Brickcom

Megapixel Day & Night

Fixed Box Network Camera

FB-100A Series

User's Manual

Quality Service Group

Product name:	Network Camera (WFB-100A/FB-100A)
Release Date:	2009/
Manual Revision:	V2.0
Web site:	www.brickcom.com
Email:	technical@brickcom.com
	info@brickcom.com
Made in Taiwan.	©2009 Brickcom Corporation. All Rights Reserved

Table of Contents

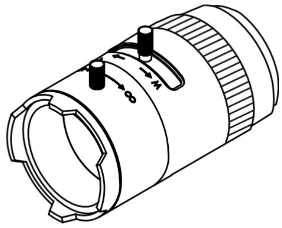
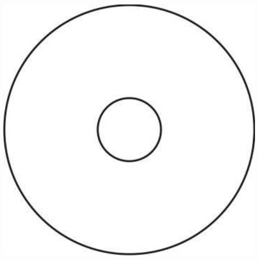
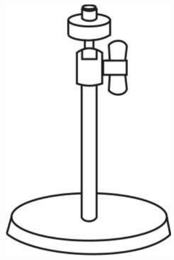
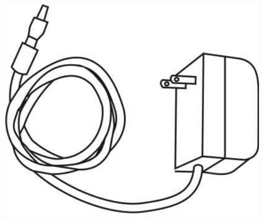
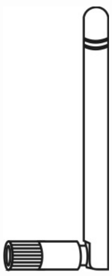
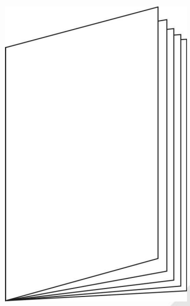
Brickcom

Before You Use This Product

The use of surveillance devices may be prohibited by law in your country. The Network Camera is not only a high-performance web-ready camera but also can be part of a flexible surveillance system. It is the user's responsibility to ensure that the operation of such devices is legal before installing this unit for its intended use.

It is important to first verify that all contents received are complete according to the list in the "Package Contents" chapter. Take notice of the warnings in “Quick installation guide” before the Network Camera is installed, then carefully read and follow the instructions in the “Installation” chapter to avoid damages due to faulty assembly and installation.

Package Contents

a. FB-100A 	b. CS mount Lens (Optional) 
c. Product CD 	d. Camera Stand 
e. Warranty Card 	f. Power Adapter 
g. Detachable Antenna (WFB-100A) 	h. Quick Guide 

Fixed Box Network Camera Overview

The Brickcom WFB-100A/FB-100A IP camera offers reliable and excellent video quality solution for 24-hour surveillance application. Users can view live, motion image from anywhere by web browser or mobile phone via Internet or 3G network respectively. With the mega pixel progressive sensor and built-in removable IR-cut Filter, it delivers the extremely clear and detailed images that CCTV cameras cannot provide.

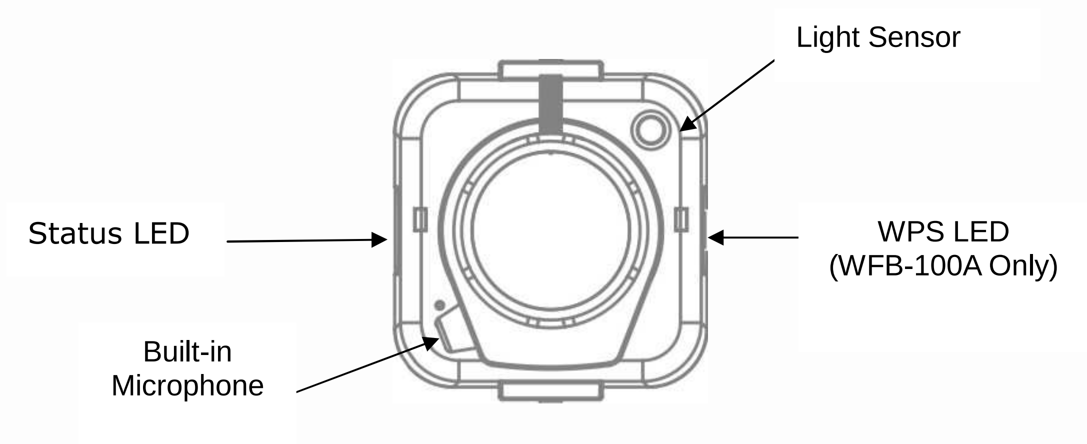
Other than motion detection function, the WFB-100A/FB-100A can also support intelligence surveillance such as object tracking, people counting, semantic region alarm and so on. In addition, the Brickcom IP camera can transmit the video to portable devices via other technology, for instance, WiMAX, 3G cell phone, NAS, Digital Frame and power line.

For easy setup, “Easy Installation Wizard” makes the configuration simple even for users without IT background. The Brickcom IP camera simplifies the hardware and software installation by flexible design and multiple applications. In other words, the WFB-100A is not only for normal home security but also suitable for professional surveillance demand such as bank, office building, and factory applications.

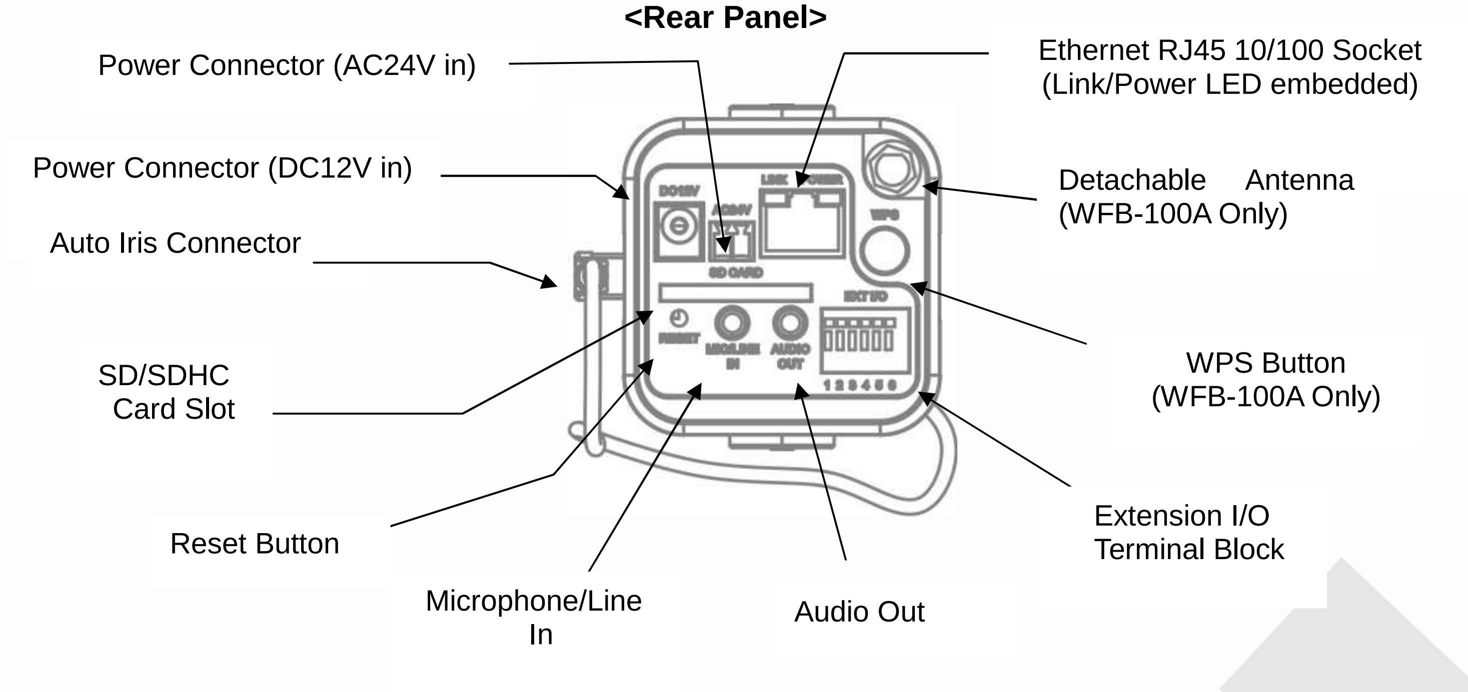

WiMAX CPE
BS
Phone
Video
Network Camera
External Microphone
AP/Router
Internet
3GPP
Power Line
Speaker
NAS
NB/PC with Web Browser
Power Line
Digital Frame

Device Appearance Description

<Front Panel>

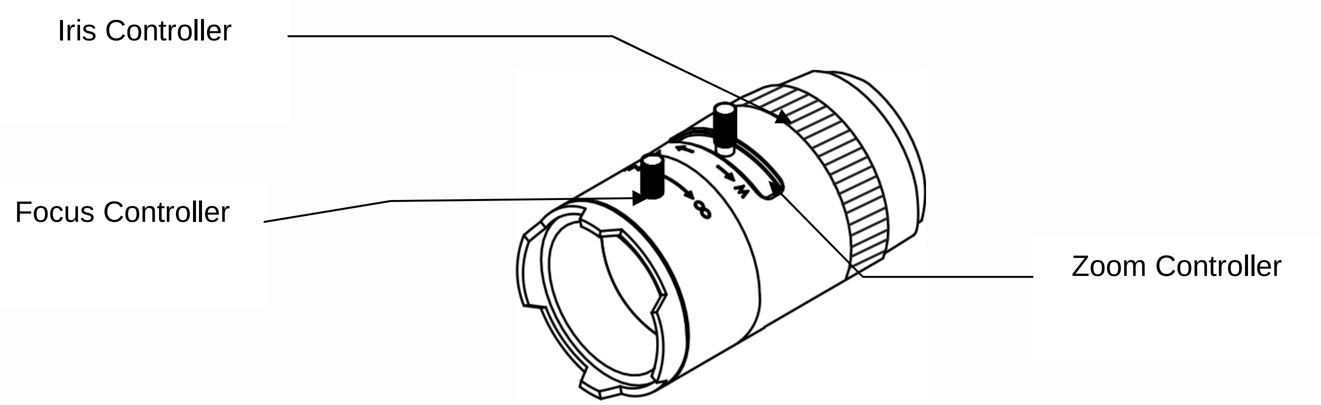


<Rear Panel>

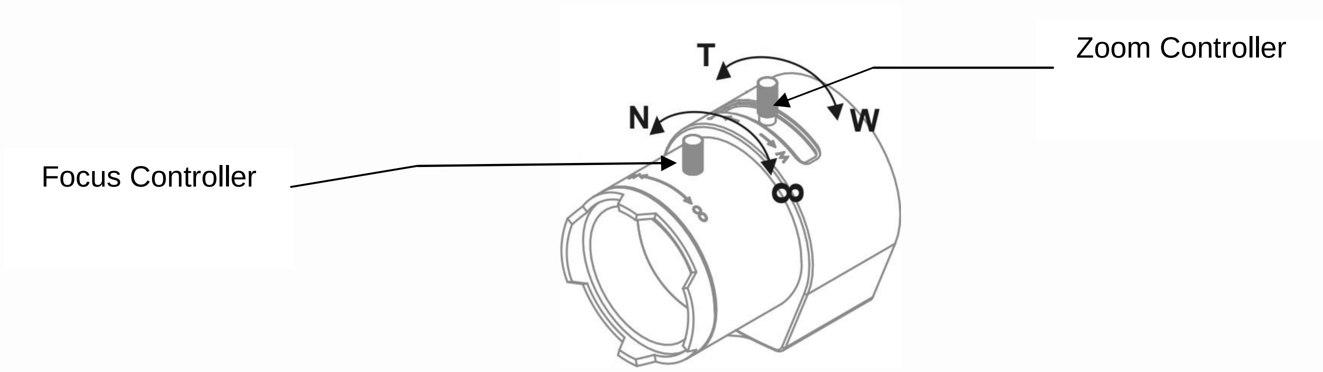


<CS Mount Lens>

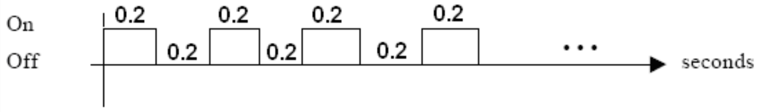
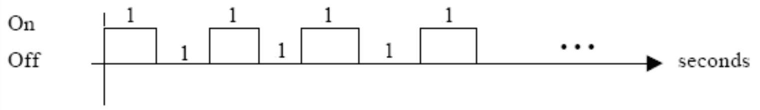
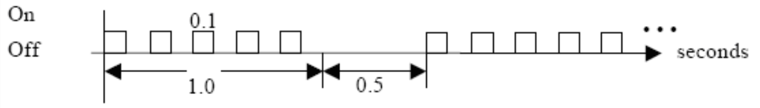
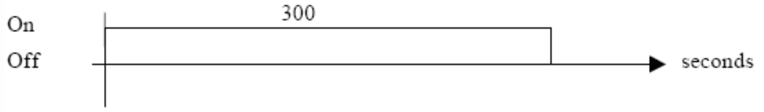
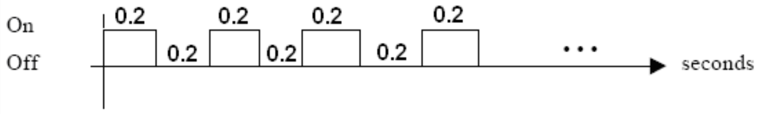
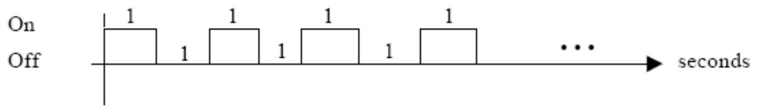
<Optional Lens>
<Vari-focal Lens with Manual Iris>



<Optional Lens>
<Vari-focal Lens with Auto Iris (DC Drive)>

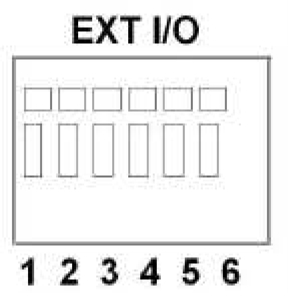


LED Behavior

Function	LED Behavior	Description	Remark
WPS		WPS in progress	WFB-100A Front Right (Blue)
WPS		WPS Error	WFB-100A Front Right (Blue)
WPS		Session overlap detected	WFB-100A Front Right (Blue)
WPS		WPS Success	WFB-100A Front Right (Blue)
Status		Hardware failure	Front Left (Green)
Status	**Steady On**	1. Restoring settings 2. Normal Operation	Front Left (Green)
Status	**Unlighted**	1. Power Off 2. Power On till System setup	The LED can be configured to be unlighted during normal operation (Green)
Status		While F/W upgrading	Front Left (Green)
Link	**Blinking**	Blinking while network connection in progress	Rear Left (Orange)
Link	**Unlighted**	No connection	Rear Left (Orange)
Power	**Steady On**	Normal Operation	Rear Right (Green)
Power	**Unlighted**	Power off	Rear Right (Green)

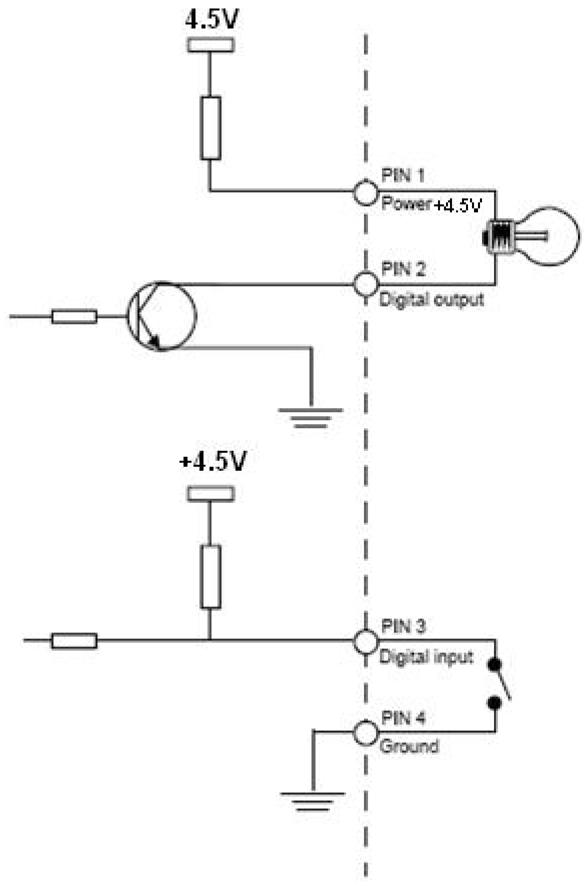
Extension I/O Terminal Block

The Network Camera provides an extension I/O terminal block which is used to connect external input/output devices. The pin definitions are listed as below.

