



Technology Specification

CGI Specification

3S Product

Version	Date	Changes
1.5	2011/3/4	records
1.4	2010/12/14	Modify for doCmd
1.3	2010/7/29	Modify for I/O
1.2	2010/6/8	Modify for document format
1.1	2010/6/4	Modify for records.cgi
1.0	2010/5/27	Created

PocketNet Technology Inc



Table of Contents

1. Camera Operation	3
2. List Parameters.....	9
3. User Setting	13
4. System log	15
5. Date / Time	16
7. Backup.....	19
8. Restore.....	20
9. Hard factory default	22
10. Factory Default	22
11. Motion Detection.....	23
12. I/O.....	24
13. IP Filter	26
14. Audio	30
15. Overlay	32
16. Firmware Upgrade.....	34
17. Test Service Status.....	35
18. Capture still images	37
19. Records	39

1. Camera Operation

1. PTZ control

To control the Pan, Tilt and Zoom behavior of a PTZ unit, the following PTZ control URL is used. Installing a PTZ driver is done in three main steps:

1. Upload the driver
2. Associate the driver with a camera address (id).
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 camera accomplishes step 3.

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 3S 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](http://tools.ietf.org/html/rfc2396): 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

Syntax:

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

Parameter:





<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.
type=<string>	normal, digital	Default value is normal type If the value is not specified. It issues a digital PT action if type is digital.
move=<string>	home, up, down, left, right, upleft, upright, downleft, downright	Moves the device 5 degrees or 200ms in the specified direction.
pan=<float>	-180.0 - 180.0	Absolute: Pans the device relative to the (0,0) position.
tilt	-180.0 - 180.0	Absolute: Tilts the device relative to the (0,0) position.
gotoserverpresetname=<string>	<preset name> ⁴	Move to the position associated with the <preset name>.
gotoserverpresetno=<int>	1, ...	Move to the position associated with the specified preset position number. That is defined as sharp number of PresetPos.P#
speed=<int>	1 - 100	Sets the head speed of the device that is connected to the specified camera.
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.
rzoom=<int>	-9999 - 9999	Absolute: Zooms the device n steps relative to the current position.



		Positive values mean zoom in, negative values mean zoom out. Relative: Zooms the device approx. n steps relative to the current position. Positive values mean zoom in, negative values mean zoom out.
rfocus=<int>	-9999 - 9999	Absolute: Move Focus n steps relative to the current position. Positive values mean focus 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.
iris=<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.
Autofocus=<string>	on, off	Autofocus On/Off.
autoiris=<string>	on, off	Autoiris On/Off.
doCmd=<string>	<string>, read. Note: Max string length is 512 chars.	Write Value : by RS485 /ptz.cgi?doCmd=A00000023D3CAF0C (comport=0) /ptz.cgi?doCmd=a00000103736af06&comport=1 down for Pelco-P /ptz.cgi?doCmd=a00000083736af06&comport=1 up for Pelco-P /ptz.cgi?doCmd=a00000000000af0f&comport=1 stop for Pelco-P Read Value : by RS485 /ptz.cgi?doCmd=read&comport=0&readlen=8&timeout=10000
comport=<int>	0,1	The comport for doCmd, default =0.
timeout=<int>	200 ~ 2000 ms	The timeout for doCmd, default =200 ms.

readlen=<int>	1 ~ 512	The readlen for doCmd., example: readlen=8.
---------------	---------	---

⁰ <preset name> is a string with a maximum of 31 characters, ~ is not allowed.

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

² Actual values are depended on device driver-specific.

Example: Request information about which PTZ speed for camera 1.

```
http://myserver/ptz.cgi?camera=1&query=speed
```

2. PTZ configuration

Set and configure PTZ preset positions..

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

Method: GET

Syntax:

```
http://<servername>/ptzconfig.cgi?<parameter>=<value>[&<parameter>=<value>...]
```

Parameters:

<parameter>=<value>	Values	Description
Camera	1,...	Camera number
serverpresetname=<string>	<preset name> ¹	Associates the current position to <preset name> as a preset position in the server.
devicepresetno=<int>	0, ...	Saves the current position as a preset position number of device in the server.

3. PTZ driver update

To handle installation and remove PTZ drivers.



Method: GET/POST

Syntax:

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

Parameters:

<parameter>=<value>	Values	Description
replacedriver=<int>	1, ... Index of the driver.	Uninstall currently installed driver. If parameter replaced driver is provided, the given driver will be uninstalled.

