method: GET

Syntax:

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

with the following parameters and values

<value>	**Values**	**Description**
action=<*string*>	settext, gettext	settext = set the dynamic text overlay. gettext = get the dynamic text overlay.
text=<*string*>	A string	The overlay text to apply, only relevant with action=settext.
camera=<string>	1, 2, 3, 4 or quad [1]	Select input source. Applies only to video servers with more than one input source. *Default:* default camera.

[1] Product dependent. Check the product specification.

Example: Set the dynamic text overlay to "Test text" for cameras 1, 3 and 4.

```
http://myserver/axis-
cgi/operator/dynamicoverlay.cgi?action=settext&camera=1,3,4
&text=Test%20text
```

Example: Get the dynamic text overlay for camera 2 and quad view.

```
http://myserver/axis-
cgi/operator/dynamicoverlay.cgi?action=gettext&camera=2,quad
```

5.3 PTZ

This section describes request for Pan/Tilt/Zoom capable products. Supported requests are however product dependent.

Note! Installing a PTZ driver is done in three main steps:

1. Upload the driver
2. Associate the driver with a serial port
3. Connect cameras to the serial port

PTZ driver update can accomplish step 1 and 2 in one request.
PTZ administration can accomplish step 2 if the driver is already present in the product.
Open serial port accomplishes step 3.

5.3.1 PTZ driver update

Handles installation and removal of PTZ drivers.

Method: POST

Syntax:

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
replacedriver=<*string*>	Names from the parameter values in group *root.PTZ.PTZDrivers.Driver#* where '#' denotes the index for the installed driver.	Uninstall currently installed driver <string>. See below for rules concerning driver replacement.
port=<*int*>	1, ... [1]	Associate the driver with this serial port. See below for rules concerning driver replacement.
force=<*string*>	yes	Force driver installation even if the driver is potentially incompatible with firmware (if for instance a relevant library has been considerably updated) but may still work well. This parameter has no effect on serious incompatibilities such as wrong CPU architecture.

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

The file content, if any, is provided in the HTTP body according to the format given in RFC1867. The body is created automatically by the browser if using HTML form with input type "file". Empty file content will result in the removal of the driver only.

The following rules apply for driver replacement, in descending order of priority:

1. If parameter replacedriver is provided, the given driver will be uninstalled.
2. If parameter port is provided, the driver currently associated with that serial port, if any, is uninstalled.
3. If the maximum number of installed drivers (equals the number of serial ports) is not reached, no driver is uninstalled.
4. Any driver not currently associated with any serial port will be uninstalled.

Example: The driver "Videmech" is currently associated with serial port 1. A client sends this request, where "\r\n" has been omitted in the HTTP body.

```
POST /axis-cgi/ptz/ptzupdate.cgi HTTP/1.0\r\n
Content-Type: multipart/form-data; boundary=AaBo3x\r\n
Content-Length: <content length>\r\n\r\n
--AaBo3x
Content-Disposition: form-data; name="port"

1
--AaBo3x
```

```
Content-Disposition: form-data; name="Upload Driver"; filename="Philips-
1.0.ptz"
Content-Type: text/plain

<file content of Philips-1.0.ptz>

--AaBo3x--
```

I.e. providing the parameter port=1 together with the file content of the installation file Philips-1.0.ptz for the new driver in the HTTP body.

This will uninstall the driver "Videmech" and install the "Philips" driver provided in the HTTP body. The new driver is then associated with serial port 1.

5.3.2 PTZ administration

Handles relations between a driver and a serial port. When associating a driver with a serial port, any currently associated driver is first disassociated. The new driver is then activated to control units on that serial port. Any disassociated driver is automatically stopped.

Method: GET**/**POST

Syntax:

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
port=<*int*>	1, ... [1]	Selects the serial port.
ptzdrivername=<*string*>	*Name of a driver from values in parameter group root.PTZ.PTZDrivers*	Associates a driver name with a serial port. If *"none"* the current driver (if any) is disassociated from the serial port.

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

Example: Associate driver "Videmech" with serial port 2.

```
http://myserver/axis-cgi/com/ptzadmin.cgi?port=2&ptzdrivername=Videmech
```

5.3.3 PTZ control

To control the Pan, Tilt and Zoom behavior of a PTZ unit, the following PTZ control URL is used. This URL has view access rights.

Important:
Some PTZ units automatically reduce pan and tilt movements as the zoom factor increases. Therefore, the actual movement may be less than what is requested of these units.
The PTZ control is device-dependent. For information about supported parameters and actual parameter values, please check the specification of the Axis PTZ driver you intend to use. The following table is only an overview.