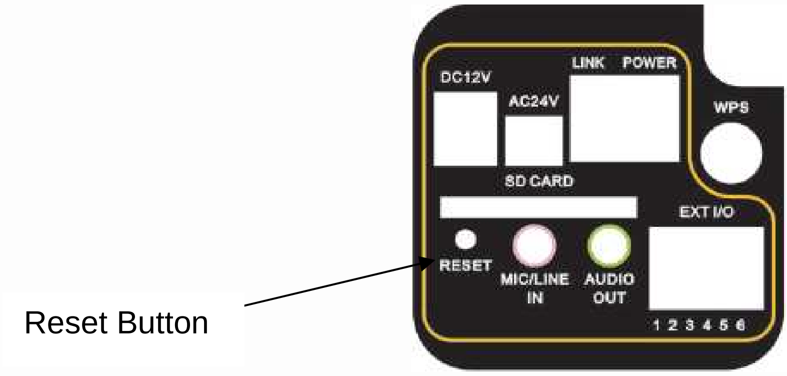


Pin	Function
1	Power +4.5V
2	Digital Output
3	Digital Input
4	Ground
5	RS-485 -
6	RS-485 +

DI/DO Diagram



Hardware Reset



The reset button is used to reset the system or restore the factory default settings. Sometimes resetting the system can return the camera to normal operation. If the problems remain after reset, please restore the factory settings and install it again.

Reboot: Please press and release the indented reset button within 1 second with paper clip or thin object. Wait for the network camera to reboot.

Restore: Please press and hold the reset button until the status of LED turns off. It takes about 10 seconds. Please note that all settings will be restored to factory default. Upon successful restore, the status of LED will be blue again during normal operation.

SD Card Capacity

The network camera is compliant with SD/SDHC (Maximum 32GB) cards.

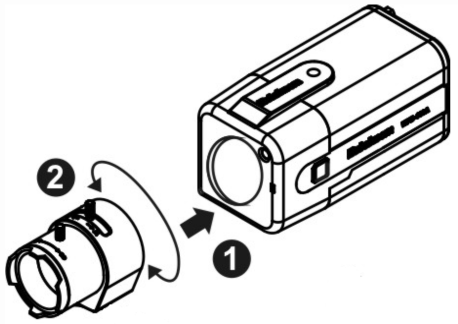
Installation

Hardware Installation

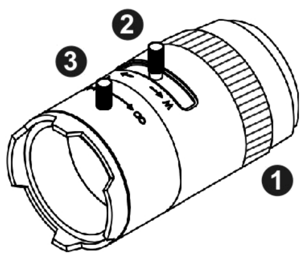
Mounting the CS-Mount Lens to the Camera

<Vari-focal Lens with Manual Iris> --- Optional Lens

1. Mount the CS-mount lens by turning it clockwise onto the camera mount until it stops.
2. If it's necessary, please turn the lens counterclockwise slowly until it gets the best position.

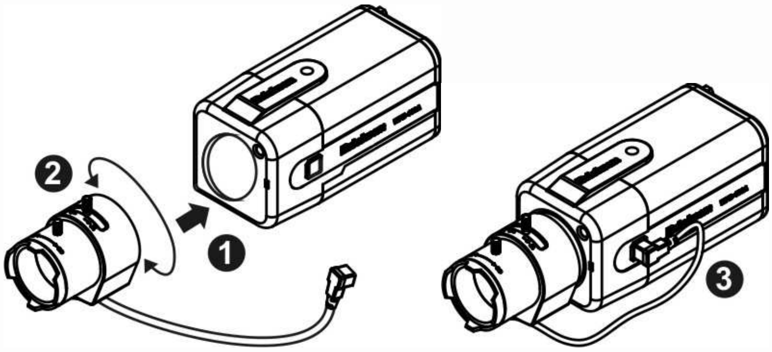


1. Turn the iris ring controller counterclockwise or clockwise until it gets the best performance.
2. Unscrew the zoom controller to adjust the zoom factor. Upon completion, tighten the zoom controller.
3. Unscrew the focus controller to adjust the focus range. Upon completion, tighten the focus controller.

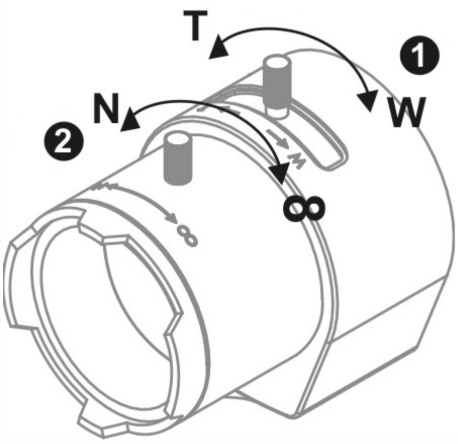


<Vari-focal Lens with Auto Iris> --- Optional Lens

1. Mount the CS-mount lens by turning it clockwise onto the camera mount until it stops.
2. If it's necessary, please turn the lens counterclockwise slowly until it gets the best position.
3. Connect the lens cable plug (DC Iris control cable) to the camera side connector.



1. Unscrew the zoom controller to adjust the zoom factor. Upon completion, tighten the zoom controller.
2. Unscrew the focus controller to adjust the focus range. Upon completion, tighten the focus controller.



For further information of vari-focal lens with auto iris, please refer to the supplied lens' instruction manual.

System Requirements

Operating System:

Microsoft Windows XP Home Edition SP2

Microsoft Windows XP Professional SP2

Computer:

IBM PC/AT Compatible

CPU:

Pentium 3GHz or faster

Memory:

1024 MB or more

Monitor:

1024 x 768 pixels or more, 24-bit True color or better

Network Interface:

10/100Mbps Network interface card must be installed

Web Browser:

Microsoft Internet Explorer 6.0 SP2

CD-ROM Drive:

It is necessary to read the operating instructions in the provided CD-ROM.

Adobe Reader:

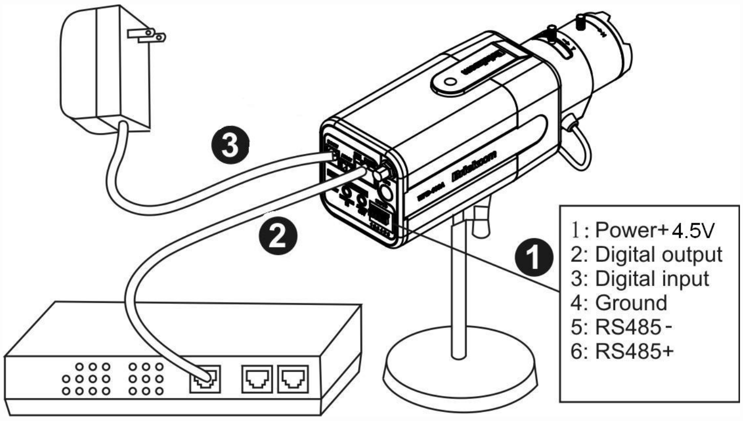
It is necessary to read the operating instructions in the provided CD-ROM.

- Audio function will not be working if a sound card is uninstalled on PC. Audio may be interrupted depending on the network environment.

Camera Connection

Basic Connection (Without PoE)

1. If you have external devices such as sensors and alarms, please make connections with extension I/O terminal block.
2. Connect the camera to a switch via Ethernet cable.
3. Connect the supplied power cable from the camera to the power outlet.



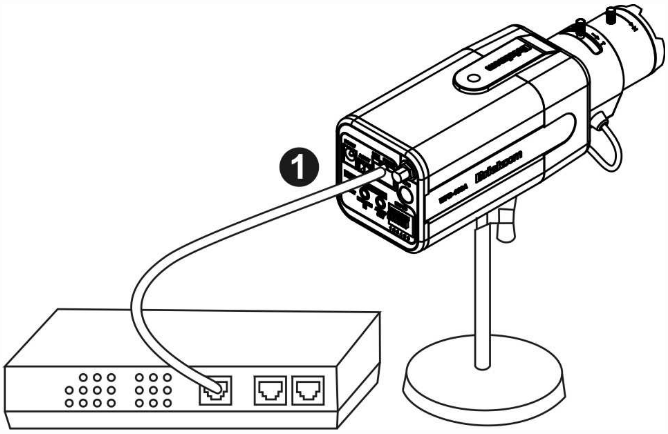
Please check your product package contains all the accessories listed in the foregoing Package Contents. Depending on the user's application, an Ethernet cable may be needed. The Ethernet cable should meet the specs of UTP Category 5 and not exceed 100 meters in length.

Upon powering up, the power LED will become lighted first and then the device will go through booting process. The link LED will be steady amber for getting IP address. After getting IP Address, the link LED will blink orange while network connection is processing.

Power over Ethernet (PoE) Connection

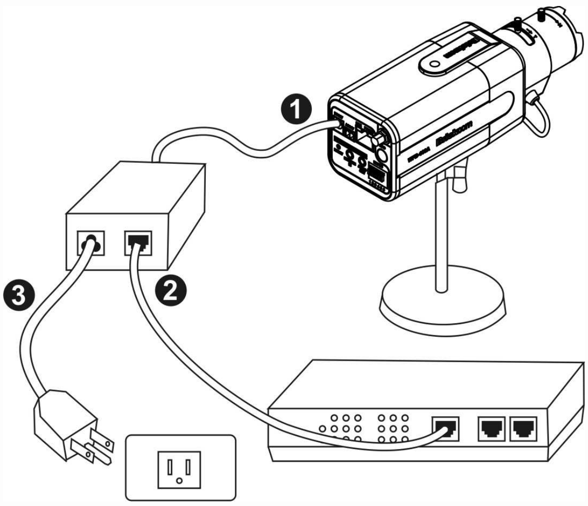
1. When connecting to PoE-enabled switch

   The camera is PoE compliant and please connects the camera to a PoE-enabled switch via single Ethernet cable.



2. When connecting to a non-PoE switch

   Please connect the camera to a non-PoE switch via PoE Injector (optional).



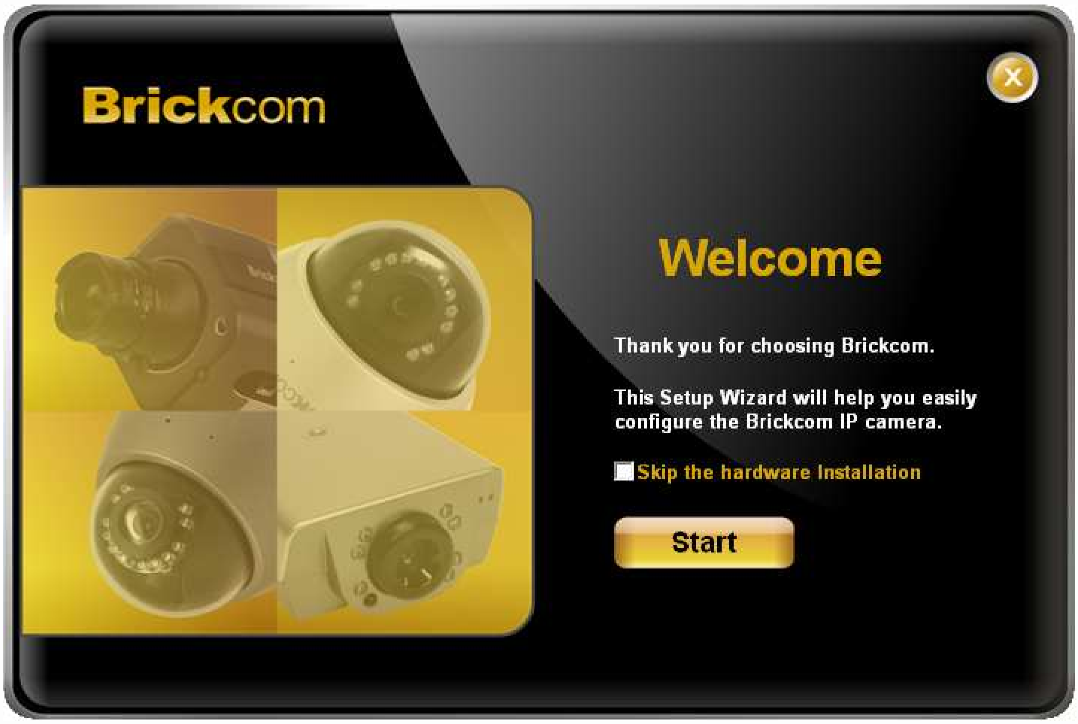
Software Installation

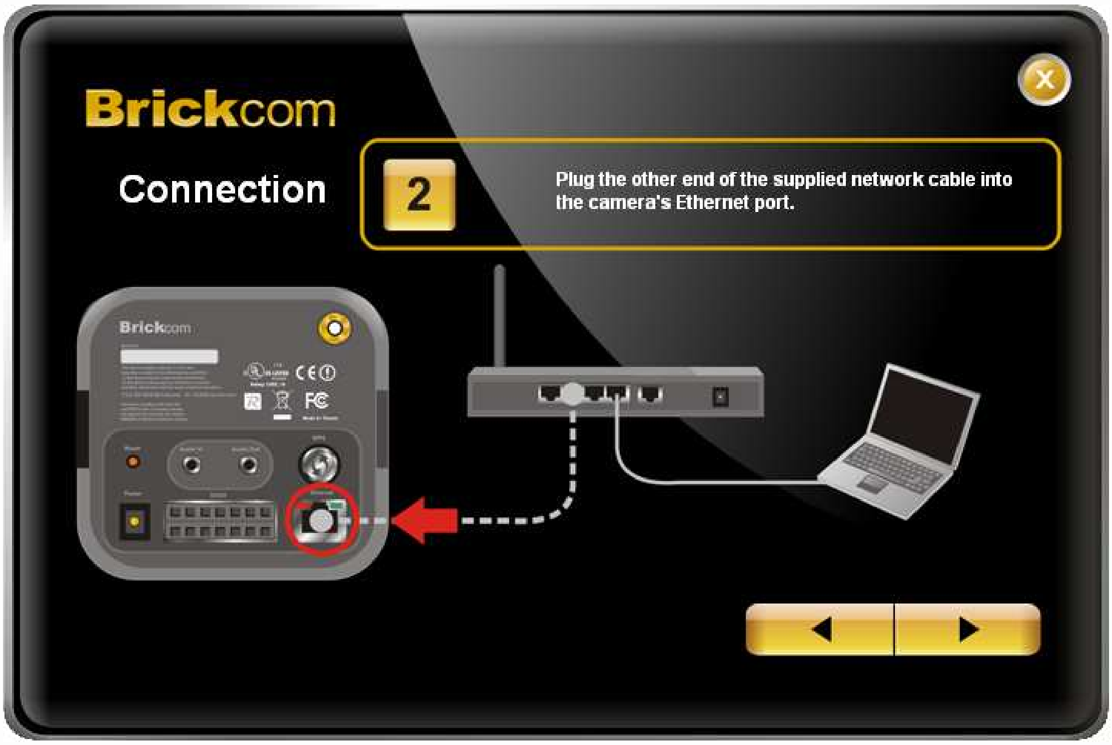
In this manual, "User" refers to whoever has access to the Network Camera, and "Administrator" refers to the person who can configure the Network Camera and grant user access to the camera.

After hardware connection checking, the users can run the Installation Wizard program included in the product CDROM to automatically search for the Network Camera in the Intranet. There may be many Network Cameras in the local network. Users can differentiate the Network Cameras with the serial number. The serial number is printed on the labels on the carton and the bottom of the Network Camera body.