4 PTZ administration

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

Method: GET

Syntax:

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

Parameters:

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



address=<int>	0,,,	Address, specified in PTZ protocol, of the camera is possible to control multiple PTZ devices through a serial port.
---------------	------	--

Example: Associate driver "Pelco-P" with camera 2 and control address is 2.

<http://myserver/ptzadmin.cgi?camera=2&ptzdrivername=Pelco-P&name=cam2&address=2>



2. List Parameters

These requests have different security levels. The security level for each parameter is specified in the [parameters document](#).

Method: GET

Syntax:

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

Parameters:

<i><parameter>=<value></i>	Values	Description
<i>action=<string></i>	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. The “add” and “remove” of actions are supported for Event group only.

1 List parameters

Note: Access right depend on parameter access right.

Syntax:

```
http://<servername>/param.cgi?action=list[&<parameter>=<value>...]
```

Parameters:

<i><parameter>=<value></i>	Values	Description
<i>group=<string>[,<string>...]</i>	<i><group[.name]>[,<group[.name]>...]</i>	Returns the values of the camera parameter which named <i><group>.<name></i> . If <i><name></i> is omitted, all



		the parameters of the <i><group></i> 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.
--	--	--

Example: List the Network parameters.

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

Example: List the all parameters of events.

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

1.1 List output format

```
HTTP/1.1 200 ok\r\n
Date: Mon, 19 Jan 2009 06:35:59 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
<parameter>=<value>\n
[ <parameter pair> ]
\r\n
```

Example: Network query response.

```
HTTP/1.1 200 ok\r\n
Date: Mon, 19 Jan 2010 06:35:59 GMT\r\n
Cache-control: no-cache\r\n
```





```
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
Network.BootProto="none"\n
Network.DefaultRouter="192.168.2.254"\n
Network.DNSServer1="192.168.1.252"\n
Network.DNSServer2="192.168.1.9"\n
Network.IPAddress="192.168.2.123"\n
Network.SubnetMask="255.255.255.0"\n
Network.MACAddress="A8:55:6A:00:00:00"\n
\r\n
```

If the CGI request an invalid parameter name, the server returns nothing

Return:

```
HTTP/1.1 200 ok\r\n
Date: Mon, 19 Jan 2010 06:38:54 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
```

2. Update parameter

Syntax:

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

Parameters:

<parameter>=<value>	Values	Description
<string>=<string>	<group.name>=<value>	Assigns <value> to the parameter <group.name>. The <value> must be URL-encoded if it contains non-alphanumeric characters. The camera parameters must be entered exactly as



		named in the camera or the video server.
--	--	--

Example: Set the maximum number of viewers to 5. Server returns changed pairs in reply.

http://myserver/param.cgi?action=update&Image.MaxViewers=5

Return:

```
HTTP/1.1 200 ok\r\n
Date: Mon, 19 Jan 2010 06:38:54 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
Nvramset="Image.MaxViewers=5"\r\n
\r\n
```

Example: Server returns nothing if changing the value of parameter and it has been already.

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

Return:

```
HTTP/1.1 200 ok\r\n
Date: Mon, 19 Jan 2010 06:38:54 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
\r\n
```

3. User Setting

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

Note: This request requires root authorization.

Method: GET

Syntax:

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

Parameters:

<parameter>=<value>	Values	Description
action=<string>	add, update, remove or get	add action that to create a new user account. update action that to change account information of specified parameters if the account exists. remove action that to remove an existing account. get action that to fetch a list of the user information.
origuser=<string>	<string>	specified the original user name to update if you want to change the old user name
user=<string>	<string>	The user account name that you want to modify. The valid characters are a thru z, A thru Z and 0 thru 9. root is reserved as super user account.
pwd=<string>	<string>	The unencrypted password of the account. The valid characters are a thru z, A thru Z, 0 thru 9, and _-!/:=*+%','?^;[]
grp=<string>	<string>	Group names of the account. The valid groups are administrator, operator and viewer.
ptz=<string>	yes, no	Privilege to control ptz



Example: Create a new administrator account.

```
http://myserver//pwdgrp.cgi?action=add&user=joe&password=foo &grp=administrator&ptz=yes
```

Example: Change the password of an existing account.

```
http://myserver/pwdgrp.cgi?action=update&user=joe&password=0118&grp=administrator&ptz=yes
```

Example: Remove an account.

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

Example: List groups and users.

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

Return:

```
HTTP/1.1 200 ok\r\n
Date: Mon, 19 Jan 2009 08:51:10 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
administrator="root,joe"\n
operator=""\n
viewer=""\n
ptz="root,joe"\n
\r\n
```

4. System log

Get system log information.

Note: This requires administrator access (administrator authorization).

Note: The response is product/release-dependent.

