IP Camera Standard API - Interface Specification

Revision: 2.03

Date: 2010-Apr-09

TABLE OF CONTENTS

DOCUMENT HISTORY

Version	Date	Comment
2.01	2009-Aug-27	Initial version.
2.02	2009-Dec-18	Add 5.5.1 2nd H.264 Streaming Syntax Add 5.1.13 Time Zone Setting Add 5.3.1 PTZ Control Setting
2.03	2009-Mar-22	Add 5.1.7 Backup Add 5.1.8 Restore

1 OVERVIEW

This document specifies the external HTTP-based application programming interface of the IP camera.

The HTTP-based video interface provides the functionality for requesting 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.

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)

- IP Camera API parameters

RTSP Protocol

- Real Time Streaming Protocol - RFC 2326

SDP Protocol

- Session Description Protocol - RFC 2327

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).
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. 192.168.0.250.

Syntax:

```
http://<servername>/cgi-bin/admin/userinfo.cgi
```

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 user privacy.

```
http://myserver/cgi-bin/admin/privacy.cgi
```

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 cgi-bin directory. The file extension of the CGI is required.

Syntax:

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

Example: List the Network parameters.

```
http://<servername>/cgi-bin/admin/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 Server responses

4.1.1 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.

5.1 General

The requests specified in the General section are supported by all video products with firmware version z20070921 and below.

5.1.1 Update and list parameters and their values

Note:

- The 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*>/cgi-bin/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*>/cgi-bin/admin/param.cgi?action=list [&<*parameter*>=<*value*>...]

with the following parameter and values

<parameter>=<value>	**Values**	**Description**
group=<*string*>[& group=<*string*>...]	<*group*[.*name*]>[,<*group*[.*name*]>...]	Returns the value of the camera parameter named <*group*>.<*name*>. The camera parameters must be entered exactly as they are named in the camera or video server.

Example: List the Network parameters.

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

Example: List the names of all Event parameters and Network parameters

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

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=192.168.0.250\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>/cgi-bin/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 exposure mode to auto.

```
http://myserver/cgi-bin/admin/param.cgi?
action=update&ImageSource.I0.Sensor.Exposure=auto
```

Example: Set the event enable.

```
http://myserver/cgi-bin/admin/param.cgi?
action=update& Event.E0.Enabled=yes
```

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>/cgi-bin/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.
sgrp=<*string*>:[<*string*>...]	<*string*>[,<*string*>...]	Colon separated existing secondary group names of the account. Ex: dido：camctrl：talk：listen

Example: Create a new administrator account.

```
http://myserver/cgi-bin/admin/pwdgrp.cgi?action=add&user=joe&pwd=foo&sgrp=dido:camctrl:talk:listen
```

Example: Change the password of an existing account.

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

Example: Remove an account.

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

Example: List groups and users.

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

5.1.3 List users information

List the user information with password or privacy.

Method: GET

Syntax:

```
http://<servername>/cgi-bin/admin/privacy.cgi
```

Example: List the username and privacy

```
http://myserver/cgi-bin/admin/privacy.cgi?
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
Username:Dido:Camset:Talk:Listen \n
Admin:1:1:1:1\n
```

Syntax:

```
http://<servername>/cgi-bin/admin/userinfo.cgi
```

Example: List the username and password.

```
http://myserver/cgi-bin/admin/userinfo.cgi?
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
List username and password\n
Admin:1234\n
```

5.1.4 Get, modify snapshot path

Get or modify Admin snapshot path, Admin can capture images by the web page snapshot button, and the images are stored at the path you set.

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

http://<*servername*>/cgi-bin/admin/snapshot.cgi? <*parameter*>=<*value*>]

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
path=<*string*>	<string>	Valid character: A-Za-z0-9and some special tokens _/\~!@#$%^&+-:
action=<string>	get set	Get the Admin snapshot path. Set the Admin snapshot path, with action=set parameter path is required.

Example: Set the Admin snapshot path to C:\capture

```
http://myserver/cgi-bin/admin/snapshot.cgi?action=set&path=C%3A%5Ccapture
```

Response:

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

5.1.5 Factory default

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

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

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

5.1.6 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>/cgi-bin/admin/hardfactorydefault.cgi
```

5.1.7 Backup

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

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

Method: GET

Syntax:

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

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: application/octet-stream\r\n
Content-length: 15899\r\n
Content-Disposition: attachment; filename=config_file.bin\r\n
<file content of config_file.bin >
```

5.1.8 Restore

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

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

Method: POST

Syntax:

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

The file 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.”

Response: Upload of backup, where “\r\n” has been omitted in the HTTP body.