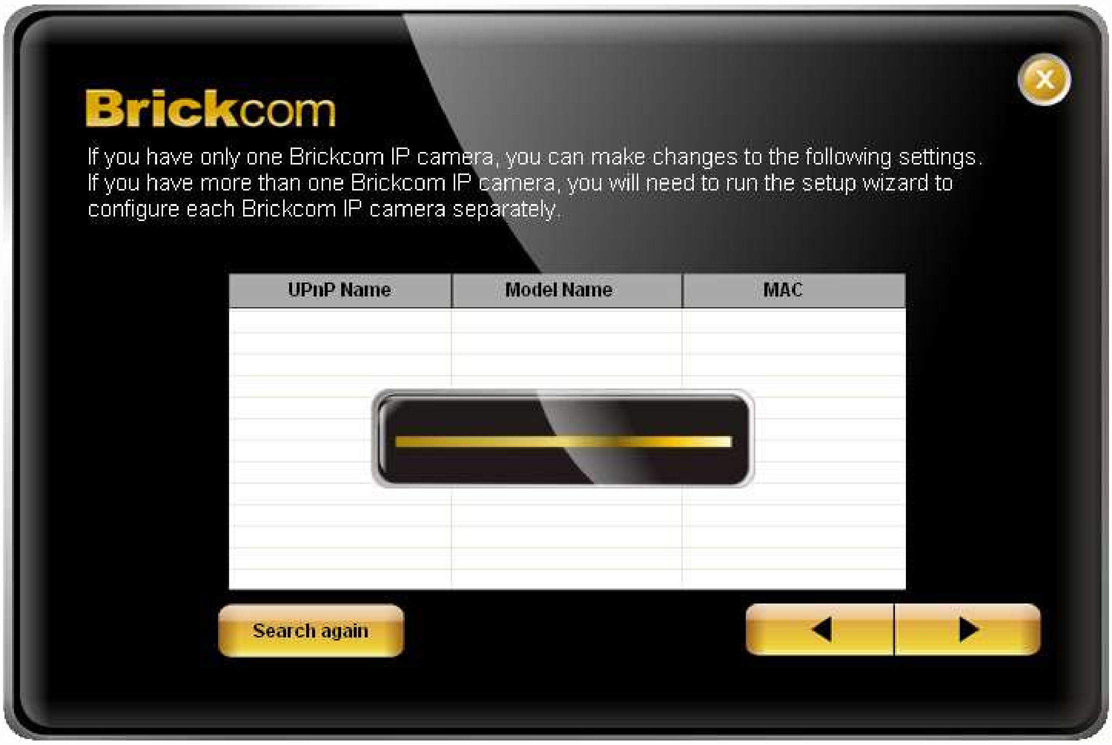
1. Insert the Installation CD into the CD-ROM driver. Click install and shows the welcome screen. Follow the steps to install the Installation wizard on user's computer.

2. Do not check the box if user would like to check the hardware installation settings, Otherwise click “Skip the hardware installation” to skip the hardware connection checking, the program will automatically search for the Network Camera in the Intranet.
Click “Start” to continue.




Brickcom
Connection
2
Plug the other end of the supplied network cable into the camera's Ethernet port.

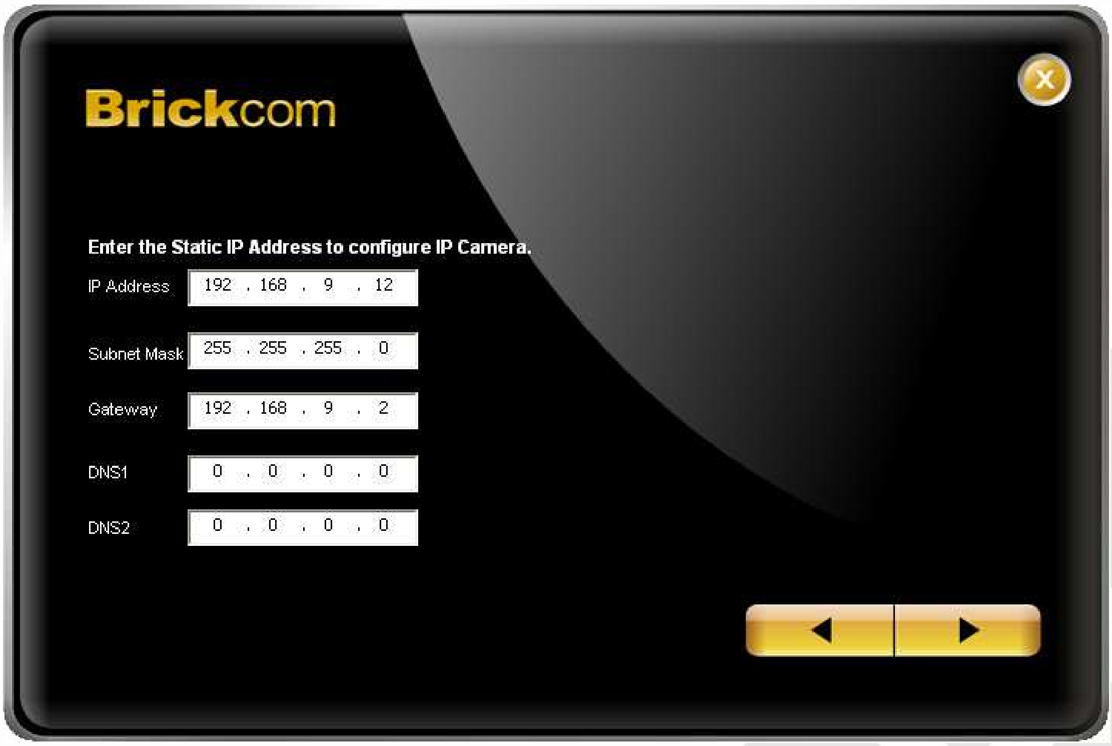

Brickcom
Connection
3
Plug the power adapter into the camera's power port. Connect the other end into the power outlet. Please check the LED light is on.



3. Select the Network Camera from the survey list and enter the user name and password. The user name and password are assigned as “**admin/admin**”.

4. Setting the Network Camera IP address
   User can either select simple mode or professional mode for network camera IP setting. If simple mode is selected, the easy configuration program will set up the connection automatically. If professional mode is selected, the user will need to configure the IP manually, The DHCP setting is recommended. If user wants to set IP address manually, please refer to the product user manual.


Brickcom
Select the IP address settings.
Automatically obtain an IP Address (DHCP)
Set IP Address configuration manually


Brickcom
Enter the Static IP Address to configure IP Camera.
IP Address 192 . 168 . 9 . 12
Subnet Mask 255 . 255 . 255 . 0
Gateway 192 . 168 . 9 . 2
DNS1 0 . 0 . 0 . 0
DNS2 0 . 0 . 0 . 0

5. After finish setting, the connection successful or fail showed. If connection failed, user can either try again or quit the installation. User can either select “Run PC-NVR” or "Start Web GUI" to continue or click “X” on the top right of the screen to finish the installation.

Once installation is completed, the Administrator should proceed to the next section "Access to the Network Camera" for necessary checks and configurations.

Access to the Network Camera

Check Network Settings

The Network Camera can be connected either before or immediately after software installation onto the Local Area Network. The Administrator should complete the network settings on the configuration page, including the correct subnet mask and IP address of gateway and DNS. Ask your network administrator or Internet service provider for the detail information.

Add Password to prevent Unauthorized Access

The Administrator should immediately implement a new password as a matter of prudent security practice. The user name and password for the Administrator are assigned as “**admin/admin**”. Once the Administrator’s password is saved, the Network Camera will ask for the user’s name and password before each access. The Administrator can set up a maximum of ten (10) user accounts. Each user can access the Network Camera except to perform system configuration. Once the password is changed, the browser will display an authentication window to ask for the new password. **Once the password is set, there is no provision to recover the Administrator’s password. The only option is to restore to the original factory default settings.**

Authentication

After opening the Web browser and typing in the URL of the Network Camera, a dialogue window pops up to request a username and password. The user name and password for the Administrator are assigned as “**admin/admin**”. Upon successful authentication, the following figure is displayed.

The foreground is the login window and the background shows the message if authentication fails. The user may check the option box to save the password for future convenience. This option is not available to the Administrator for obvious reason.

Installing plug-in

For the initial access to the Network Camera in Windows, the web browser may prompt for permission to install a new plug-in for the Network Camera on the Internet Explorer. Permission request depends on the Internet security settings of the user's PC or notebook. If the highest security level is set, the computer may prohibit any installation and execution attempt. This plug-in has been registered for certificate and is used to display the video in the browser. Users may click on  to proceed. If the web browser does not allow the user to continue to install, check the Internet security option and lower the security levels or contact your IT or networking supervisor for help.

Live View

Live View is the default page that opens when accessing the Network Camera. Live video is displayed directly in the browser window.

- **Stream1/Stream2 Channels**

The network camera offers simultaneous dual stream for optimized quality and bandwidth. To configure the codec compression and video resolution, please go to the Configuration->Camera/video/audio->Video to make the changes, or refer to the Video configuration on page 30.

- **TCP/UDP protocol**

TCP - This protocol guarantees the complete delivery of streaming data and thus provides better video quality. Nevertheless, the downside with this protocol is that its real-time effect is not as good as that of the UDP protocol.

UDP - This protocol allows for more real-time audio and video streams. However, network packets may be lost due to network burst traffic and images may be broken. Activate UDP connection when occasions require time-sensitive responses and the video quality is less important.

/ Recording on/off: shows the status of recording video

/ MIC on /off: shows the status of MIC volume.

/ MD on/off: shows the status of Motion Detection

- **Camera Control Panel -** There are two slider bars and eight control buttons on the remote controller. They are describe as below:
- **Brightness and Mic volume adjustment -** Drag the slider bar to adjust the image brightness level and Mic volume. Click "Default" for default brightness setting and "Mute" for no sound. For more Audio setting, please refer to the Audio configuration on page 33.
- **Speaker –**The build-in speaker plays the sound of an audio clip from computer MIC when it is enabled.

Play or Stop - Click this button to play or stop the video.

Recording - Click this button to record video to your computer.

Snapshot - Click this button to capture and save still images.

Digital Zoom - Click this button to enable the zoom operation.

Mirror - horizontally reflect the display of the live video.

Flip - vertically reflect the display of the live video.

Real Size - click this button to view the object in real size. Press this button again to switch back to normal mode.

Full Screen - Click this button to switch to full screen mode. Press "Esc" key to switch back to normal mode.

Motion Detection Alert: Click this button to enable motion detection alert function.

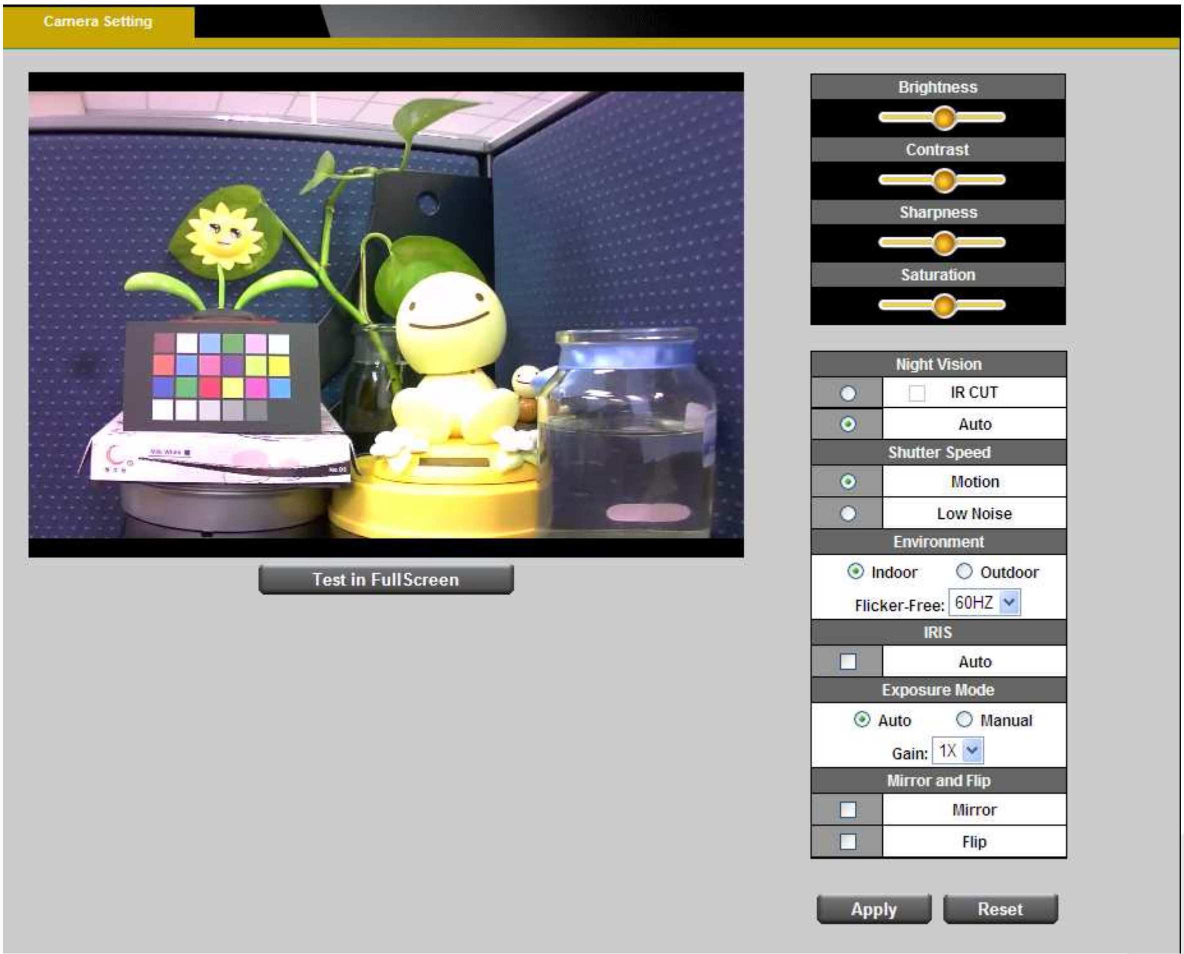
NOTE - The <Video Control Panel> function has no effect on the recorded video. Whatever changes made to the <Video Control Panel> **will not** be applied to the recorded video.

Configuration

Click **Configuration** on the main page to enter the camera setting pages. Note that only Administrators can access the configuration page.

Camera/Video/Audio

Camera



Camera Setting

Brightness - Drag the slider bar to adjust the image brightness level, which ranges from -5 to +5.

Contrast - Drag the slider bar to adjust the image contrast level, which ranges from -5 to +5.

Sharpness - Drag the slider bar to adjust the image sharpness level, which ranges from -5 to +5.

Saturation - Drag the slider bar to adjust the image saturation level, which ranges from -5 to +5.

Night Vision

IR CUT - the Network Camera switches off the IR cut filter at all times for the sensor to accept the infrared light, thus helps improve low light sensitivity.

Auto - The Network Camera automatically removes the filter by judging the level of ambient light.

Shutter Speed – Only effects in low lighting conditions

Motion – enable this function to have normal motion image, but slight blur.

Low Noise – enable this function to have sharper image, but slight tardy motion.

Environment - User can look for a place that best suits your needs, either outdoor or indoor.

Flicker-Free - While flicker-free technology eliminates the problem of flicker, it can cause slight judder on fast moving images or blurring problems; fast scrolling text for example may blur.

NOTE - The "Environment" setting adjusts the sampling rate of the camera sensor to achieve Flicker-free effect on the video.

The sampling rate, however, may not work best with the frame rate chosen in the "Video" setting.

For best recording experience, configure your IP camera to one of the following frame rates based on the Environment used:

Environment / Flicker-Free	Frame Rate
Outdoor	25, 10, 7, 5, 3, 2
Indoor (50/60 Hz)	20, 10, 7, 5, 3, 2

IRIS

Auto Iris lens - Select when the auto Iris lens is installed. Manual Iris lens is the default lens.

Exposure Mode

Auto - The camera sets the exposure by auto exposure time. Auto exposure setting including the long exposure mode is performed according to the lighting conditions.

Manual - The camera fixes the exposure time and performs auto exposure settings.

Gain - Set the Gain rate higher for a better video illumination. However, higher gain rate may cause bigger judder on fast moving images or blurring problems.

Mirror and Flip

Mirror - Enable to horizontally reflect the display of the live video.