Note:
The URL must follow the standard way of writing a URL, (RFC 2396: Uniform Resource Identifiers (URI) Generic Syntax); that is, spaces and other reserved characters (";", "/", "?", ":", "@", "&", "=", "+", "," and "$") within a <parameter> or a <value> must be replaced with %<ASCII hex>. For example, in the string My camera, the space will have to be replaced with %20, My%20camera.

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.
		Digital: Used to send the coordinates for the point in the image where the user clicked. This information is then used by the server to center the clicked point.

areazoom=*<int>*,*<int>*,*<int>*	*<x>*,*<y>*,*<z>*	**Absolute:** Centers on positions *x*,*y* (like the *center* command) and zooms by a factor of *z*/100. If *z* is more than 100 the image is zoomed in (for example; z=300 zooms in to 1/3 of the current field of view). If *z* is less than 100 the image is zoomed out (for example; z=50 zooms out to twice the current field of view). **Relative:** n/a **Note 1:** In some camera models, the precision of *areazoom* can be strongly improved by calibrating the lens offset parameters. **Note 2:** The HTTP API for area zoom is currently only supported by Axis PTZ and Dome cameras.
imagewidth=*<int>*	1, ...[1]	Required in conjunction with *center* and *areazoom* if the image width displayed is different from the default size of the image, which is product-specific.
imageheight=*<int>*	1, ...[1]	Needed in conjunction with *center* and *areazoom* if the image height is different from the default size, which is product-specific.
move=*<string>*	home, up, down, left,	**Absolute:** Moves the device 5 degrees in the specified direction. **Relative:** Moves the device approx. 50-90 degrees[2] in

	right, upleft, upright, downleft, downright	the specified direction. **Digital:** Moves the image 25% of the image field width in the specified direction. **Note:** home is only valid if any home position has been previously set with "home=yes".
pan=<*float*>	-180.0 - 180.0	**Absolute:** Pans the device relative to the (0,0) position. **Relative:** n/a **Digital:** Pans the device relative to the (0,0) position.
tilt	-180.0 - 180.0	**Absolute:** Tilts the device relative to the (0,0) position. **Relative:** n/a **Digital:** Tilts the device relative to the (0,0) position.
zoom=<*int*>	1 - 9999	**Absolute:** Zooms the device *n* steps. **Relative:** n/a **Digital:** Zooms the device *n* steps.
focus=<*int*>	1 - 9999	**Absolute:** Move Focus *n* steps. **Relative:** n/a **Digital:** n/a
iris=<*int*>	1 - 9999	**Absolute:** Move iris *n* steps. **Relative:** n/a **Digital:** n/a
brightness=<*int*>[1]	1 - 9999	**Absolute:** Move brightness *n* steps. **Relative:** n/a **Digital:** 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. **Digital:** Pans the device *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. **Digital:** Tilts the device *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. **Digital:** Zooms the device *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 far, negative values mean focus near. **Relative:** Move Focus approx. *n* steps relative to the current position. Positive values mean focus far, negative values mean focus near. **Digital:** n/a
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. **Digital:** n/a
rbrightness=<*int*>[1]	-9999 - 9999	**Absolute:** Move brightness *n* steps relative to the current position. Positive values mean brighter image, negative values mean darker image. **Relative:** Move brightness approx. *n* steps relative to the current position. Positive values mean brighter image, negative values mean darker image. **Digital:** n/a
autofocus=<*string*>	on, off	Autofocus On/Off. **Digital:** n/a
autoiris=<*string*>	on, off	Autoiris On/Off. **Digital:** n/a
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. **Digital:** n/a

continuousirismove=<*int*>	-100 - 100	Continuous iris motion. Positive values mean iris open and negative values mean iris close. "0" means stop. **Digital:** n/a
continuousbrightnessmove=<*int*>[1]	-100 - 100	Continuous brightness motion. Positive values mean brighter image and negative values mean darker image. "0" means stop. **Digital:** n/a
auxiliary=<*string*>	<*function name*>	Activates/deactivates auxiliary functions of the device where <*function name*> is the name of the device-specific function. **Digital:** n/a
~~setserverpresetname=<*string*>~~	~~<*preset name*>~~[4]	~~Associates the current position to <*preset name*> as a preset position in the server.~~ **Note: This request is moved to ptzconfig.cgi**
~~setserverpresetno=<*int*>~~	~~1, ...~~	~~Saves the current position as a preset position number in the server.~~ **Note: This request is moved to ptzconfig.cgi**
~~home=<*string*>~~	~~yes~~	~~Makes the current position the home position for the camera. Used with *setserverpresetname* or *setserverpresetno.*~~ **Note: This request is moved to ptzconfig.cgi**
~~removeserverpresetname=<*string*>~~	~~<*preset name*>~~[4]	~~Removes the specified preset position associated with <*preset name*>.~~ **Note: This request is moved to ptzconfig.cgi**
~~removeserverpresetno=<*int*>~~	~~1, ...~~	~~Removes the specified preset position.~~ **Note: This request is moved to ptzconfig.cgi**
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.~~ **Note: This request is moved to ptzconfig.cgi**
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. **Digital:** n/a