```
POST /cgi-bin/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=" config_file.bin ";
filename=" config_file.bin "\r\n
Content-Type: application/octet-stream\r\n
\r\n
<file content of config_file.bin>
\r\n
--AaBo3x--\r\n
```

5.1.9 Firmware upgrade

5.1.9.1 Before firmware upgrade

It will stop some process (like stream server, image transfer .. etc) to prepare firmware upgrade.

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

Method: GET

Syntax:

```
http://<servername>/cgi-bin/admin/beforeupgrade.cgi
```

5.1.9.2 Start firmware upgrade

Upgrade the firmware version.

Note: This requires administrator access (administrator authorization).

Method: POST

Syntax:

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
filename=<*string*>	uImage, cameraFw, userland.jffs2, var	Specifies the filename of firmware upgrade. uImage = kernel package binary file. cameraFw = camera parameters binary file. userland.jffs2 = JFFS2 image binary file. var = variable binary file.

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 /cgi-bin/admin/firmwareupgrade.cgi?filename=userland.jffs2 HTTP/1.0\r\n
Content-Type: multipart/form-data; boundary=AsCg5y\r\n
Content-Length: <content length>\r\n
```

```
Authorization: Basic QWRtaW46MTIzNA==
\r\n
--AsCg5y\r\n
Content-Disposition: form-data; name="userland.jffs2"; uploadfile="userland.jffs2"\r\n
Content-Type: application/octet-stream\r\n
\r\n
<firmware file content>
\r\n
--AsCg5y--\r\n
```

5.1.10 Restart server

Restart server.

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

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

5.1.11 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>/cgi-bin/admin/serverreport.cgi
```

5.1.12 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>/cgi-bin/admin/systemlog.cgi
```

Return:

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

5.1.13 System date and time

Get or set the system date and time.

Method: GET/POST

Syntax:

```
http://<servername>/cgi-bin/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.

5.1.13.1 Get system date and time

Syntax:

```
http://<servername>/cgi-bin/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
```

5.1.13.2 Set system date and time

Syntax:

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

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
year=<*int*>	2007 - 2099	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-12…GMT+13	Time zone.

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/cgi-bin/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

5.2.1 MJPEG images (snapshot) CGI request

Method : GET

Syntax :

```
http://<servername>/cgi-bin/jpg/image.cgi?
```

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
```

5.3 PTZ

5.3.1 PTZ

Provide CGI commands for PTZ function control.

5.3.1.1 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. 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*>/cgi-bin/com/ptz.cgi?<*parameter*>=<*value*>[&<*parameter*>=<*value*>...]

with the following parameters and values

Parameter name	Default value	Valid values	Description
move		home up down left	Moves the device 5 degrees in the specific direction.

		right upleft upright downleft downright	
pan		-180.0 to 180.0	Pans the device relative to the (0, 0)position
tilt		-180.0 to 180.0	Tilts the device relative to the (0, 0)position
zoom		0 to 9999	Zoom the device n steps
focus		0 to 9999	Move focus n steps
rpan		-360.0 to 360.0	Pans the device n degrees relative to the current position
rtilt		-360.0 to 360.0	Tilts the device n degrees relative to the current position
rzoom		-9999 to 9999	Zoom the device n steps relative to the current position; Positive values mean zoom in, and negative values mean zoom out.
rfocus		-9999 to 9999	Move device n steps relative to the current position; Positive values mean focus far, and negative values mean focus near.
autofocus		on, off	Autofocus on/off.
continuouspantiltmove		-100 to 100, -100 to 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.
continuouszoommove		-100 to 100	Continuous zoom motion. Positive values mean zoom in and negative values mean zoom out. "0" means stop.
continuousfocusmove		-100 to 100	Continuous focus motion. Positive values mean

			focus near and negative values mean focus far. "0" means stop.
gotoserverpresetno		1, ...	Move to the position associated with the specified preset position number.
query		speed position zoomratio	Returns the current parameter values.
info	1	1	Returns a description of available PTZ commands.