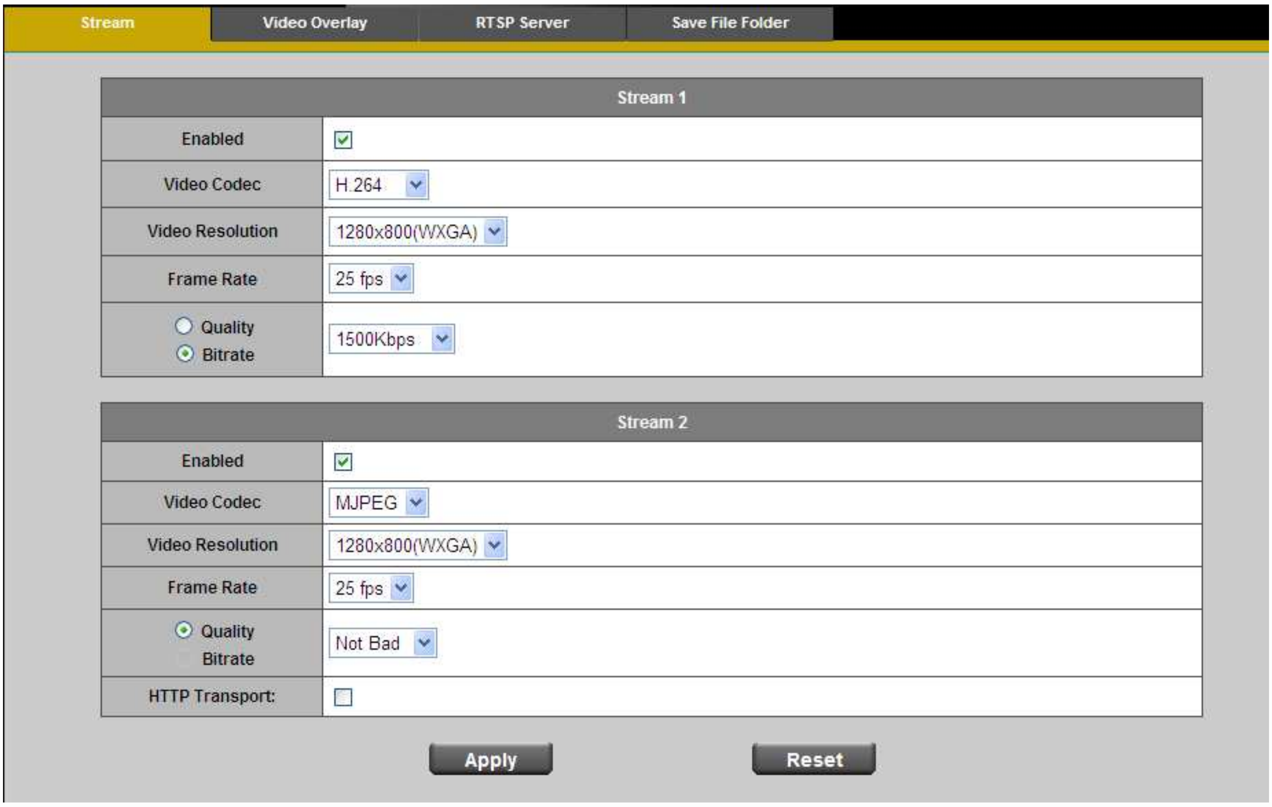
Flip - Enable to vertically reflect the display of the live video.

Click **Apply** or **Reset** to take effect.

.

Video

You can set up two separate streams for the Network Camera for different viewing devices.



Stream 1 & Stream 2

Video Codec - The Network Camera offers three choices of video codec standards for real-time viewing: H.264, MPEG-4 and MJPEG.

Video Resolution - Select from the drop down list to choose the best resolution that fit your need.

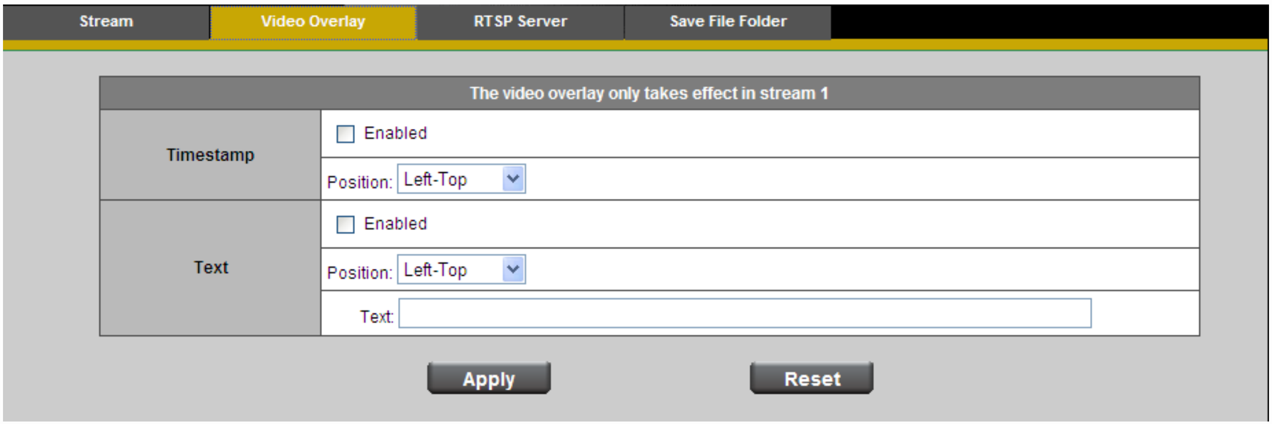
Frame Rate - Select from the drop down list of the frame rate, which ranges from 2 to 30 fps when H.264 or MJPEG is selected. Only 3 to 15 fps can be chosen when MPEG-4 is selected. Set the frame rate higher for a smoother video quality.

Video quality and bit rate - User can either choose “quality” or “bitrate” to control the video quality with video codec at H.264 or MPEG4. Only “quality” can be chosen when video codec at MJPEG is selected. Set the bitrate higher for a better video quality. However, high bitrate may cost high network bandwidth resources.

The video qualities are selectable at the following settings: SoSo, OK, Not Bad, Medium, Standard, and Good.

HTTP Transport – Enable to use HTTP protocol for video/audio communication. Click **Apply** or **Reset** to take effect.

Video Overlay

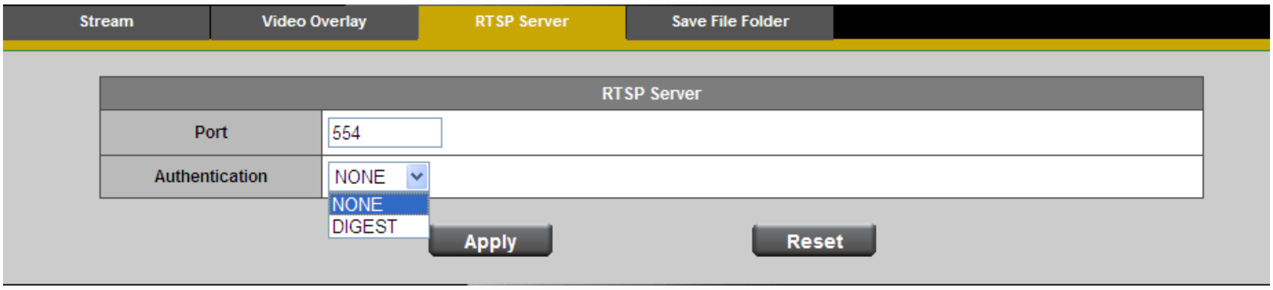


Video Overlay - Check to enable the timestamp function and select display position from the drop-down list if user wants date and time to be shown on the screen of the live video. User may also enable and enter the video description in text box; and select display position from the drop-down list if user wants to make a note about the network camera.

Click **Apply** or **Reset** to take effect.

NOTE - The video overlay only takes effect in stream 1.

RTSP Server



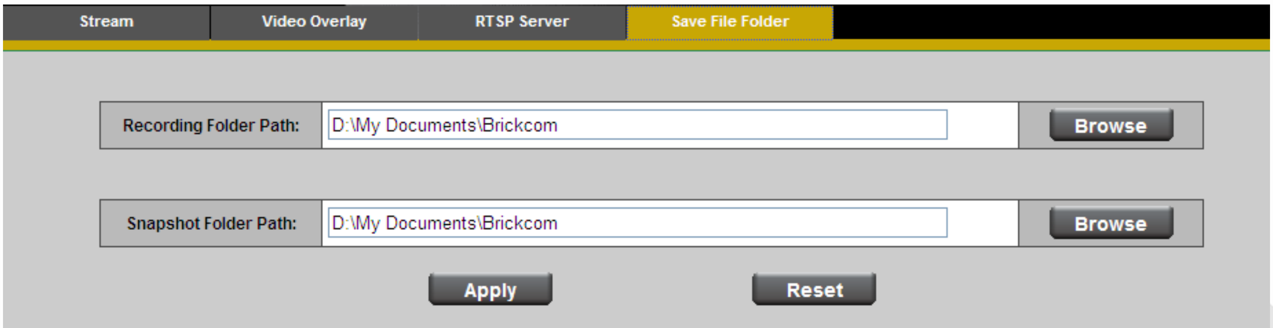
To utilize RTSP authentication, make sure that you have set a password for the Network Camera first.

RTSP (Real-Time Streaming Protocol) controls the delivery of streaming media. By default the port number is set to 554.

Authentication - Depending on your network security requirements, the Network Camera provides two types of security settings for streaming via RTSP protocol: NONE and DIGEST.

If DIGEST authentication is selected, user credentials are encrypted using MD5 algorithm, thus providing better protection against unauthorized access.

Save file folder



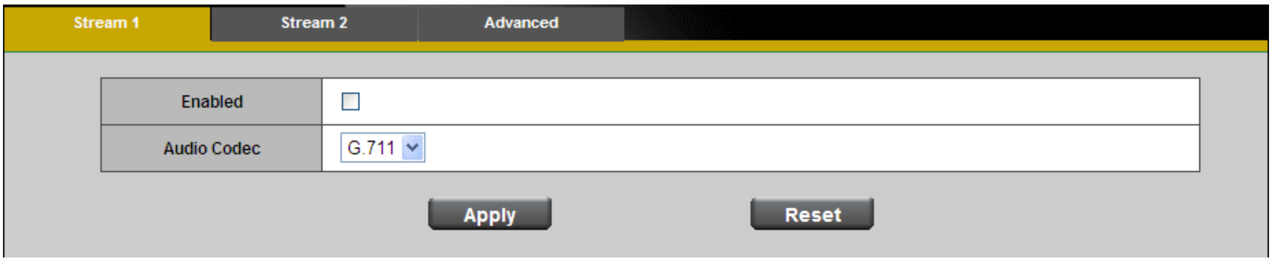
Recording folder path - The destination for saving the recording video files. Click browse to specify the saving path.

Snapshot folder path - The destination for saving the snapshot files. Click browse to specify the saving path.

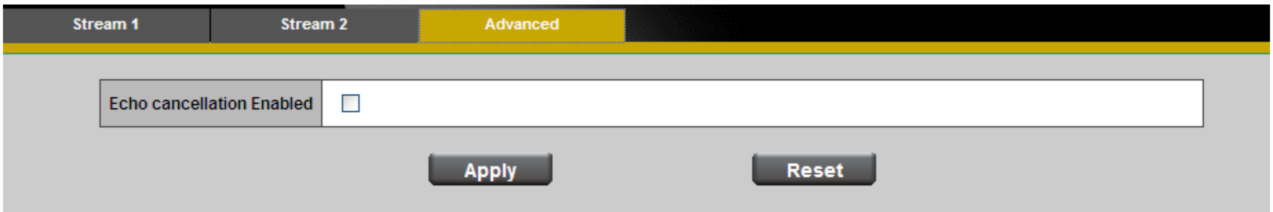
Click **Apply** or **Reset** to take effect.

Audio

You can set up two separate streams for the Network Camera for different viewing devices. User can either enable or disable the audio function. If audio enable is selected, select the Audio codec from the drop down list.



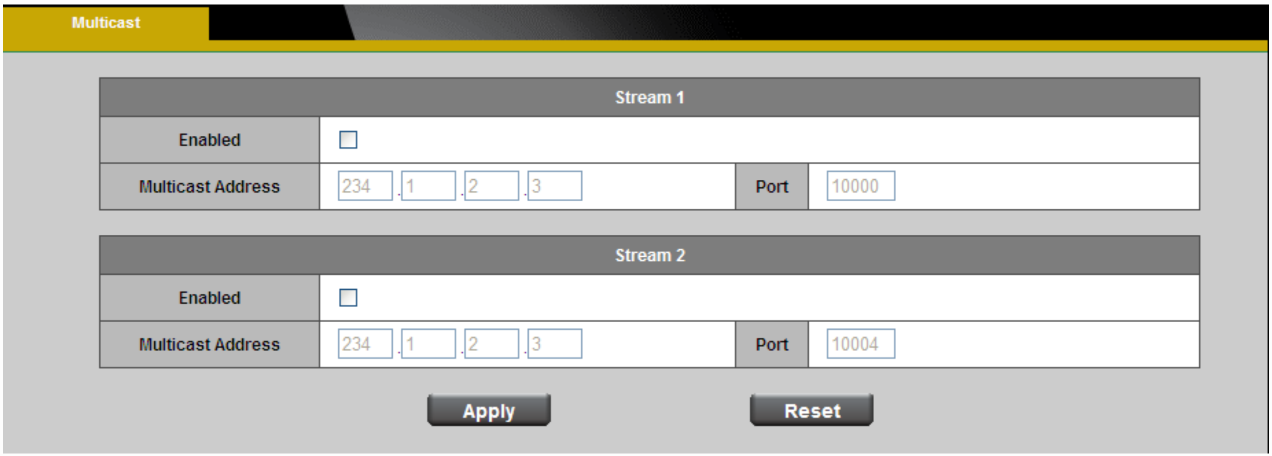
Advanced



Echo cancellation Enabled: Enable to avoid an echo.

Click **Apply** or **Reset** to take effect.

Multicast



Multicast sends a stream to the multicast group address and allows multiple clients to acquire the stream at the same time by requesting a copy from the multicast group address. Therefore, multicast can effectively save Internet bandwidth. The RTSP (Real-Time Streaming Protocol) controls the delivery of streaming media. Click to enable Multicast settings for stream 1 / Multicast settings for stream 2. The default value for multicast address and port are 234.1.2.3 and 10000. Use different port number for different stream. Use default value is recommended if you are not sure how to setting.

Note: Using the IP address of the camera enables you to view the video.

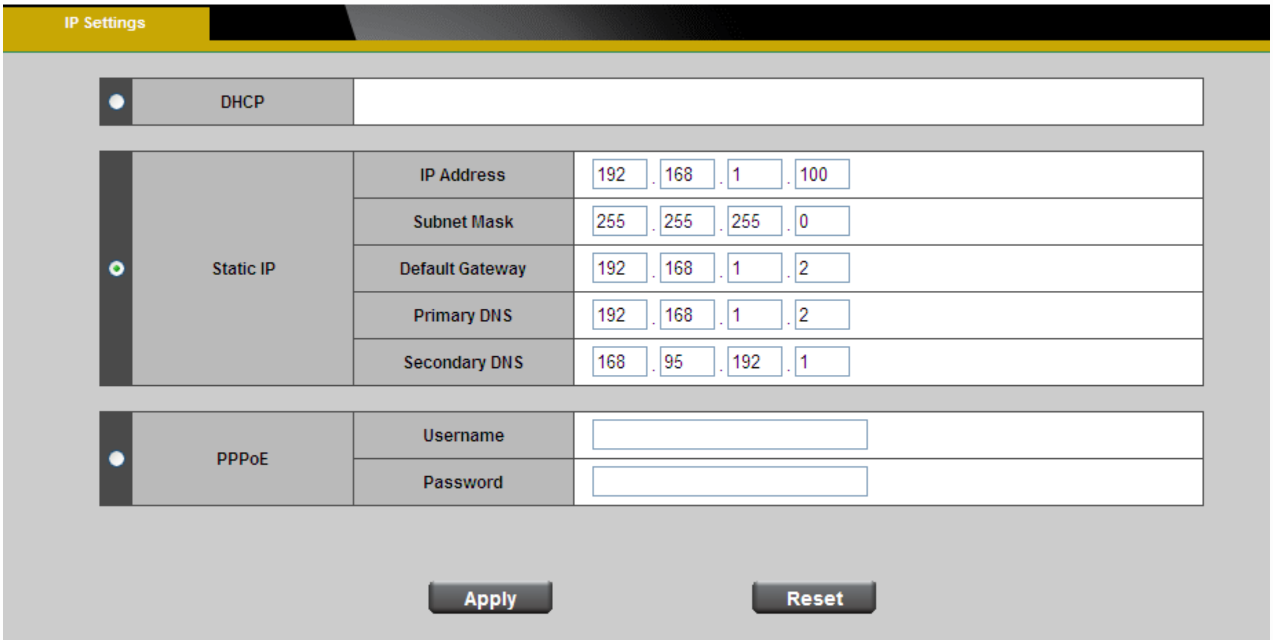
Example: rtsp://192.168.1.1/channel1

Click **Apply** or **Reset** to take effect.

Network

IP Setting

This section explains how to configure wired network connection for the Network Camera. There are several ways to setup the Network Camera over the Internet. The first way is to obtain an available dynamic IP address assigned by a DHCP server. The second way is to utilize a static IP. The third way is to use PPPoE.