bartype=<*string*>	absolute, relative	Used together with barcoord and determines how the bar shall be interpreted. If "absolute", the endpoints of the bar correspond to the current limits. If "relative", the behavior is device-dependent. The default interpretation is "*absolute*" for panbar, tiltbar and zoombar and "*relative*" for focusbar and irisbar. **Digital:** n/a (always "absolute").
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 "*horizontal*" 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 "*horizontal*" 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 "*horizontal*" 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 "*horizontal*" 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. **Digital:** n/a
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 "*horizontal*" 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. **Digital:** n/a
brightnessbar=<*int*>,<*string*>[1]	<*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 "*horizontal*" 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. **Digital:** n/a
speed=<*int*>	1 - 100	Sets the head speed of the device that is connected to the specified camera. **Digital:** n/a
imagerotation=<*int*>	0, 90, 180, 270	If PTZ command refers to an image stream that is rotated differently than the current image setup, then the image stream rotation must be added to each command with this parameter to allow the server to compensate.
query=<*string*>	speed, position, presetposcam, presetposall	Returns the current parameter values.
info=<*int*>	1	Returns a description of available PTZ commands. No PTZ control is performed.

[1] Product-dependent. Check the product's 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
```

5.3.4 PTZ configuration

Configure PTZ preset positions. On Screen Display (OSD) control.

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

Method: GET**/**POST

Syntax:

```
http://<servername>/axis-cgi/com/ptzconfig.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.
osdmenu=*<string>*	open, close, up, down, left, right, select, back	Commands to control the OSD menu in the camera. Note that the support of the different commands, and the behavior of the commands, is driver dependant. **Digital:** n/a
setserverpresetname=*<string>*	*<preset name>*[1]	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.
home=*<string>*	yes	Makes the current position the home position for the camera. Used with *setserverpresetname* or *setserverpresetno.*
removeserverpresetname=*<string>*	*<preset name>*[1]	Removes the specified preset position associated with *<preset name>*.
removeserverpresetno=*<int>*	1, ...	Removes the specified preset position.
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. **Digital:** n/a

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

5.3.5 Set PTZ parameters

Set PTZ parameters. Listing the PTZ parameters is done with param.cgi, List parameters.

Method: GET**/**POST

Syntax:

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
port=*<int>*	1, ... [1]	Selects the serial port. Used together with *grpdefault* below, for group "*Serial*".
camera=*<int>*	1, ... [1]	Selects the camera. If omitted, the default camera is used. Used together with *grpdefault* below, for all groups but "*Serial*".
<PTZ	Any parameter in the	Sets a PTZ limit parameter. Max values are adjusted so that

parameter>[2]=*<value>*	groups PTZ.Limit.L#, where # denotes the camera number, e.g. PTZ_Limit_L2_MaxPan	they are greater than or equal to min values, e.g. PTZ.Limit.L2.MaxPan >= PTZ.Limit.L2.MinPan.
pardefault=*<string>*[2]	Any parameter in the group: Serial.Ser# where # denotes the serial port number, and the groups: PTZ.Support.S# PTZ.Limit.L# PTZ.Various.V# PTZ.UserBasic.U# PTZ.UserAdv.U# where # denotes the camera number.	Sets the default value for a parameter. Note: No max value validation is performed as with previous CGI parameter.
grpdefault=*<string>*	Serial, Support, Limit, Various, UserBasic, UserAdv	Sets the default value for a parameter group. The default values are taken from the driver currently associated with the port or camera.

[1] Product-dependent. Check the product's specification.
[2] The path shall have the format root_PTZ..., e.g. root_PTZ_Limit_L1_MaxPan for parameter PTZ.Limit.L1.MaxPan. The reason for this format is related to internal ssi implementation.

In the examples below, assume the driver installed on index 1 is associated with serial port 2 (ttyS1) and camera 3 is mapped to serial port 2.