Method: GET

Syntax:

```
http://<servername>/systemlog.cgi
```

Parameters:

No available parameter.

Example:

```
http://myserver/systemlog.cgi
```

Return:

```
HTTP/1.1 200 ok\r\n
Date: Mon, 19 Jan 2009 09:01:14 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
2009 Jan 19 12:26:24 : System boot up.\n
2009 Jan 19 12:26:29 : HTTPD listening on port 80\n
\r\n
```

5. Date / Time

Get or set the system date and time.

Method: GET

Syntax:

```
http://<servername>/date.cgi?<parameter>=<value>[&<parameter>=<value>...]
```

Parameters:

<i><parameter>=<value></i>	Values	Description
action= <i><string></i>	get or set	get action that to get the current date and time. set action that to set the current date and/or time.

1. Get system date and time

Syntax:

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

Return:

```
HTTP/1.1 200 ok\r\n
Date: Mon, 19 Jan 2009 09:07:34 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
<month> <day>, <year> <hour>:<minute>:<second>\r\n
```

Example:



```
HTTP/1.1 200 ok\r\n
Date: Mon, 19 Jan 2009 09:07:34 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
Jan 19, 2009 17:07:34\r\n
```

2. Set system date and time

Note: This requires administrator authorization.

Syntax:

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

Parameters:

<parameter>=<value>	Values	Description
year=<int>	1970 – 2037	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.

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/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
```

6. Reboot

Restart server.

Note: This requires administrator authorization.

Method: GET

Syntax:

```
http://<servername>/restart.cgi
```

7. Backup

Backup the all of parameters.

Note: This requires administrator authorization.

Method: GET

Syntax:

```
http://<servername>/backup.cgi
```

Return:

```
HTTP/1.1 200 ok\r\n
Accept-Ranges: bytes\r\n
Connection: close\r\n
Content-Type: application/octet-stream\r\n
Content-Disposition: attachment; filename=param.bin\r\n
[parameter pairs]
```

8. Restore

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

Note: This requires administrator authorization. If the parameters are restored successfully, the system will automatically reboot.

Method: POST

Syntax:

```
http://<servername>/restore.cgi
```

The file content is provided in the HTTP body according to the format given in [RFC 1867](http://tools.ietf.org/html/rfc1867). The body is created automatically by the browser if using HTML form with input type "file".

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

```
POST /restore.cgi HTTP/1.1\r\n
Accept: */*\r\n
Referer: http://myserver/restore_param.asp\r\n
Accept-Language: zh-tw\r\n
Content-Type: multipart/form-data; boundary=-----7d91481a1b0a00\r\n
UA-CPU: x86\r\n
Accept-Encoding: gzip, deflate\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; GTB5; .NET CLR 2.0.50727; .NET CLR
3.0.04506.648; .NET CLR 3.5.21022; InfoPath.1) \r\n
Host: 192.168.2.218\r\n
Content-Length: 8532\r\n
Connection: Keep-Alive\r\n
Cache-Control: no-cache\r\n
Authorization: Basic cm9vdDpyb290\r\n
\r\n
```



```
-----7d91481a1b0a00\r\n
Content-Disposition: form-data; name="file"; filename="C:\Documents and Settings\Administrator\param.bin"\r\n
Content-Type: text/plain\r\n
```

[Content]

```
-----7d91481a1b0a00\r\n
Content-Disposition: form-data; name="submit"\r\n
\r\n
Submit\r\n
-----7d91481a1b0a00--\r\n
```

9. Hard factory default

Reload all of parameters to their factory default value except PTZ driver. If the parameters are reloaded successfully, the system will automatically reboot.

Note: This request requires administrator authorization.

Method: GET

Syntax:

```
http://<servername>/hardfactorydefault.cgi
```

10. Factory Default

Reload factory default. All parameters except network and PTZ driver are set to their factory default values. If the parameters are reloaded successfully, the system will automatically reboot.

Note: This requires administrator authorization.

Method: GET

Syntax:

```
http://<servername>/factorydefault.cgi
```

11. 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<groupnumber>.<parameter name>. The group number is a unique number which is stated when a new dynamic parameter group is created, i.e. Motion.M0.

1. Update the Motion Detection parameters

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

```
http://myserver/param.cgi?action=update&Motion.M1.Top=1500  
&Motion.M1.Bottom=8000
```

2. List the Motion Detection parameters

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

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

Example: List all Motion Detection windows.