DHCP - Get IP address automatically. Select this option to obtain an available dynamic IP address assigned by a DHCP server each time the camera is connected to the LAN.

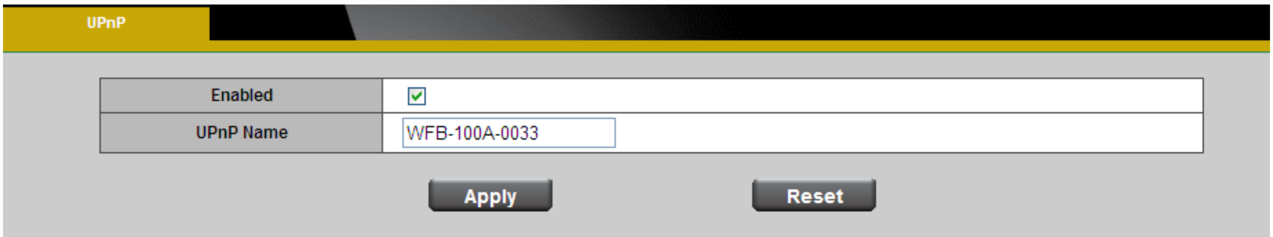
Static IP - Select this option to manually assign a static IP address to the Network Camera. Enter the static IP address, Subnet mask, Default Gateway, Primary and Secondary DNS provided by your ISP.

PPPoE - (Point-to-point over Ethernet)**:** Choose this connection type if you are connected to the Internet via a DSL Line. Note that to utilize this feature, it requires an account provided by your ISP. Enter the user name and password provided by your ISP.

Click **Apply** or **Reset** to take effect.

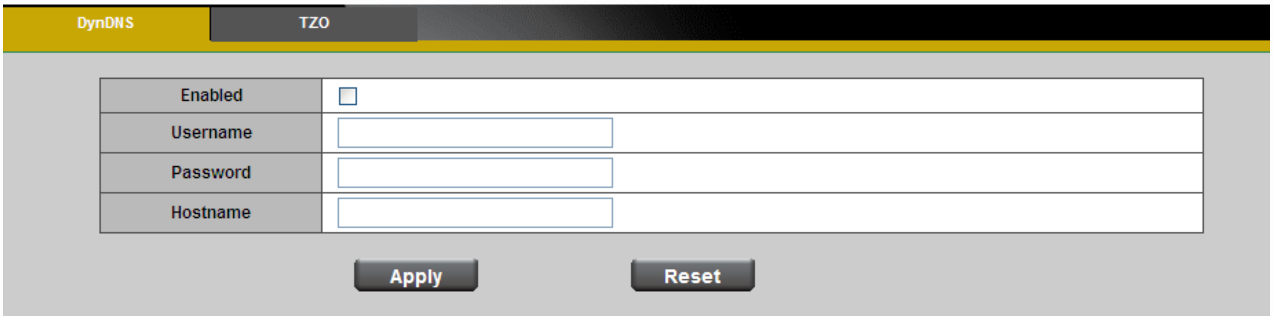
UPnP

Only UPnP discovery supported. Enable this function to allow the user to search for devices of interest on the network. Enter the UPnP name as you wish to show on the intranet.



Click **Apply** or **Reset** to take effect.

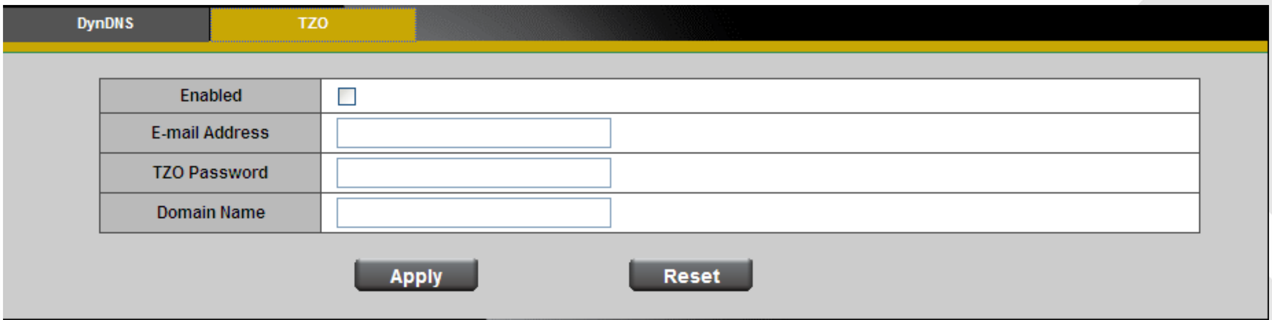
DDNS (dynamic domain name service)



DynDNS - Enable the DDNS service allows your Network Camera, especially when assigned with a dynamic IP address, to have a fixed host and domain name. Note that before utilizing this function; please apply a dynamic domain account first. Enter the username, password and hostname when enabled the DDNS.

Click **Apply** or **Reset** to take effect.

TZO

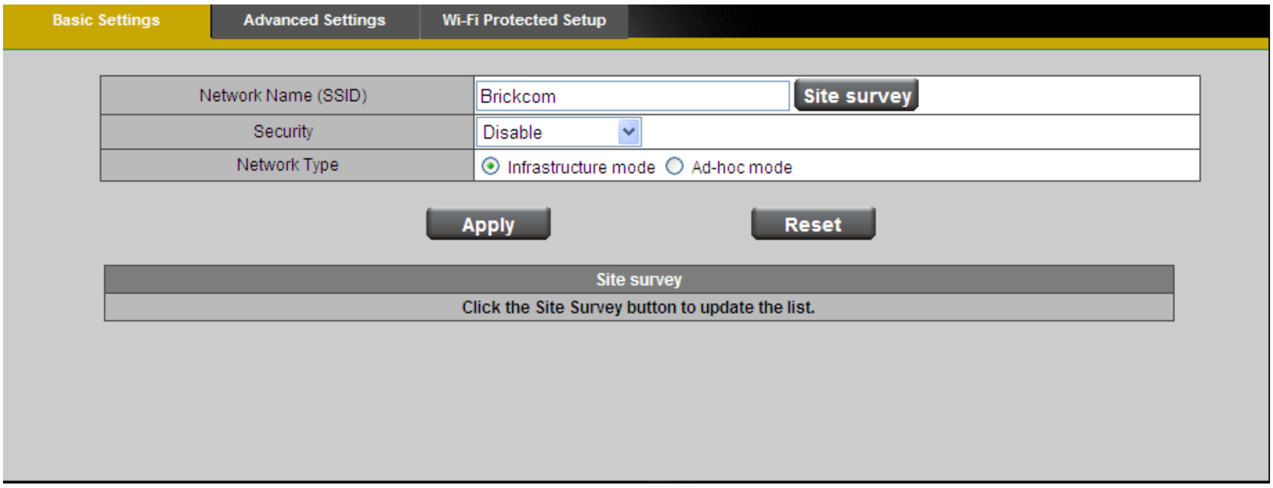


TZO - TZO is one kind of the DDNS providers. User can refer to the TZO.com: visit http://www.tzo.com/ to apply a dynamic domain account when selecting this DDNS provider. Enter the e-mail address, password and domain name when enabled the TZO.

Click **Apply** or **Reset** to take effect.

Wireless

Basic Settings



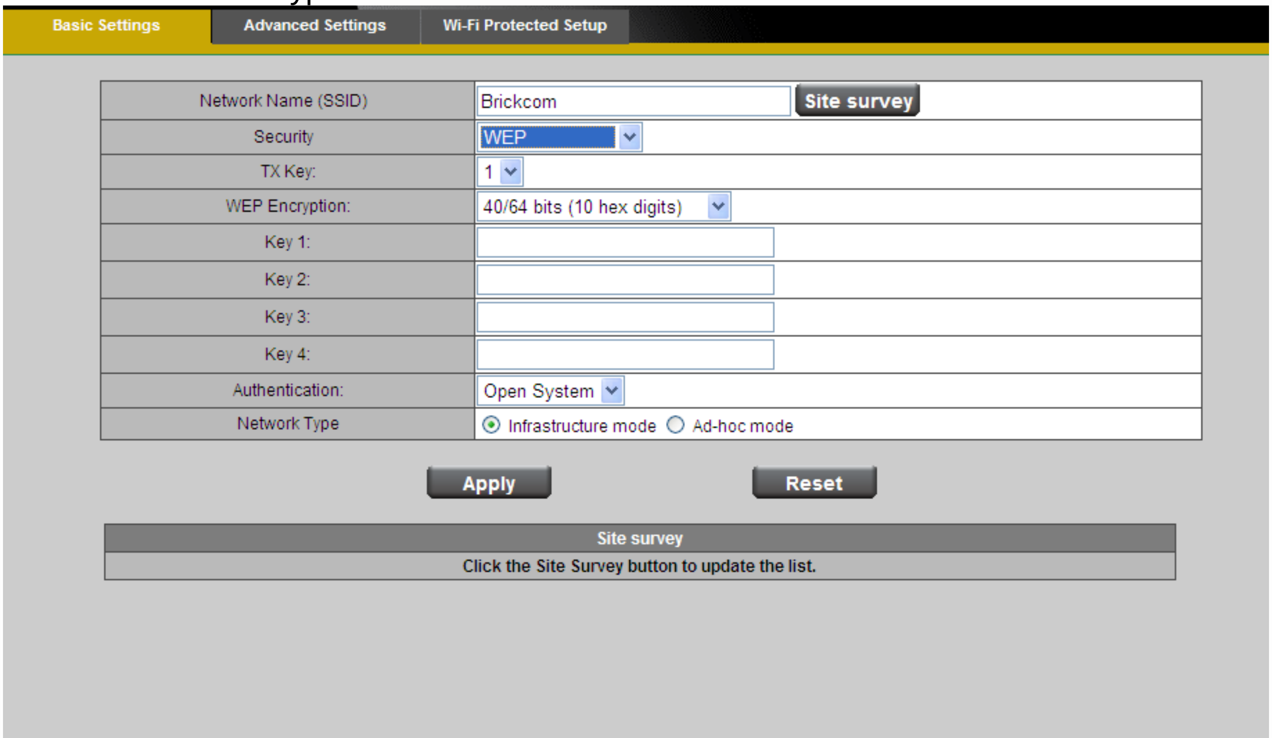
Network Name (SSID) - The SSID is the network name shared among all points in a wireless network. The SSID must be identical for all devices in the wireless network. It is case-sensitive and can be up to 32 characters in length. Make sure this setting is the same for all points in your wireless network.

Wireless devices have a default wireless network name or Service Set Identifier (SSID) set by the factory,. Brickcom wireless products use **Brickcom** as the default wireless network name. You should change the wireless network name to something unique to distinguish your wireless network from other wireless networks that may exist around you, but do not use personal information, because this information may be available for anyone to see when browsing for wireless networks.

Security - Encryption protects data transmitted over a wireless network. Wi-Fi Protected Access (WPA-Personal/WPA2-personal) and Wired Equivalent Privacy (WEP) offer different levels of security for wireless communication. A network encrypted with WPA-Personal/WPA2-personal is more secure than a network encrypted with WEP, because WPA-Personal/WPA2-personal uses dynamic key encryption. To protect the information as it passes over the airwaves, you should enable the highest level of encryption supported by your network equipment.

WEP

WEP is a basic encryption method that is not as secure as WPA.



Tx Key - Select a key from the drop-down menu.

WEP Encryption: Select a level of WEP encryption, 64 bits 10 hex digits or 128 bits 26 hex digits. The default is 64 bits 10 hex digits.

Key 1-4 - Enter the WEP key(s) manually

Authentication Type - The default is set to open system, which allows either Shared Key or Auto authentication to be used. With Open System authentication, the sender and the recipient do NOT use a WEP key for authentication. With Shared Key authentication, the sender and recipient use a WEP key for authentication.

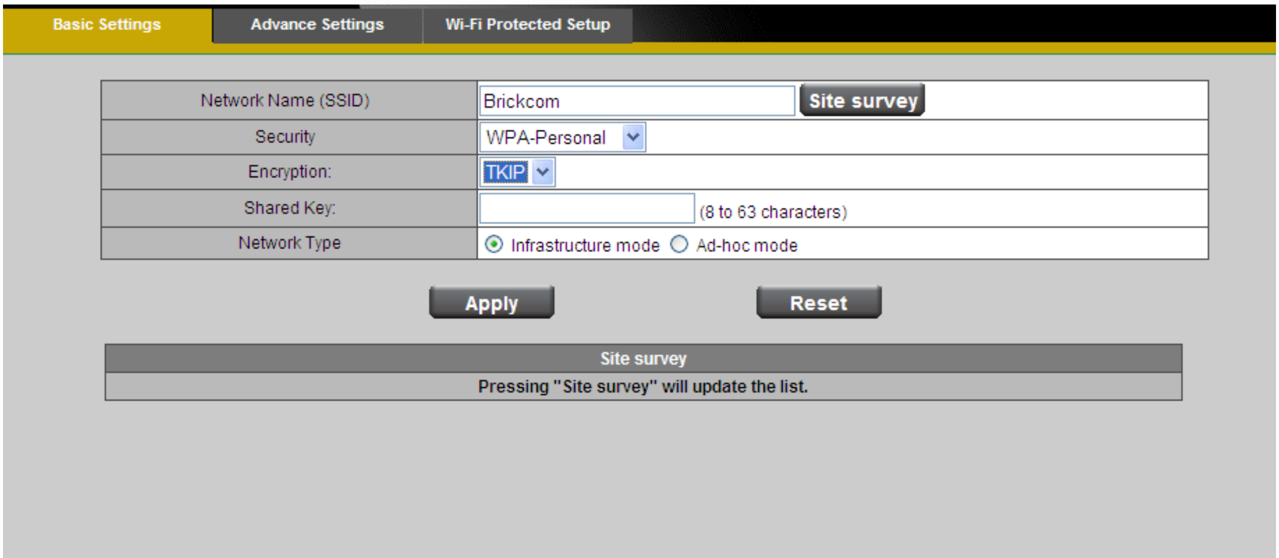
Network Type - Select Infrastructure if your network consists of both wired and wireless devices that communicate through a central device, such as an access point. Select Ad-hoc if your network consists of only wireless devices that communicate with each other directly.

Click Apply or Reset to take effect.

Site Survey

SSID Broadcast, when wireless clients survey the local area for wireless networks to associate with, they will detect the SSID broadcast of the camera.

WPA-Personal



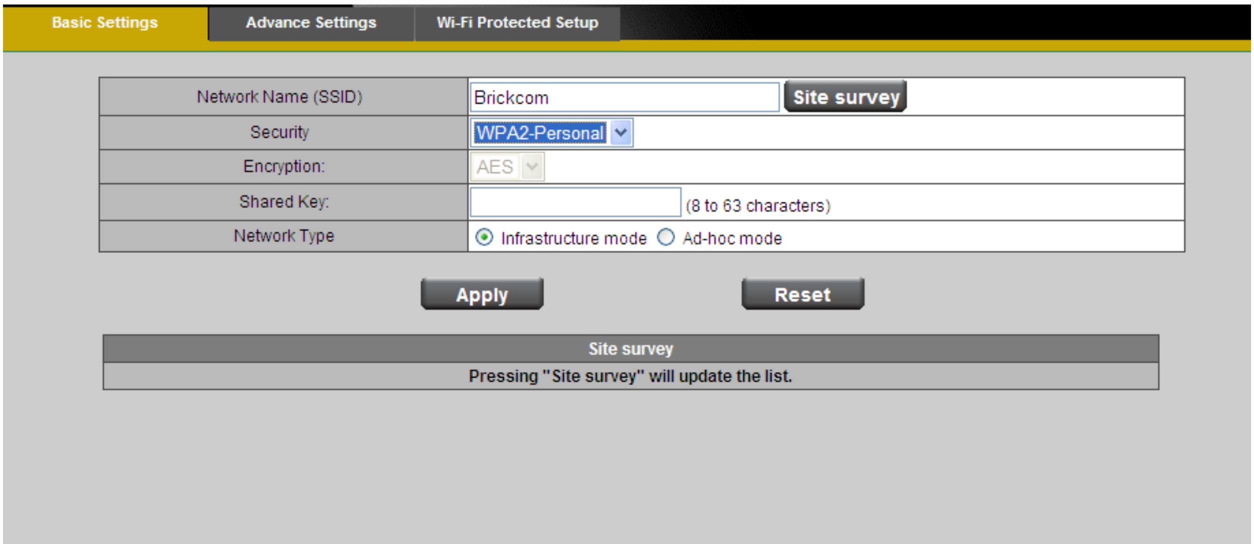
WPA supports two encryption methods, TKIP and AES, with dynamic encryption keys. Select the type of algorithm, TKIP or AES. The default is TKIP.