Example: Set PTZ.Limit.L3.MaxPan to 110 and adjust it so that it is >= PTZ.Limit.L3.MinPan.

```
http://myserver/axis-cgi/com/ptzparam.cgi?root_PTZ_Limit_L3_MaxPan=110
```

Example: Set PTZ.Limit.L3.MaxPan to the value of PTZ.Driver1.Limit.L0.MaxPan.

```
http://myserver/axis-cgi/com/ptzparam.cgi?pardefault=root_PTZ_Limit_L3_MaxPan
```

Example: Copy the parameter values from group PTZ.Driver1.Limit.L0 to group PTZ.Limit.L3.

```
http://myserver/axis-cgi/com/ptzparam.cgi?camera=3&grpdefault=Limit
```

Example: Copy the parameter values from group PTZ.Driver1.Serial.S0 to group Serial.Ser1.

```
http://myserver/axis-cgi/com/ptzparam.cgi?port=2&grpdefault=Serial
```

5.3.6 PTZ control queue

If the PTZ control queuing mechanism is enabled for the camera concerned, control of PTZ units is limited to the client currently possessing the control, if any. This CGI request handles requests concerning the control queue. Note that cookies are enabled by default when enabling PTZ control queue.

Method: GET**/**POST

Syntax:

```
http://<servername>/axis-cgi/com/ptzqueue.cgi?
```

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
control=*<string>*	request, drop, query	"request" requests PTZ control. "drop" drops the PTZ control or leaves the queue. "query" reports the current status for the client. For possessing clients with no peers existing in the queue, "request" will reset the control timer. For all other clients, "request" will have the same effect as "query".
camera=*<int>*	1, ... [1]	The camera number. If omitted, the default camera is used.

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

Example: Request PTZ control for camera 2.

```
http://myserver/axis-cgi/com/ptzqueue.cgi?control=request&camera=2
```

5.3.7 PTZ control queue response

The 200 OK response on success for "request" and "query" has a format that enables simple java-script implementations using the DOM API and should be easy to parse for components such as ActiveX as well.

Return: A successful *request*:

```
HTTP/1.0 200 OK\r\n
Content-Length: <length>\r\n
\r\n
<body><a name="<pos>"></a><a name="<seconds>"></a><a
name="<period>"></a></body>
```

where

<pos> can have a value from 0 to the maximum value of how many clients are allowed in the queue. This value is the given position in the queue. 0 means that the client is not in the queue, 1 means control is possessed. For 0 the other parameters are undefined.

< seconds> is the estimated number of seconds remaining, i.e. for position 1 the remaining control time and for other positions, the time until position 1 is reached. -1 denotes that the time remaining to get control cannot be estimated. This means that a client in the queue has the "TimeoutType" set to infinity.

< period> is the recommended time in seconds when the client should send a new "control=query" requests. To stay active in the queue the client must regularly send PTZ requests to the camera. An inactive client will automatically be removed from the queue.

On failure no anchor elements are provided but simply the error message in plain text.

Return: A failed *request*:

```
HTTP/1.0 200 OK\r\n
Content-Length: <length>\r\n
\r\n
Error:\r\n
<error information>
```

Example: Control requested.

```
HTTP/1.0 200 OK\r\n
Content-Length: 63\r\n
\r\n
<body><a name="3"></a><a name="410"></a><a name="5"></a></body>
```

This means the client was assigned queue position 3. The expected number of seconds until control is possessed is 410 and the recommended time until the next request is 5 seconds.

5.4 Motion Detection

To be able to define Motion Detection parameters, the video product must have built-in Motion Detection.
A motion detection window is defined by several parameters. The motion detection parameters reside within a dynamic parameter group. Accordingly it is possible to add, remove, list and update the motion detection parameters with param.cgi, The dynamic motion detection parameter groups are divided into sub groups of the main motion parameter group, i.e. Motion.M<group number>.<parameter name>. group number is a unique number which is stated when a new dynamic parameter group is created, i.e. Motion.M3.

5.4.1 Add a Motion Detection window

When adding a Motion Detection window, the template file motion is used. The group number should be excluded when adding a new Motion Detection window with specified values since the group number will be defined when the new dynamic group is created.