Example: Request information about which PTZ commands are available.

```
http://myserver/cgi-bin/com/ptz.cgi?info=1
```

5.3.1.3 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>/cgi-bin/com/ptzconfig.cgi?
<parameter>=<value>[&<parameter>=<value>...]
```

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
setserverpresetname=<int>,<*string*>	<preset no><*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.
removeserverpresetname=<*string*>	<*preset name*>[1]	Removes the specified preset position associated with <*preset name*>.
removeserverpresetno=<*int*>	1, ...	Removes the specified preset position.
setserverautopan=<int>,<string>	<autopan line>,<state>	
setserverautopandirspeed=<int>,<string>,<int>	<autopanline>, <direction>, <speed>	Set autopan direction and speed. <direction>:left,right <speed>:0-3
setservercruise=<int>,<string>	<cruise line>,<state>	Set cruise line. <state> start : start cruise setting end : end cruise setting

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

5.4 I/O

The requests in the I/O section are supported by the products with Input/Output functions

5.4.1 I/O control

5.4.1.1 Input

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

http://<*servername*>/cgi-bin/io/input.cgi? <*parameter*>=<*value*>

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
check=<int>[,<int>,…]	<id1>[,<id2>…]	Returns the status (1 or 0) of one of more inputs numbered id1, id2,…
checkactive=<int>[,<int>,…]	<id1>[,<id2>…]	Returns the status (active or inactive) of one or more inputs numbered id1,id2

Number of inputs may be different according to the camera model. Please see the product's specification

```
http://myserver/cgi-bin/io/input.cgi?check=1
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Input1=0
```

5.4.1.2 Output

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

http://<*servername*>/cgi-bin/io/output.cgi? <*parameter*>=<*value*>

with the following parameters and values

<parameter>=<value>	**Values**	**Description**
check=<int>[,<int>,...]	<id1>[,<id2>...]	Returns the status (1 or 0) of one of more outputs numbered id1, id2,...
checkactive=<int>[,<int>,...]	<id1>[,<id2>...]	Returns the status (active or inactive) of one or more outputs numbered id1,id2
action=<string>	<id1>:<a>	<id> = Output number. If omitted, output 1 is selected. <a> = Action character: / of \ /=active, \=inactive

Number of outputs may be different according to the camera model. Please see the product's specification

Example: Set output 1 active

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

Response:

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

5.5 Video and Audio

5.5.1 Connect video and audio stream

Connect a video and audio stream by UDP or TCP with default resolution and compression as defined in the system configuration.

Syntax: connect to H.264

```
rtsp://<servername>/h264
```

Syntax: connect to 2nd H.264 streaming in dual H.264 mode.

```
rtsp://<servername>/h264_2
```

Syntax: connect to MPEG-4

```
rtsp://<servername>/mpeg4
```

Syntax: connect to MJPEG

```
rtsp://<servername>/jpeg
```

5.5.2 Connect video stream by http

Connect a video stream by HTTP with default resolution and compression as defined in the system configuration.

Syntax: connect to MJPEG

```
http://<servername>:8008
```

5.5.3 RTSP

This document specifies the external RTSP-based application programming interface of the camera and video servers.

The RTSP URL is rtsp://<server name>/mpeg4 where <server name> is the host name or IP address of the server. The DESCRIBE, SETUP, OPTIONS, PLAY, PAUSE and TEARDOWN methods are supported. The RTSP protocol is described in RFC 2326.

Request syntax:

COMMAND URI RTSP/1.0<CRLF>
Headerfield1: val1<CRLF>
Headerfield2: val2<CRLF>
...
<CRLF>

Response syntax:

RTSP/1.0 *ResultCode ResultString*<CRLF>
Headerfield3: val3<CRLF>
Headerfield4: val4<CRLF>
...
<CRLF>

The following header fields are accepted by all commands. Other header fields are silently ignored (unless stated otherwise in the sections below).

Field	Description
CSeq	Request sequence number.
Session	Session identifier (returned by server in SETUP response).
Content-Length	Length of content.

The following header fields can be generated for all responses by the RTSP server:

Field	Description

CSeq	Response sequence number (matches the sequence number of the request).
Session	Session identifier.

5.5.3.1 RTSP DESCRIBE

The DESCRIBE command returns the SDP (RFC 2327) description for the URI. The DESCRIBE command accepts the following additional header field:

Accept	List of content types that client supports (application/sdp is the only supported type).

The DESCRIBE command generates the following additional header fields:

Content-Type	Type of content (application/sdp).
Content-Length	Length of SDP description.
Content-Base	If relative URLs are used in the SDP description, then this is the base URL.

Example:

```
DESCRIBE rtsp://192.168.0.200/mpeg4 RTSP/1.0
CSeq: 0
Accept: application/sdp
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 0
Date: Fri, Jan 05 2007 18:10:03 GMT
Content-Base: rtsp://192.168.0.200/mpeg4/
Content-Type: application/sdp
Content-Length: 572
```