```
http://myserver/param.cgi?action=list&group=Motion.*
```

12. I/O

This request is inquired the alarm input and updated alarm output status.

Note: This requires administrator authorization.

Method: GET

Syntax:

```
http://<servername>/portctrl.cgi?<parameter>=<value>[&<parameter>=<value>...]
```

Parameters:

<i><parameter>=<value></i>	Values	Description
action= <i><string></i>	list, update	list = get action that to get the current alarm input status. update = update action that to set the alarm output status

Example: Fetch the status of alarm input 1.

```
http://myserver/portctrl.cgi&action=list&group=in1
```

Return: The example above returns the following reply:

```
HTTP/1.1 200 ok\r\n
Date: Mon, 19 Jan 2009 09:49:02 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
in1="0"\r\n
\r\n
```



www.3Svision.com.tw

Example: Fetch the status of alarm output 1.

```
http://myserver/portctrl.cgi?action=list&group=out1
```

Return: The example above returns the following reply:

```
HTTP/1.1 200 ok\r\n
Date: Thu, 29 Jul 2010 06:21:23 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
\r\n
out1="1"\r\n
\r\n
```

Example: Set the status of alarm output 1.

```
http://myserver/portctrl.cgi?action=update&out1=1
```

Return: The example above returns the following reply:

```
HTTP/1.1 200 ok\r\n
Date: Thu, 29 Jul 2010 06:19:38 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: text/plain\r\n
```



13. IP Filter

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

1. IP address filter administration

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

Method: GET

Syntax:

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

Parameters:

<parameter>=<value>	Values	Description
action=<string>	add, remove, removeall 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 device via a proxy server, the



		proxy server's IP address must be added to the list of allowed addresses.
verify=<string>	yes, no	Verify that you are able to access the 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 3S 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/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/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/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/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/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/ipfilter.cgi?action=removeall
```

1.1 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\nAccept addresses: <IP addresses>\r\n
```

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

```
HTTP/1.0 200 OK\r\nContent-Type: text/plain\r\n\r\nError="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/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\nContent-Type: text/plain\r\n\r\nAccept addresses: 10.13.10.12\r\nEnabled: no\r\nVerification failed: IP address "10.13.17.245" is not an accepted address
```

14. Audio

1 Audio data transmit

Transmit the audio data stream. The each audio frame is packaged in a RTP packet and with 4 bytes header. The '\$' is a special character to be start of packet. The whole format is following as.

\$ 1 byte	RTP/RTCP 1 byte	Length of data 2 bytes
Audio frame encoded by G.711 or G.726 and with RTP header ...		

Method: POST

Syntax:

```
http://<servername>/transmit.cgi
```

Example: Upload the audio data

```
http://myserver/transmit.cgi
```

Response: The x-codec is specified the codec of audio data.

```
POST /transmit.cgi HTTP/1.0
```

```
User-Agent: 3S Agent
```

```
Content-Type: application/x-audio-tunneled
```

```
x-sessioncookie: D9hKgAAfB8ABCftAAAAAw
```

```
x-Codec: G726
```

```
Pragma: no-cache
```

```
Cache-Control: no-cache
```

```
Content-Length: 32767
```



www.3Svision.com.tw

Expires: Sun, 9 Jan 1972 00:00:00 GMT

Authorization: basic YWRtaW46YWRtaW4=

{continuously RTP tunneled packets}



15. Overlay

1 Upload image

Upload the overlay icon or symbol image.

Method: POST

Syntax:

```
http://<servername>/call_overlay_upload.cgi
```

Supported file format is BMP. The maximum size of the image is 1.5MB.

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

```
POST /call_overlay_upload.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="symbol"; channel="1"

--AaBo3x\r\n
--AaBo3x\r\n
Content-Disposition: form-data; name="fileInput"; filename="symbol.bmp"\r\n
Content-Type: image/bmp\r\n
\r\n
<file content of symbol.bmp>
\r\n
--AaBo3x--\r\n
```

2 Overlay Setting

Set the overlay parameters.



Method: GET

Syntax:

http://<servername>/overlay.cgi

Parameter:

<parameter>=<value>	Values	Description
channel=<int>	1, ... ¹	Applies only to video servers with more than one video input. Selects the source channel. If omitted, the default channel is used.
text=<String>	Text	The text overlay on the stream. The maximum length is 24 characters.
showtext=<String>	yes,no	Shows the text
showicon=<String>	yes,no	Shows the icon
showdate=<String>	yes,no	Shows the date
showtime=<String>	yes,no	Shows the time

Example:

http://<servername>/overlay.cgi?channel=1&showtext=yes&showicon=no

16. Firmware Upgrade

1 Firmware upgrade

Upgrade the firmware version.

Note: This requires administrator access (administrator authorization).

Method: POST

Syntax:

```
http://<servername>/firmwareupgrade.cgi
```

Example:

```
POST /firmwareupgrade.cgi 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
```

17. Test Service Status

1 Test the services

To test the status of service.

Note: This requires operator access.

Method: POST

Syntax:

```
http://<servername>/testservice.cgi[?<parameter>=<value>]
```

Parameter:

<parameter>=<value>	Values	Description
Name=<string>	Samba, SMTP	Specifies the name of service test.
Username=<string>	A string	Samba, SMTP services required
Password=<string>	A string	Samba, SMTP services required.
FolderPath=<string>	A string	Samba service required.
FromMail=<string>	A string	SMTP service required.
MailTo=<string>	A string	SMTP service required. Specified the address of receiver.
MailServer=<string>	A string	SMTP service required. Specified the host of server.
WeakestMethod=<string>	Plain, login, md5 or digest	SMTP service required. Specified the authentication method.

Example:

```
POST /testservice.cgi?name=samba&UserName=root&Pasword=root&FolderPath=/ HTTP/1.0\r\n Content-Length:
<content length>\r\n \r\n
```



www.3Svision.com.tw

If the service testing is done, the server returns an successful message.

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Msg=" service testing is done"
```



18. Capture still images

1 The snapshot function from the Client

To capture still images with the snapshot function from the Client.

Note: This requires operator access.

(2010/01/01 later to support new parameter: image.jpg, camera & type)

Method: GET

Syntax:

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

OR

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

Parameter:

<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.
type =<string>	single, motion	If type = single for single JPEG images. If type = motion for streaming MJPEG video.

Example: Get a current JPEG file of the live video.

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

Example: Get a MJPEG video of the live video.

```
http://<servername>/image.cgi?camera=1&type=motion
```



www.3Svision.com.tw

If the values is right, the server returns mjpeg stream.

Return:

```
HTTP/1.1 200 ok\r\n
Date: Mon, 04 Jan 2010 09:16:19 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: multipart/x-mixed-replace; boundary=--mjpegboundary\r\n

--mjpegboundary\r\n
Content-Type: image/jpeg\r\n
```



19. Records

1. The records function for storage file system (Hard disk/SD Card / samba)

To management records files of PLAYBACK, DOWNLAND, REMOVE, CHECKFILE and LIST function.

Method: GET

Syntax:

```
http://<servername>/ records.cgi[?<parameter>=<value>[&<parameter>=<value>... ]
```

Parameter:

<parameter>=<value>	Values	Description
action =<string>	play, download, remove, checkfile or list	Applies only to storage function for PLAYBACK, DOWNLAND, REMOVE, CHECKFILE and LIST the records files.
storage =<string>	single, motion	file search from which storage
filename =<string>		filename length should less than 255 characters
cam =<int>	0~99	indicate which camera file to be search (For action=list)
type =<string>	All / schedule / alarm / motion / videoloss / bd / netloss / pir	trigger type to be used to filter the related file (For action=list)
date =<string>	1970/01/01~xxxx/xx/xx	date to be used to filter file (For action=list)
page =<int>	1~999	current page number to be query (For action=list)
lines =<int>	0~100	lines per page to be listed (For action=list)



afterhour =<int>	0~23	hours interval after current query search time (For action=list)
beforehour =<int>	0~23	hours interval before current query search time (For action=list)
month =<string>	1970/01/ ~ xxxx/xx	Month to be used to filter file (For action=list)
filetype =<string>	All / avi / jpg	file format to be used to fileter the related file (For action=list)

Example: Playback the video file.

```
http://<servername>/records.cgi?action=play&storage=???&filename=???
```

Example: Download the video file.

```
http://<servername>/records.cgi?action=download&storage=???&filename=???
```

Example: Remove the video file.

```
http://<servername>/records.cgi?action=remove&storage=???&filename=???
```

Example: Check the video file is valid.

```
http://<servername>/records.cgi?action= checkfile&storage=???&filename=???
```

Example: List the video file.

```
http://<servername>/records.cgi?action=list&storage=???&cam=???&type=???&date=???&page=???  
&lines=50&afterhour=???&beforehour=???
```



www.3Svision.com.tw

If the values is right, the server returns mjpeg stream.

Return:

```
HTTP/1.1 200 ok\r\n
Date: Mon, 04 Jan 2010 09:16:19 GMT\r\n
Cache-control: no-cache\r\n
Connection: close\r\n
Content-Type: multipart/x-mixed-replace; boundary=--mjpegboundary\r\n

--mjpegboundary\r\n
Content-Type: image/jpeg\r\n
```