Example: Add a new Motion Detection window with default values.

```
http://myserver/axis-
cgi/operator/param.cgi?action=add&group=Motion&template=motion
```

Example: Add a new Motion Detection window with specified values.

```
http://myserver/axis-
cgi/operator/param.cgi?action=add&group=Motion&template=motion
&Motion.M.Name=Entrance&Motion.M.Top=500&Motion.M.Bottom=7000&Motion.M.Left=5
000
&Motion.M.Right=8500
```

Return:

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

5.4.2 Remove a Motion Detection window

Example: Remove Motion Detection window defined within Motion.M3 and Motion.M5.

```
http://myserver/axis-cgi/operator/param.cgi?
action=remove&group=Motion.M3,group=Motion.M5
```

5.4.3 Update the Motion Detection parameters

Example: Update the parameters for an existing Motion Detection window.

```
http://myserver/axis-cgi/operator/param.cgi?action=update&Motion.M1.Top=1500
```

```
&Motion.M1.Bottom=8000
```

5.4.4 List the Motion Detection parameters

Example: List the Motion.M1 and Motion.M2 parameters.

```
http://myserver/axis-cgi/operator/param.cgi?
action=list&group=Motion.M1,group=Motion.M2
```

Example: List all Motion Detection windows.

```
http://myserver/axis-cgi/operator/param.cgi?action=list&group=Motion
```

5.4.5 Get the Motion Detection level

It is possible to get the current Motion Detection levels from certain Motion Detection windows or from all MD windows, except from those windows that are defined to be Motion Detection exclude windows. It is also possible to define a Motion Detection window configuration and get the related motion levels in return. The URL stated below is used.

Method: GET/POST

Syntax:

```
http://<servername>/axis-cgi/motion/motiondata.cgi[?<parameter>=<value>...]
```

with the following parameter and value

<parameter>=<value>	**Values**	**Description**
group=<*int*>[,<*int*>, ...]	<*group number*>[,<*group number*>, ...]	Specify the Motion Detection windows that are of interest. Excluding the group parameter will return all Motion Detection level information from all Motion Detection windows. Exclude windows are ignored.
<*string*>=<*string*>	<*parameter name*>=<*value*>	Get the Motion Detection levels related to a specific window configuration. The last part of the parameters in the parameter group Motion.M#.<name> (where # is a number) can be used, e.g. Sensitivity and History.

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 with lowest group number>
--<boundary>\r\n
```

and <*motion level for window with group number* n>" is

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

Example: Get Motion Detection levels related to Motion Detection windows defined within Motion.M0 and Motion.M1.

```
http://myserver/axis-cgi/motion/motiondata.cgi?group=0,1
```

Return: The example above returns the following.

```
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
group=0;level=28;threshold=45;\r\n
group=1;level=43;threshold=25;\r\n
--axismdb\r\n
Content-Type:text/plain\r\n\r\n
group=0;level=54;threshold=45;\r\n
group=1;level=38;threshold=25;\r\n
--axismdb\r\n
Content-Type:text/plain\r\n\r\n
group=0;level=49;threshold=45;\r\n
group=1;level=19;threshold=25;\r\n
--axismdb\r\n
 .
 .
 .
```

Example: Get Motion Detection levels related to a specific window configuration. The group number related to this Motion Detection window configuration will always be X.

```
http://myserver/axis-cgi/motion/motiondata.cgi?
Name=Temp&ImageSource=2&WindowType=include
&Left=300&Right=500&Top=300&Bottom=500&Sensitivity=65&History=50&Size=35
```

Return: The example above returns the following:

```
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
group=X;level=16;threshold=35;\r\n
--axismdb\r\n
Content-Type:text/plain\r\n\r\n
group=X;level=55;threshold=35;\r\n
--axismdb\r\n
Content-Type:text/plain\r\n\r\n
group=X;level=39;threshold=35;\r\n
--axismdb\r\n
 .
 .
 .
```

- If no Motion Detection windows are defined or only exclude windows are defined, HTTP/1.0 204 No Content is returned.
- If any errors are found in the CGI request, HTTP/1.0 400 Bad Request is returned.
- If too many clients try to get motion data, HTTP/1.0 503 Service Unavailable is returned.

5.5 I/O

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

5.5.1 I/O control

5.5.1.1 Input

Input

Method: GET

Syntax:

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
check=<*int*>[,<*int*>, ...]	<*id1*>[,<*id2*>, ...] [1]	Returns the status (1 or 0) of one or more inputs numbered *id1 ,id2, ...*
checkactive=<*int*>[,<*int*>, ...]	<*id1*>[,<*id2*>, ...] [1]	Returns the status (active or inactive) of one or more inputs numbered *id1,id2, ...*
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 cameras and video servers. See the product's 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>:<char>\r\n
```