Shared Key - Enter the key shared between the Router and the server keys. Enter a passphrase of 8-63 characters.

Network Type - Select Infrastructure if your network consists of both wired and wireless devices that communicate through a central device, such as an access point. Select Ad-hoc if your network consists of only wireless devices that communicate with each other directly.

Click **Apply** or **Reset** to take effect.

WPA2-Personal

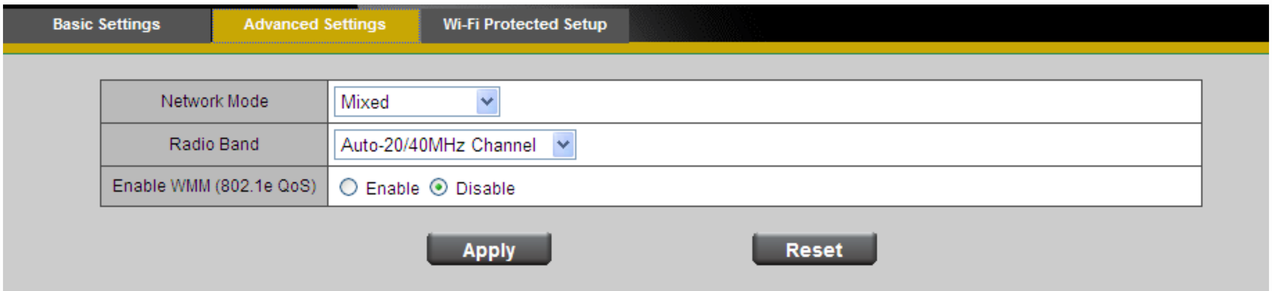


WPA2 supports AES encryption methods with dynamic encryption keys.

Shared Key - Enter the key shared between the Router and the server keys. Enter a passphrase of 8-63 characters.

NOTE: If you are using WPA or WPA2, each device in your wireless network MUST use the same WPA or WPA2 method and shared key, or else the network will not function properly.

Advanced Settings

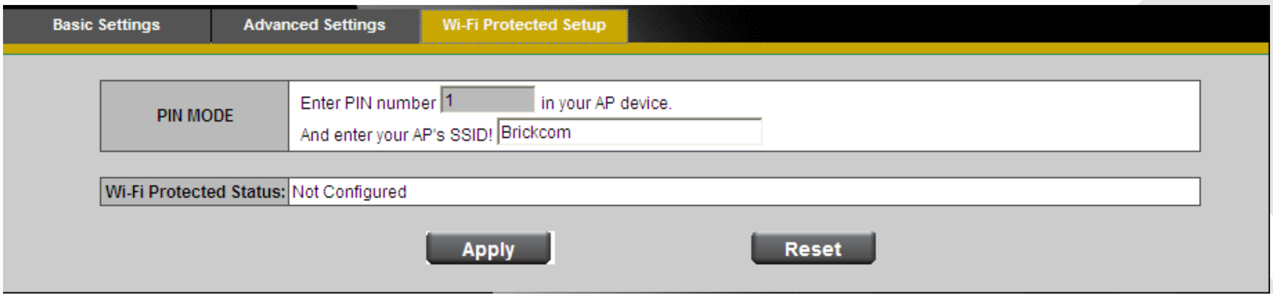


Network Mode - From this drop-down menu, you can select the wireless standards running on your network. If you have both Wireless-B, Wireless-G and Wireless-N (2.4GHz) devices in your network, keep the default setting, **Mixed**. If you have both Wireless-B, Wireless-G devices in your network, select **BG-Mixed**. If you have only Wireless-B devices, select **Wireless-B Only**. If you have only Wireless-G devices, select **Wireless-G Only**. If you have only Wireless-N (2.4GHz) devices, select **Wireless-N Only**.

Radio Band - The settings are available for the Auto-20/40MHz channel and Standard-20 MHz channel. The Auto-20/40MHz channel set up a network using the 20/40MHz band, and the Standard-20 MHz channel set up a network using the 20 MHz band.

Enable WMM (802.1e QoS) - WMM is a wireless Quality of Service feature that improves quality for audio, video, and voice applications by prioritizing wireless traffic. To use this feature, your wireless client devices in your network must support Wireless WMM. If you would like to disable this feature, select **Disabled**. Otherwise, keep the default, **Enabled**.

Wi-Fi Protected Setup

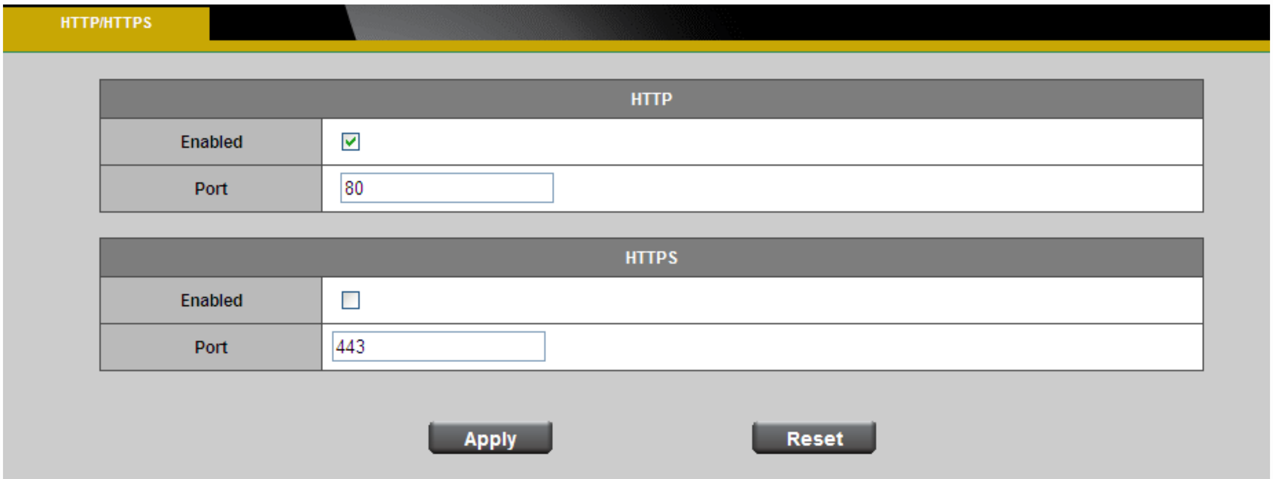


Use this method if your client device has a Wi-Fi Protected Setup PIN number.

1. Enter the PIN number and SSID from the device in the field on the screen.

Click **Apply** or **Reset** to take effect.

HTTP/HTTPS



HTTP - This protocol allows the same quality as TCP protocol without needing to open specific ports for streaming under some network environments. Users inside a firewall can utilize this protocol to allow streaming data through.

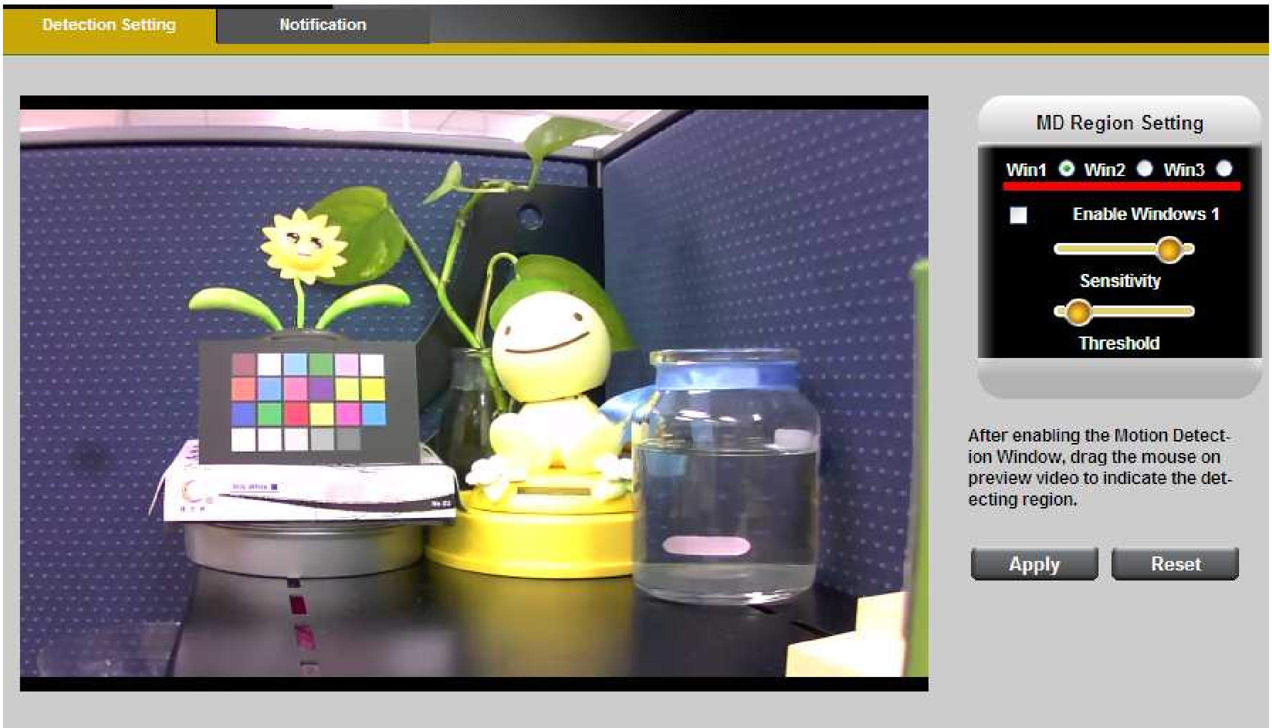
HTTPS - (Hypertext Transfer Protocol over SSL): This section explains how to enable authentication and encrypted communication over SSL (Secure Socket Layer). It helps protect streaming data transmission over the Internet on higher security level.

Click to enable and click **Apply** or **Reset** to take effect.

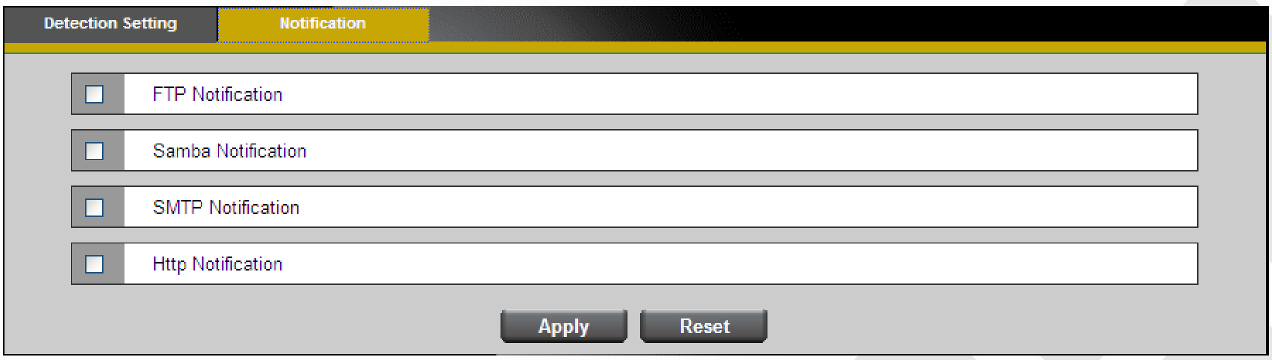
Event

Motion Detection

Motion can be detected by measuring change in speed or vector of an object or objects in the field of view. This section explains how to configure the Network Camera to enable motion detection. There are three motion detection windows can be configured.



Detection Setting - Select and enable the motion detection windows function. Easier to trigger event by higher the sensitivity value and lower the Threshold value.



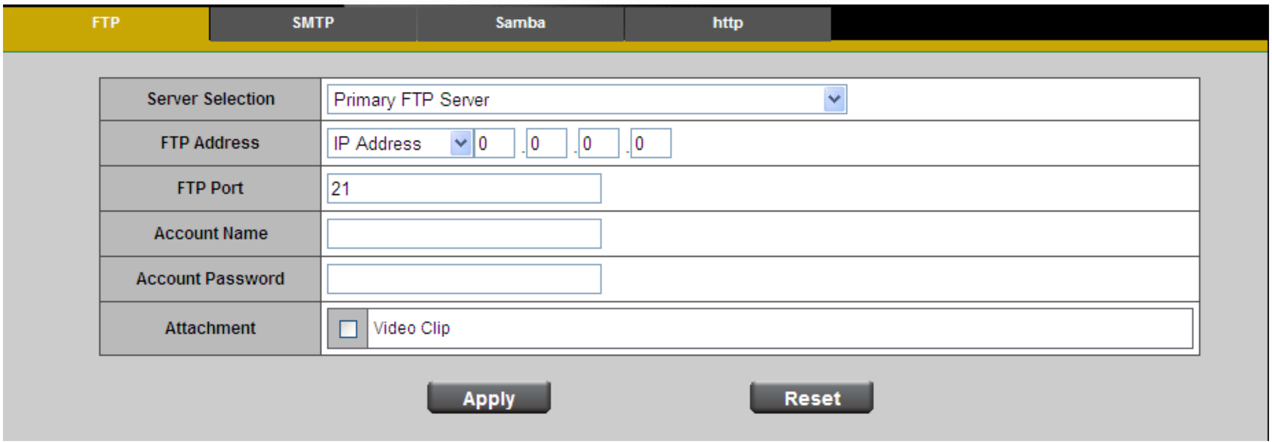
Notification - To react in response to particular events. A typical application is that when a motion is detected, the Network Camera sends buffered images to a FTP server, Samba, SMTP or HTTP as notifications. In this page, you can specify which notification messages will be sent when a trigger is activated. You can configure the Network Camera to send video streaming URL or video clips to your email address or FTP site.

Click **Apply** or **Reset** to take effect.

Notification setting

When an event is triggered, you can specify what kind of action will be performed. You can attach video clip to your email address, FTP site, samba or HTTP.

FTP - File Transfer Protocol (FTP) is often used as an application component to automatically transfer files for program internal functions. Select to send the media files to a FTP server when a trigger is activated. Enter the FTP IP address or hostname; by default, the FTP port server is set to 21, enter account name and password to configure the setting.



Click **Apply** or **Reset** to take effect.

SMTP - Select to send the media files via Email when a trigger is activated.

From - Enter the email address of the sender.

To - Enter the email address of the recipient. Many recipients are separated by commas.

My name - The title shown in the email.

Subject - Enter the subject of the email.

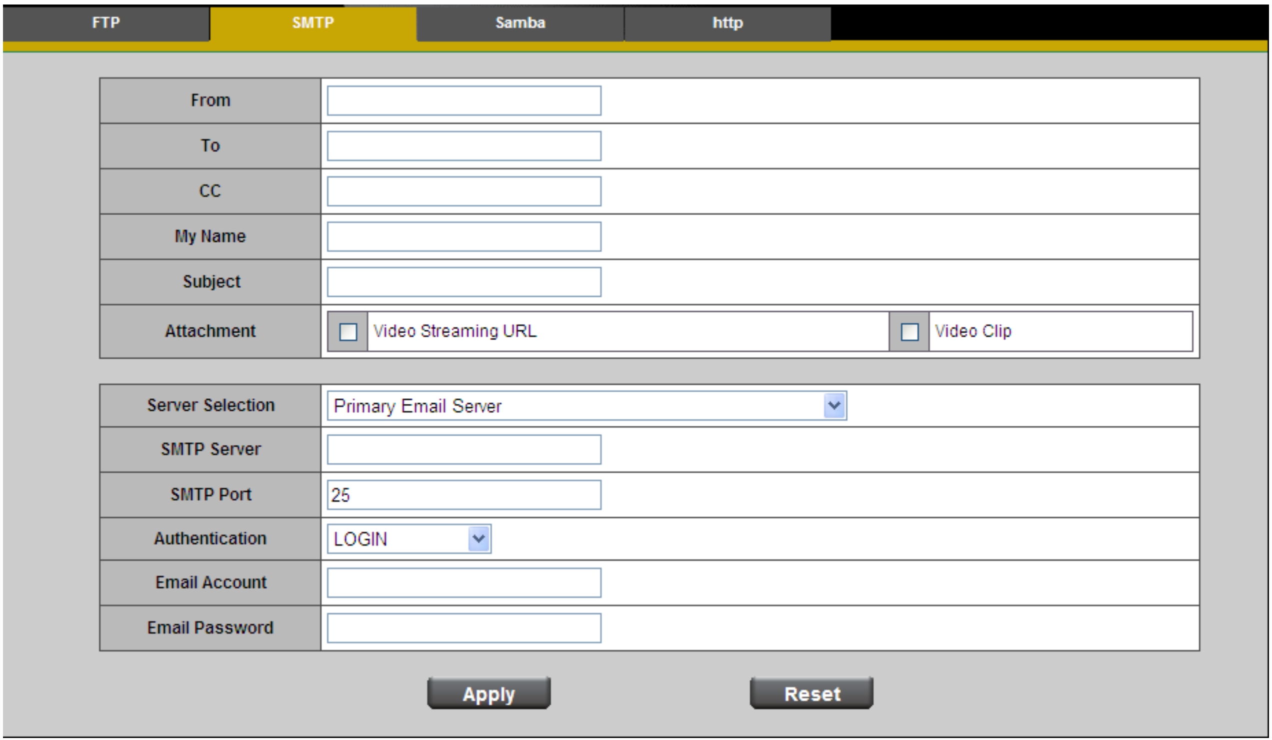
Attached - There are two choices of media types available: video streaming URL and video clip.