```
v=0
o=- 1168019957830000 1 IN IP4 0.0.0.0
s=Session streamed by "nessyMediaServer"
i=mpeg4
t=0 0
a=tool:LIVE555 Streaming Media v2008.04.09
a=type:broadcast
a=control:*
a=range:npt=0-
a=x-qt-text-nam:Session streamed by "nessyMediaServer"
a=x-qt-text-inf:mpeg4
m=video 0 RTP/AVP 96
c=IN IP4 0.0.0.0
a=rtpmap:96 MP4V-ES/90000
a=fmtp:96
profile-level-id=3;config=000001B003000001B50C2020202F000001010000012000844003F18A021
E0A31
a=control:track1
a=cliprect:0,0,640,480
a=framerate:15.000000
m=audio 7878 RTP/AVP 0
a=rtpmap:0 PCMU/8000/1
a=control:track2
RTSP/1.0 200 OK
CSeq: 1
Date: Fri, Jan 05 2007 18:10:03 GMT
Transport:
RTP/AVP;unicast;destination=192.168.0.12;source=192.168.0.200;client_port=7990-7991;server_
port=6970-6971
Session: 2
```

5.5.3.2 RTSP OPTIONS

The OPTIONS command returns a list of supported RTSP commands.

Example:

```
OPTIONS * RTSP/1.0
CSeq: 1
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 1
Date: Fri, Jan 05 2007 18:32:15 GMT
Public: OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE
```

5.5.3.3 RTSP SETUP

The SETUP command configures the delivery method for the data. The SETUP command requires and generates the following additional header field:

Transport	Specifies how the data stream is transported. Supported variants: RTP/AVP;unicast;client_port=port1-port2 RTP/AVP;multicast;client_port=port1-port2 RTP/AVP/TCP;unicast

The response returns a session identifier that should be used with stream control commands to the server (PLAY, PAUSE, TEARDOWN). If the Session header includes a timeout parameter, then the session needs to be kept alive. This can be done by sending RTSP requests to the server containing the session identifier (e.g. OPTIONS) within the specified timeout time or through the use of RTCP. The RTSP server does not support reconfiguring of the transport parameters.

Example:

```
SETUP rtsp://192.168.0.200/mpeg4/track1 RTSP/1.0
CSeq: 2
Transport: RTP/AVP;unicast;client_port=7990-7991
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 2
```

```
Date: Fri, Jan 05 2007 18:00:04 GMT
Transport:
RTP/AVP;unicast;destination=192.168.0.12;source=192.168.0.200;client_port=7990-7991;server_
port=6970-6971
Session: 1
```

Example:

```
SETUP rtsp://192.168.0.200/mpeg4/track2 RTSP/1.0
CSeq: 3
Transport: RTP/AVP;unicast;client_port=7995-7996
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 3
Date: Fri, Jan 05 2007 18:00:04 GMT
Transport:
RTP/AVP;unicast;destination=192.168.0.12;source=192.168.0.200;client_port=7995-7996;server_
port=6972-6973
Session: 1
```

5.5.3.4 RTSP PLAY

The PLAY command starts (or restarts if paused) the data delivery to the client. The PLAY command generates the following additional header fields:

Range	Specifies the range of time being played. Since only live streams are used, the specified time will always begin now and have no stop time.
RTP-Info	Information about the RTP stream. More specifically, it includes the next RTP sequence number that will be used.

Example:

```
PLAY rtsp://192.168.0.200/mpeg4 RTSP/1.0
CSeq: 4
Range: npt=0.000-
```

```
Session:  1
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 4
Date: Fri, Jan 05 2007 18:00:04 GMT
Range: npt=0.000-
Session: 1
RTP-Info:
url=rtsp://192.168.0.200/mpeg4/track1;seq=58119;rtptime=1339768530,url=rtsp://192.168.0.20
0/mpeg4/track2;seq=53323;rtptime=2812669845
```

5.5.3.5 RTSP PAUSE

The PAUSE command pauses the data delivery from the server.

Example:

```
PAUSE rtsp://192.168.0.200/mpeg4 RTSP/1.0
CSeq: 5
Session: 1
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 5
Date: Fri, Jan 05 2007 19:03:59 GMT
Session: 1
```

5.5.3.6 RTSP TEARDOWN

The TEARDOWN command terminates the data delivery from the server.

Example:

```
TEARDOWN rtsp://192.168.0.250/mpeg4 RTSP/1.0
CSeq: 6
Session: 1
```

Response example:

```
RTSP/1.0 200 OK
CSeq: 6
Session: 1
```