and <*n*> is the I/O port number and <*char*> is / or H when the port is active and \ or L when the port is inactive.
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
 .
 .
 .
```

5.5.1.2 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 (1 or 0) of one or more outputs numbered *id1 ,id2, ...*.
checkactive=<*int*>[,<*int*>, ...]	<*id1*>[,<*id2*>, ...] [1]	Returns the status (active or inactive) of one or more outputs numbered *id1 ,id2, ...*.
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*> active or inactive and waits <*wait*> milliseconds. Note that only one output relay can be activated/deactivated per request. <*id*> = Output number. If omitted, output *1* is selected. <*a*> = Action character: / or \ / = active, \ = inactive. <*wait*> = Delay in milliseconds.

[1] Number of outputs may differ for different cameras and video servers. See the product's specification.
[2] Support for this parameter is product/release-dependent.

Example: Set output 1 active.

```
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 active.

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

5.5.2 Virtual I/O control

5.5.2.1 Input

Input

Method: GET

Syntax:

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

with the following parameter and values

<parameter>=<value>	**Values**	**Description**
action=*<string>*	[*<id>*]:*<a>*	Simulates an activation or inactivation of an Input connector or sets a virtual input *<id>* active or inactive. *<id>* = Input number. If omitted, input *1* is selected. *<a>* = Action character: / or \ / = active, \ = inactive.

Example: Set Input 1 active.

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

Example: Pulse the Input. First set it active, wait 2 seconds, then set it inactive again.

```
http://myserver/axis-cgi/io/virtualinput.cgi?action=1:/2000\
```

5.6 Serial Port

The requests specified in the Serial Port section are supported by products with PTZ support.

5.6.1 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]	Select COM port. -1 means disconnect the camera from the serial port.

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's specification.
[2] Obsolete.

5.6.2 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. -1 = disconnect the camera from any serial 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	Instructs the server to keep the connection open and serve as a link between the client and the serial port.

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

Example: Open serial port 1 if not already opened and connect camera 2 to it.

```
http://myserver/axis-cgi/com/serial.cgi?port=1&camera=2
```

Example: Disconnect camera 2 from any serial port (but keep that port open).

```
http://myserver/axis-cgi/com/serial.cgi?port=-1&camera=2
```

5.7 IP filter

The requests specified in the IP filter section are supported by products that support IP address filtering.

5.7.1 IP address filter administration

Allow or deny the listed IP addresses to access the Axis device.

Method: GET

Syntax:

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
action=*<string>*	add, remove, removeall, update or list	Specifies the action to take. add = Add new IP address (or addresses). remove = remove an entry in the IP address list. removeall = Remove all IP addresses. The IP address filtering function will automatically be disabled. update = Update settings for the IP address filtering function. list = List the settings for the IP address filtering function.
ipaddress=*<string>[%20<string>...]*	*<IP addresses>*	The addresses allowed passing through the filter. A space separated list of IP addresses and network addresses in the CIDR notation (IP address/netmask bits). Note: If accessing the Axis device via a proxy server, the proxy server's IP address must be added to the list of allowed addresses.
enable=*<string>*	yes, no	Enable/disable the IP address filtering function.
policy=*<string>*	allow, deny	Allow or deny access for the addresses in the list. If omitting this parameter the policy will be allow by default. **Note:** The policy used will apply for all the addresses in the list!
verify=*<string>*	yes, no	Verify that you are able to access the Axis device with the new settings. A failed verification will make the settings to remain as before. verify is automatically set to yes when using requests that risk to make the device inaccessible, i.e. setting enable to yes, or adding, updating and removing IP addresses when the filter function is enabled. In the same way verify is by default set to no when using requests that do not risk to make the device inaccessible, i.e. listing filter settings or adding, updating and removing IP addresses when the filter function is disabled. This behaviour can be changed by

		setting this parameter in the request. yes = Use verification. no = Do not use verification. **Note: setting verify to no, may cause the Axis device to be inaccessible.**

Example: Add a list of IP addresses and enable the IP address filtering function. Verification that the device is still accessible will automatically be done.

```
http://myserver/axis-cgi/admin/ipfilter.cgi?action=add
&ipaddress=10.13.10.12%2010.13.17.0/24&enable=yes
```

Example: List settings for the IP address filtering function.

```
http://myserver/axis-cgi/admin/ipfilter.cgi?action=list
```

Example: Remove an entry in the list of addresses. Verification will automatically be done if the IP filter function is enabled.

```
http://myserver/axis-
cgi/admin/ipfilter.cgi?action=remove&ipaddress=10.13.10.12
```