SMTP Server and port number - Enter the server host name and port number of the email server.

Authentication - Select the authentication type from the drop-down list.

Email Account - Enter the user name of the email account if necessary.

Email Password - Enter the password of the email account if necessary.



Click **Apply** or **Reset** to take effect.

Samba - Select to send the network file system media files via network neighborhood when a trigger is activated.

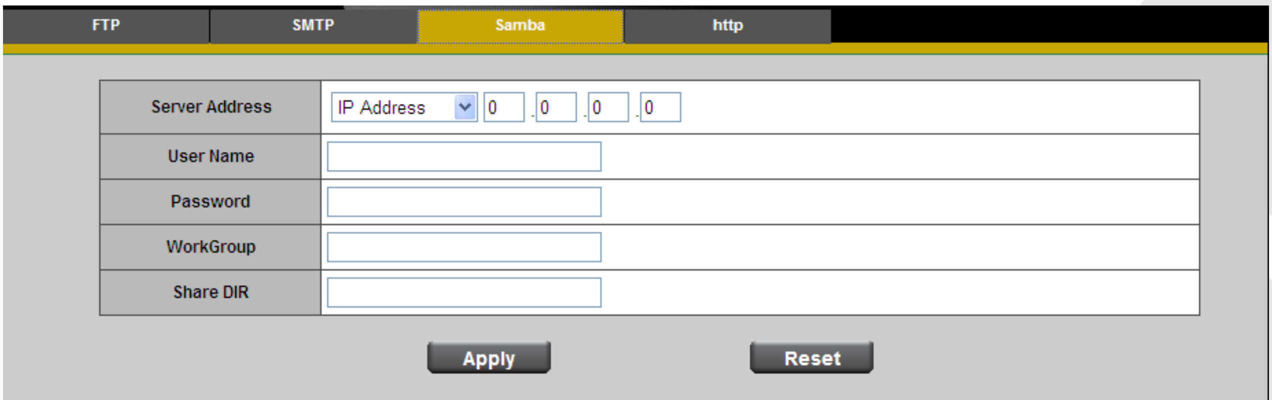
IP Address - Enter the IP address of the samba server.

User Name - Enter the user name of the samba server.

Password - Enter the password of the samba server.

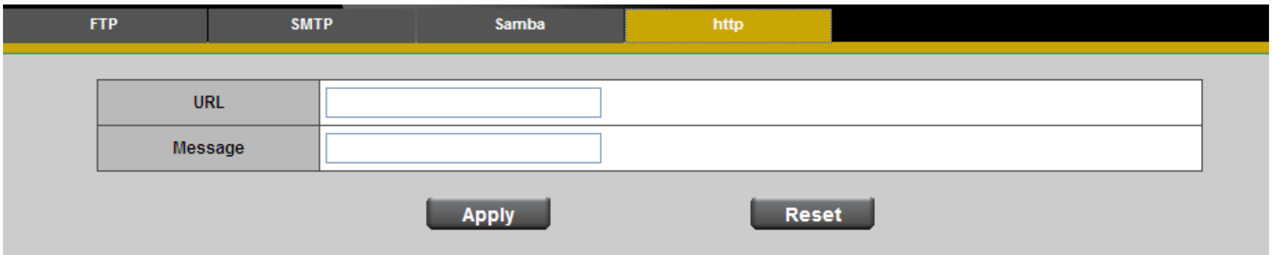
Workgroup - Enter the workgroup of the samba server.

Share DIR - Enter the share DIR of the samba server.



Click **Apply** or **Reset** to take effect.

HTTP - Select to send the HTTP notification when a trigger is activated.



URL – Specify the URL to send HTTP requests, the URL is normally written as follows:

http://ip_address/ notification.cgi?parameter

ip_address – type the IP address or host name of the host to which you want to connect.

Parameter – type the notification parameter if necessary.

Example

URL - http://192.168.1.1/xxxx.cgi

Message - name1=value1&name2=vlaue2

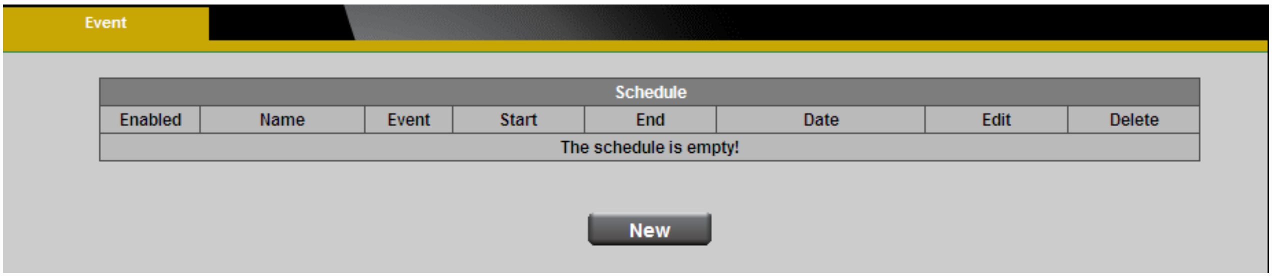
Result - http://192.168.1.1/xxxx.cgi? name1=value1&name2=vlaue2

Ex:

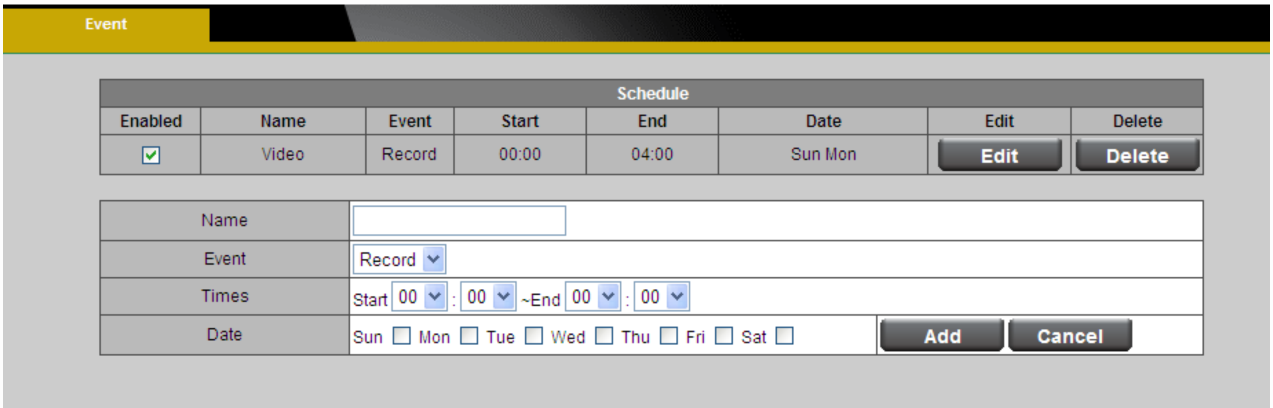
https://192.168.1.1/notification.cgi?event=MD&camera=FB-100A

Message - Enter the message notification that informs you when a trigger is activated.

Scheduled Event



Click **New** to open the recording setting page. In this page, you can define the recording schedule and recording capacity.



Name - Enter a descriptive name for the recording setting.

Event - Select from the drop-down menu for the recording or rebooting event.

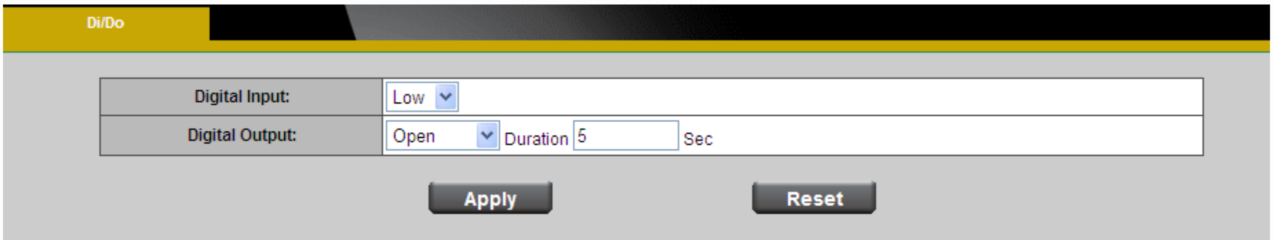
Time - Specify the recording duration.

■ Select the time for recording in 24-hr time format. End time must be more than start time.

■ Select the days on weekly basis.

When completed, Click Add to have recording name appears in the recording list on the recording page. Select **Enabled**; the system begins recording and send recorded file to the Network Storage. To **edit** a recording setting; click Edit to modify. Upon the completion, click update to finish the modification. To remove a recording setting from the list, select a recording name from the list and then click **Delete**. Click New to add more events.

DI/DO



Digital input - Select High or Low to define normal status of the digital input. The Network Camera will report the current status.

Digital output - Select Grounded or Open and enter the duration to define normal status of the digital output.

System

System Log

Log - To send a system log to the network camera when a trigger is activated.

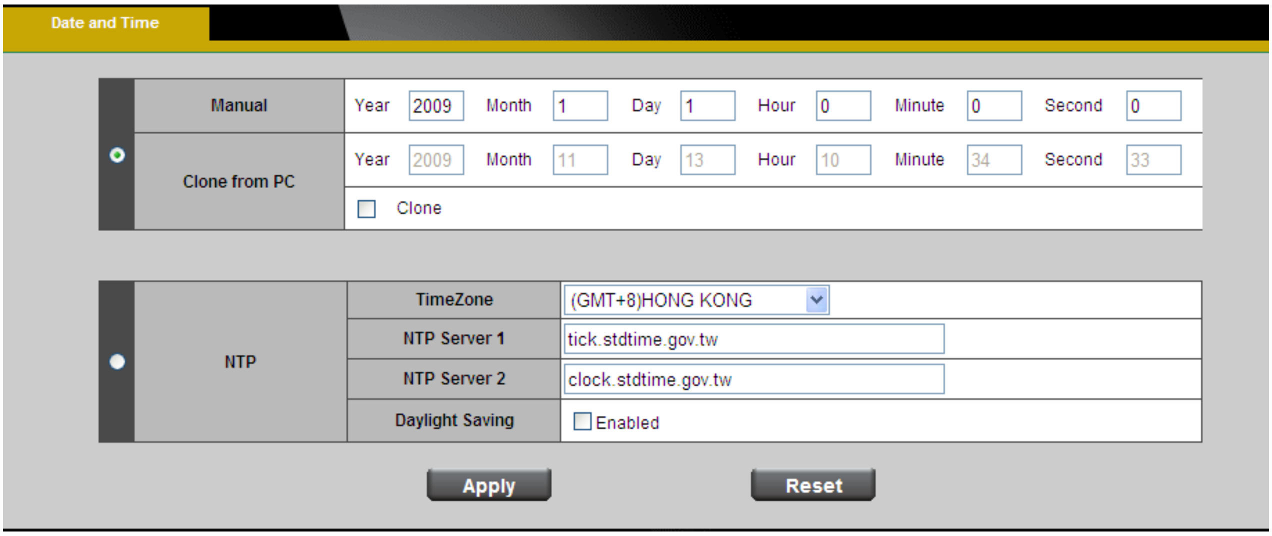
This page displays the system's log in chronological order. The system log is stored in the Network Camera's buffer area and will be overwritten when reaching a certain amount.

Click **Retrieve** to retrieve the log, or click **Save to file** to save the file in the specify location.

Date & Time Settings

Manual - The user enters the date and time manually.

Clone from PC - Sync with computer time; click clone to synchronize the date and time of the Network Camera with the local computer. The read-only date and time of the PC is displayed as updated.



NTP - Select to update the time with the NTP server on hourly, daily, weekly, or monthly basis.

Time Zone - According to your local time zone, select one from the drop-down list.

NTP Server 1 and Server 2 - Enter the address of the NTP server.

Daylight Saving: Enable this option to retain the Daylight Saving Time changes automatically.

Click **Apply** or **Reset** to take effect.

Device Information

Video/Audio Setting - To view the entire video/audio setting information about the network camera.

Video/Audio Settings | Network Settings | System Information

Stream 1	
Video Codec	H264
Video Resolution	1280x800(WXGA)
Video FramRate	25 fps
Video Bitrate	1500 Kbps
Audio Codec	N/A
Multicast IP	N/A
Stream 2	
Video Codec	MJPEG
Video Resolution	1280x800(WXGA)
Video FramRate	25 fps
Video Quality	Not Bad
Audio Codec	N/A
Multicast IP	N/A

Network Setting - To view the entire network setting information about the network camera.

Video/Audio Settings | Network Settings | System Information

IP setting type	Static
IP Address	192.168.1.100
Subnet Mask	255.255.255.0
Default Gateway	192.168.1.2
Primary DNS	192.168.1.2
Secondary DNS	168.95.192.1
UPnP	Enabled
DynDNS	Disabled
TZO	Disabled

System Information - To view the entire system information about the network camera.

Video/Audio Settings | Network Settings | System Information

MAC Address	00:40:25:00:00:33
Firmware Version	v2.3.5.3
Firmware Release	11/11/2009
Product Name	WFB-100A
Model Number	100
Company Name	Brickcom Corporation
Comments	[Fixed Box HD IPCam Pro][Wireless]
UPnP Name	WFB-100A-0033

Storage Management - To view the entire recorded files in SD card.


Storage Management

Local Storage Information	
Total capacity	3.734GB
Used size	350MB
Available size	3.383GB
Remaining time	5 Hours 15 Minutes
Safely Remove Card	Remove

Filename	Size	Time	Delete	Download
200901011031Wed_test.mp4	81,551KB	2009/11/4 Wed 10:40:2	Delete	Download
197001011046Wed_test2.mp4	24,375KB	2009/11/4 Wed 10:50:2	Delete	Download
200911092130Mon_test.mp4	626KB	2009/11/9 Mon 21:30:4	Delete	Download
200911101010Tue_test1.mp4	25,640KB	2009/11/10 Tue 2:15:2	Delete	Download
200901011016Thu_test3.mp4	26,012KB	2009/11/12 Thu 10:19:2	Delete	Download
200901011023Thu_test5-0.mp4	43,330KB	2009/11/12 Thu 10:28:2	Delete	Download
200901011023Thu_test5-1.mp4	43,445KB	2009/11/12 Thu 10:33:2	Delete	Download
200901011023Thu_test5-2.mp4	43,079KB	2009/11/12 Thu 10:38:2	Delete	Download
200901011023Thu_test5-3.mp4	43,441KB	2009/11/12 Thu 10:43:2	Delete	Download
197001011002Thu_test1.mp4	26,766KB	2009/11/12 Thu 10:5:2	Delete	Download

Reload

Click **Remove** to safely remove the storage device.

Click **Reload** to view the list.