Example: Add 10.13.10.12 to the list of addresses which will be allowed access to the device.

```
http://myserver/axis-cgi/admin/ipfilter.cgi?action=add&ipaddress=10.13.10.12
&policy=allow&enable=yes
```

Example: Add 10.13.10.12 to the list of addresses which will be denied access to the device.

```
http://myserver/axis-cgi/admin/ipfilter.cgi?action=add&ipaddress=10.13.10.12
&policy=deny&enable=yes
```

Example: Remove all IP addresses and automatically disable the IP address filtering function

```
http://myserver/axis-cgi/admin/ipfilter.cgi?action=removeall
```

5.7.2 Server responses

Return: A successful *add, remove, removeall or update.*

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

Return: A list.

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

```
\r\n
Accept addresses: <IP addresses>\r\n
Enabled: <yes/no>\r\n
```

Return: Verification failed. The settings did not take place.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Verification failed: IP address "<IP address>" is not an accepted
address\r\n
```

Example: List settings for the IP address filtering function, set verify to yes to check that my computers IP address is accepted.

```
http://myserver/axis-cgi/admin/ipfilter.cgi?action=list&verify=yes
```

Response: Only the IP address 10.13.10.12 is an accepted address. The computer 10.13.17.245 will not be able to access the device if enabling the IP address filtering function.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Accept addresses: 10.13.10.12\r\n
Enabled: no\r\n
Verification failed: IP address "10.13.17.245" is not an accepted address
```

5.8 Audio

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

5.8.1 Audio MIME types

Supported MIME types for audio

`audio/basic`	which is G.711 µ-law 64kbit/s
`audio/32KADPCM`	which is G.726 32kbit/s
`audio/G723`	which is G.726 24kbit/s

5.8.2 Audio data request

Request an audio stream.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/audio/receive.cgi[?<parameter>=<value>]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
httptype=<*string*>	singlepart, multipart	Choose streaming method. Default value is defined by the parameter Audiod.HttpMessageType

Example: Request a singlepart audio stream

```
http://myserver/axis-cgi/audio/receive.cgi?httptype=singlepart
```

5.8.3 Singlepart audio data response

When an audio stream is requested/transmitted, the server returns/receives a continuous flow of audio packets. The content type is only set at the beginning of the connection. When the connection is up and running the audio packets will come right after another without any extra information between the packets. The message body contains a block of binary data. Each block of coded audio data is 240 byte.

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: <audio MIME>\r\n
\r\n
<Audio data>
```

Example: Singlepart Audio data encoded with G.711 µ-law.

```
HTTP/1.0 200 OK\r\n
Content-Type: audio/basic\r\n
\r\n
<Audio data>
<Audio data>
<Audio data>
.
.
.
```

5.8.4 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 "mutipart/x-mixed-replace" and each audio packet ends with a boundary string. The message body contains a block of binary data. Each block of coded audio data is 240 byte.

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 MIME>\r\n
\r\n
<Audio data>\r\n
--<boundary>\r\n
<audio>
```

Example: Multipart Audio data encoded with G.726 32kbit/s (G.721).

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

5.8.5 Audio data transmit

Transmit a Singlepart/Multipart Audio data stream. Each block of coded audio data is 240 byte.

Method: POST

Syntax:

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

There are no valid parameters and values.

Example 1: Singlepart audio data transmit with G.711 µ-law (authorization omitted)

```
POST /axis-cgi/audio/transmit.cgi HTTP/1.0\r\n
Content-Type: audio/basic\r\n
Content-Length: 9999999\r\n
Connection: Keep-Alive\r\n
Cache-Control: no-cache\r\n
\r\n
<Audio data>
```

```
<Audio data>
<Audio data>
.
.
.
```

Example 2: Multipart audio data transmit with G.711 µ-law (authorization omitted)

```
POST /axis-cgi/audio/transmit.cgi HTTP/1.0\r\n
Content-Type: multipart/x-mixed-replace; boundary=--myboundary\r\n
Content-Length: 9999999\r\n
Connection: Keep-Alive\r\n
Cache-Control: no-cache\r\n
\r\n
--myboundary\r\n
Content-Type: audio/basic\r\n
\r\n
<Audio data>\r\n
--myboundary\r\n
Content-Type: audio/basic\r\n
\r\n
<Audio data>
--myboundary\r\n
Content-Type: audio/basic\r\n
\r\n
<Audio data>
--myboundary\r\n
Content-Type: audio/basic\r\n
\r\n
.
.
.
```

5.9 AXIS 292 Network Video Decoder

5.9.1 Alarm

When the decoder operates in manual mode it can receive alarms from an encoder. This can be a motion detected or input triggered event on the encoder. The decoder will automatically switch to the encoder <sourcename> when receiving an alarm, where <sourcename> is the name of the given encoder in the video source list. The On Screen Display (OSD) will display "ALARM: <sourcename>" and an optional line with a text message of maximum 40 characters.

Note: The alarm function can only be used in manual mode and requires administrator access.

Method: GET

Syntax:

```
http://<servername>/axis-
cgi/admin/alarm.cgi?sourcename=<camera>&textmessage=<text>
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
sourcename=<*camera*>	A string	Reflects the same name found in Encoder.E#.Name.

textmessage=<*string*>	A string	The text message to be displayed on the screen. Maximum 40 characters will be displayed. Note that this message must be URI encoded.

Example: Send an alarm from camera1

```
http://myserver/axis-cgi/admin/alarm.cgi?sourcename=camera1
```

Example: Send an alarm from camera3 with the message "Door is open"

```
http://myserver/axis-cgi/admin/alarm.cgi?sourcename=camera3
&textmessage=Door%20is%20open
```

Response: Response from a successful alarm

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
Expires: 0\r\n
Pragma: no-cache\r\n
\r\n
OK\r\n
```

Response: Response from an unsuccessful alarm with unknown sourcename

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
Expires: 0\r\n
Pragma: no-cache\r\n
\r\n
# Error: Unknown sourcename\r\n
```

Response: Response from an unsuccessful alarm with sourcename missing

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
Expires: 0\r\n
Pragma: no-cache\r\n
\r\n
# Error: sourcename must be specified\r\n
```

5.9.2 Video control

Handles various video states such as Connect, Disconnect, Invalidatecache and Goto.

Note: This requires administrator access.

Method: GET

Syntax:

```
http://<servername>/axis-cgi/admin/videocontrol.cgi?action=<string>
```

Response: Response from a videocontrol request with no action specified

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

```
Content-Type: text/plain\r\n
Expires: 0\r\n
Pragma: no-cache\r\n
\r\n
# Error: Action value is missing. Supported actions are invalidatecache,
goto, connect,
disconnect\r\n
```

5.9.2.1 Connect

The video decoder will connect to the first source in the video source list.

Syntax:

```
http://<servername>/axis-cgi/admin/videocontrol.cgi?action=connect
```

Example: Connect the Network Video Decoder

```
http://myserver/axis-cgi/admin/videocontrol.cgi?action=connect
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
Expires: 0\r\n
Pragma: no-cache\r\n
\r\n
OK\r\n
```

5.9.2.2 Disconnect

Disconnect the Network Video Decoder.

Syntax:

```
http://<servername>/axis-cgi/admin/videocontrol.cgi?action=disconnect
```

Example: Disconnect the Network Video Decoder

```
http://myserver/axis-cgi/admin/videocontrol.cgi?action=disconnect
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
Expires: 0\r\n
Pragma: no-cache\r\n
\r\n
OK\r\n
```

5.9.2.3 Invalidate Cache

Removes all cached information. This cache contains information about the video sources. Values that are cached can be found in the Axis Video Parameter document under Encoder.E#.Info. When connecting to a source the second time, the decoder will use the information found in this cache for how to connect, instead of doing an autodetect.

Syntax:

```
http://<servername>/axis-cgi/admin/videocontrol.cgi?action=invalidatecache
```

Example: Remove cached information

```
http://myserver/axis-cgi/admin/videocontrol.cgi?action=invalidatecache
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
Expires: 0\r\n
Pragma: no-cache\r\n
\r\n
OK\r\n
```

5.9.2.4 Select source

The decoder will connect to a specified video source.

Syntax:

```
http://<servername>/axis-
cgi/admin/videocontrol.cgi?action=goto&sourcename=<name>
```

Example: Go to the video source named "camera1"

```
http://myserver/axis-
cgi/admin/videocontrol.cgi?action=goto&sourcename=camera1
```

Response: A successful goto.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
Expires: 0\r\n
Pragma: no-cache\r\n
\r\n
OK\r\n
```

Response: Goto failed, the video source name was wrong.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
Expires: 0\r\n
Pragma: no-cache\r\n
\r\n
# Error: Unknown sourcename\r\n
```

Response: Goto failed, the video source name was missing

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
Expires: 0\r\n
Pragma: no-cache\r\n
\r\n
# Error: sourcename must be specified\r\n
```