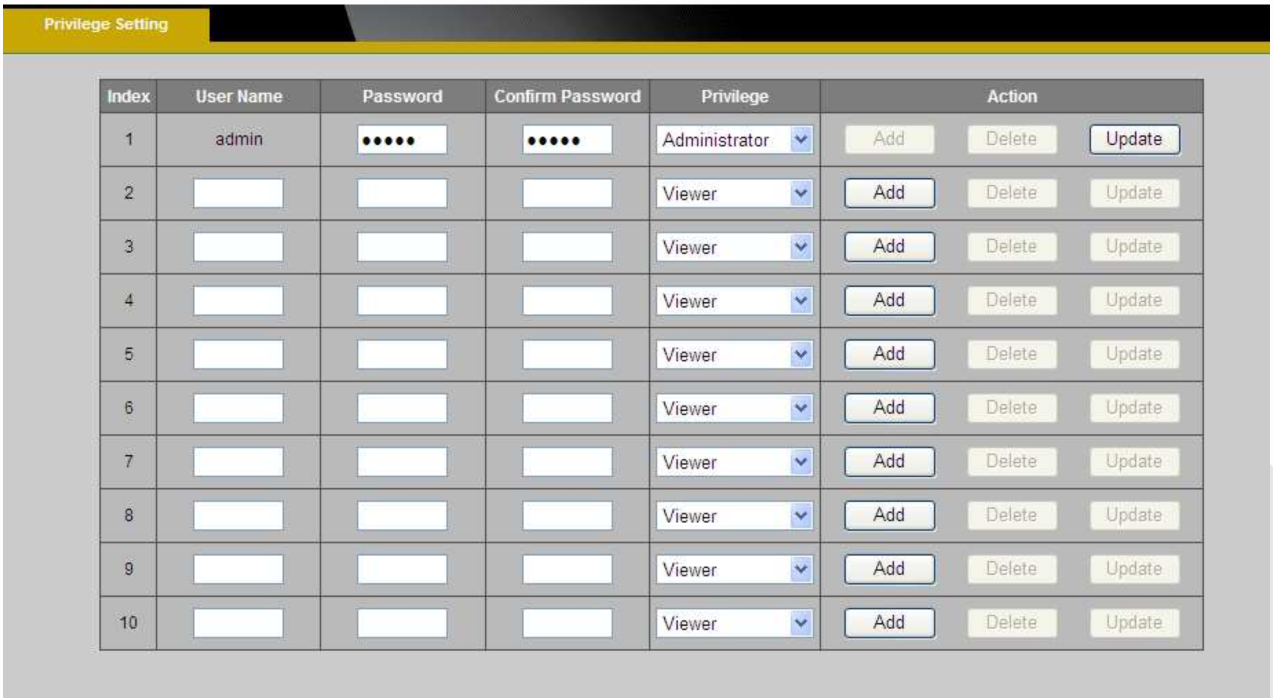
Maintenance

User Management

This section explains how to enable password protection and create multiple accounts.

Privilege Setting - Enter the new user's name and password. Select the privilege for new user account. Click **Add** to take effect. The administrator account name is "admin", which is permanent and can not be deleted.

Access rights are sorted as following (Viewer, Administrator and Remote Viewer). Only administrators can access the Configuration page. Viewers can access the main page for live viewing only. The privilege of Remote Viewer is same as viewer except TCP protocol can only be selected for live viewing page. Administrators can add up to 10 user accounts. Administrator also can change user's access rights or delete user accounts. Select an existing account to modify and make necessary changes; then click **Update** or **Delete** to take effect.

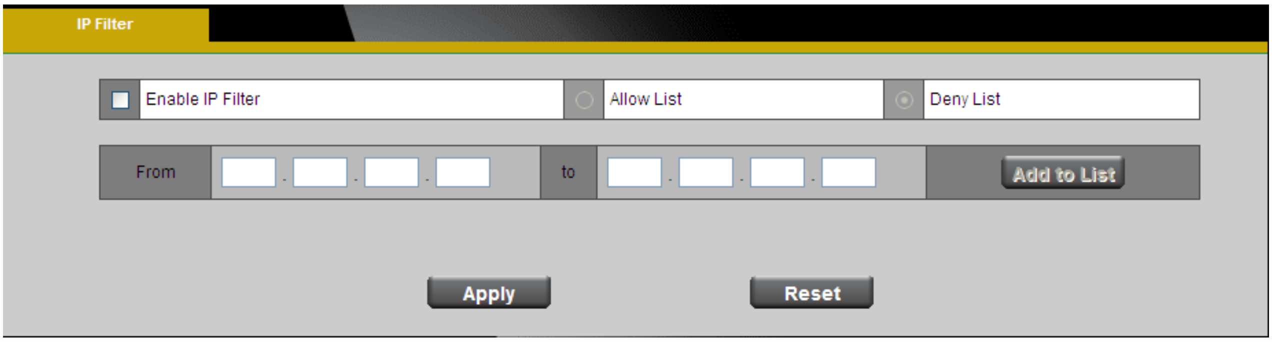


Privilege Setting

Index	User Name	Password	Confirm Password	Privilege	Action		
1	admin	•••••	•••••	Administrator	Add	Delete	Update
2				Viewer	Add	Delete	Update
3				Viewer	Add	Delete	Update
4				Viewer	Add	Delete	Update
5				Viewer	Add	Delete	Update
6				Viewer	Add	Delete	Update
7				Viewer	Add	Delete	Update
8				Viewer	Add	Delete	Update
9				Viewer	Add	Delete	Update
10				Viewer	Add	Delete	Update

IP Filter

IP Filter - Enable the IP filter and set of allow or deny IP address range to server. Click **Add to list** to add the IP range to the IP filter list.

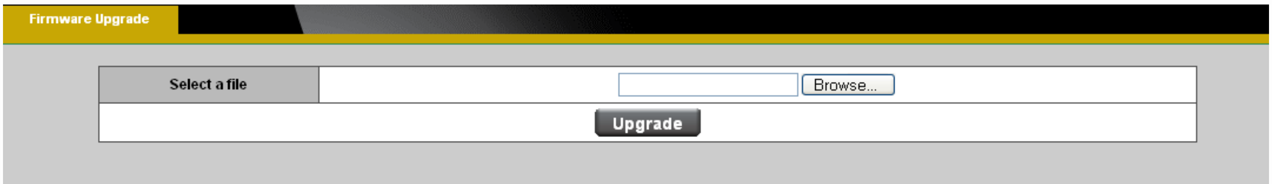


Click **Apply** or **Reset** to take effect.

Firmware Upgrade

This feature allows you to upgrade the firmware on your Network Camera. It takes about few minutes to complete the process. Note that do not power off the Network Camera during the upgrade.

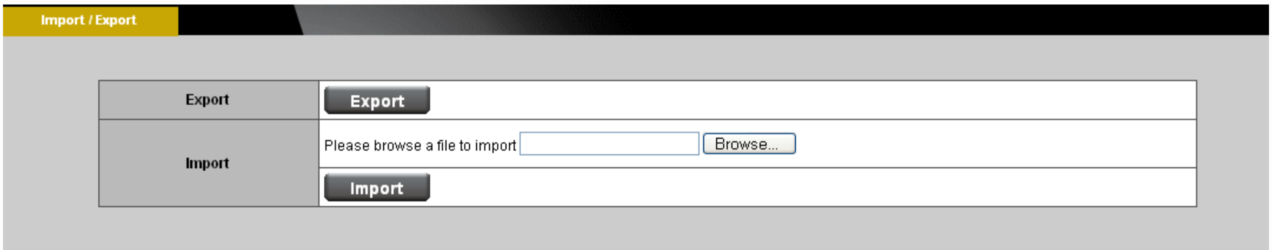
> **Upgrade -** Click **Browse…** and specify the firmware file. Click **Upgrade**. The Network Camera starts to upgrade and will reboot automatically when the upgrade completes.



Configuration

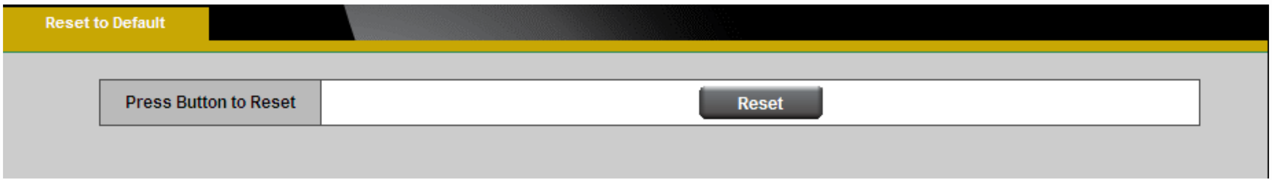
This feature allows you to export/import the configuration files of the network camera.

Import/Export - Click **export** to pop up a dialog to indicate the location and file to export. Click **browse** to indicate the location and file of the camera configuration and click **import** to import the configuration file back into the network camera.

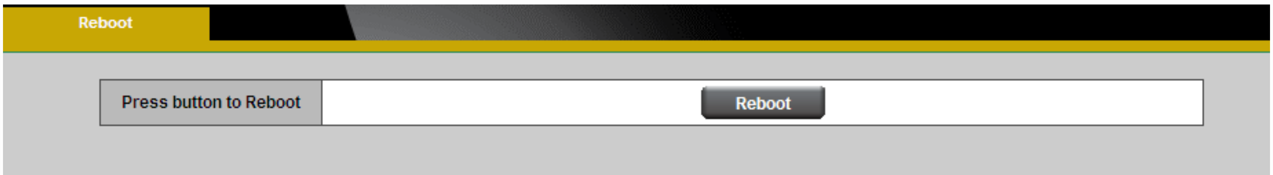


Reset to default

Click **Reset** to restore the network camera to factory default setting.



Reboot



This feature allows you to reboot the Network Camera, which takes about one minute to complete. When completed, the live video page will be displayed in your browser. The following message will show during the rebooting process.

BRICKCOM IPCAM HTTP API

Preface

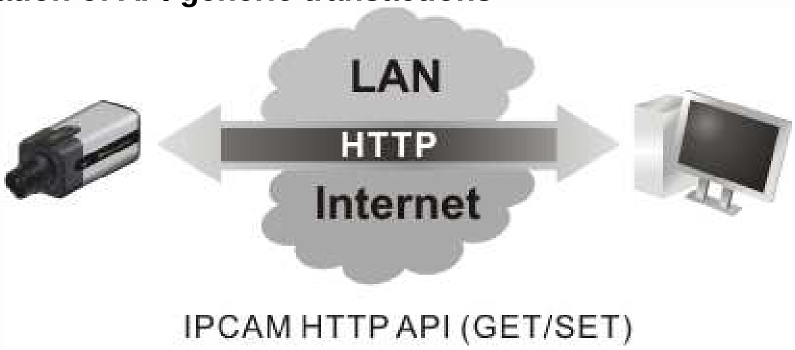
This document specifies the Brickcom IPCAM HTTP API which enables applications to access and/or configure the IP Cameras manufactured by Brickcom over a TCP/IP capable network. Developers who wish to write their own utility should follow the API specification herein.

Overview

Brickcom IPCAM HTTP API is the proprietary network control protocol designed by Brickcom Technology to enable applications to access IP Cameras manufactured by Brickcom. The API allows for configuration of the settings and inquiry of current status on these IP Cameras. The API is structured and transmitted over HTTP protocols and hence is given the name HTTP API.

The complete API is further divided into several categories for ease of management. We dedicate one chapter for each API category to better expound on that API subset.

Figure 1, Illustration of API generic transactions



HTTP API Transaction

An HTTP API transaction is always started with a request from a client application, which is received by the Web server on the IP Camera device and processed by the IP Camera and finally ends with a response sent back to the requesting client.

The client HTTP request takes in either one of the two forms:

- HTTP GET: Normally used to retrieve the settings or status of the IP Camera
- HTTP POST: Normally used to configure the settings of the IP Camera

If the request is successfully received by the IP Camera, the response will contain a HTTP header with a 200 OK response code and the HTTP body with the actual response data or other value if error occurs. An example is provided for each request type below:

Illustration 1, Get the network setting from the IP Camera

Client request

GET http://<IP Camera address>/**network.cgi** HTTP/1.0
…

Server response

HTTP/1.0 200 OK
Content-Type: text/plain

IPAddress=192.168.1.1
SubnetMask=255.255.255.0
…

Illustration 2, Set the network setting from the IP Camera

Client request

POST http://<IP Camera address>/**network.cgi** HTTP/1.0

IPAddress=192.168.1.1
SubnetMask=255.255.255.0

Server response

HTTP/1.0 200 OK
…

Error Response

If the IP Camera is unable to handle the client HTTP API request due to certain conditions such as system busy, incorrect parameters, or any other reason, an appropriate HTTP status code **400 Bad Request** is returned accompanied with an error code and error string that explains the failure.

Client request

GET/POST …

Server response

HTTP/1.0 **400 Bad Request** … ErrorCode=XXX ErrorString=Invalid IP Address

API Categories

The API categories are listed in the table below.

Table 1, API Categories

API Category	**Description**
Streaming	Enable users to set/get the setting about multimedia streaming.
Camera	Enable users to set/get the camera/lens setting.
Audio	Enable user to set/get the audio devices' setting.
Network	Enable users to set/get the network setting.
Event	Enable users to register to listen for notification coming from IPCAM.
Storage	Enable users to configure storage device for storing media content.
System	Enable users to set/get miscellaneous system settings.
Admin	Enables users to perform administrative tasks over the IP Camera.
Capability	Provide users with the list of available features supported by the IP Camera.
Motion detection	Enable user to set/get the motion detection setting and add/delete/update detection region.
Event	Enable user to set/get the event setting and set/get the notification setting.
I/O control	Enable user to control I/O status

Ps: Fields marked in gray are reserved.

Streaming API

Streaming API allows applications to
1) set/get the IP Camera streaming setting
2) help users to view video streaming

Data structures

Data Structure	Description
SVideoFormatSetting	The selected video codec format, encode rate, etc.
SAudioFormatSetting	The selected audio codec format, encode rate, etc.
STransportSetting	The selected network transport.
SVideoSessionSetting	The selected setting of video session used for streaming
SAudioSessionSetting	The selected setting of audio session used for streaming
SChannelSetting	The selected setting of media session (audio+video) used for aggregate streaming.
SChannelSetSetting	The set of available channels on this IPCam

```
enum _ConstantBitrate{
	VBR = 0,
	CBR,
};


enum _bitrateKbps{
	kbps_64 = 64,
	kbps_128 = 128,
	kbps_256 = 256,
	kbps_384 = 384,
	kbps_512= 512,
	kbps_768 = 768,
	kbps_1500 = 1500,
	kbps_2000 = 2000,
	kbps_4000 = 4000,
	kbps_6000 = 6000,
	kbps_8000 = 8000,
	kbps_10000 = 10000,
	kbps_12000 = 12000,

kbps_15000 = 15000,
};


/* SVideoFormatSetting */
typedef struct _videoFormatSetting   {
	int sourceDevice;			// reserved
	char codecType [16];			//
	char codecSubType [16];
	int constantBitrate;			// 0:enabled 1:disabled
	int bitrateInKbps;			// Kbps
	int resolutionWidth;
```

```
    int resolutionHeight;
    int quality;                    // JPEG Specific
    int frameRate;                  // FPS
    int gop;                        // (reserved)

} SVideoFormatSetting;


typedef struct _audioFormatSetting {
    int sourceDevice;           // reserved
    char codecType[16];                 // G711
    char codecSubType[16];              // AUTO
    int numberOfChannel;            // (reserved) Mono, Stereo   =>0
    int sampleRate;                 // (reserved) 8KHZ
    int frameIntervalMS;            //(reserved) 10MS
    int sampleSizeBit;              //(reserved)16 Bit

} SAudioFormatSetting;

/* SMetaFormatSetting */
typedef struct _metaFormatSetting {
    int mdAlarmEnabled;
} SMetaFormatSetting;


/* STransportSetting */
typedef struct _transportSetting {
    int   multicastEnabled;
    char multicastAddress[16];
    int multicastPort;
    int   ttl;                  // 0-255
} STransportSetting;


/* SVideoSessionSetting */
typedef struct _videoSessionSetting {
    int enabled;
    SVideoFormatSetting format;
    STransportSetting transport;
} SVideoSessionSetting;


/* SAudioSessionSetting */
typedef struct _audioSessionSetting {
    int enabled;
    SAudioFormatSetting format;
    STransportSetting transport;
} SAudioSessionSetting;


/* SMetaSessionSetting */
typedef struct _metaSessionSetting {
```

```
        int enabled;
        SMetaFormatSetting format;
        STransportSetting transport;
} SMetaSessionSetting;


/* SChannelSetting */
typedef struct _channelSetting {
        int enabled;
        int index;                      // (Unique) 0: reserved. 1+: valid index
        char name[16];
        int transportType;
        SVideoSessionSetting video;
        SAudioSessionSetting audio;
        SMetaSessionSetting meta;
} SChannelSetting;


/* SChannelSetting */
enum _TransportType {
        TRANSPORT_TYPE_RTSP_RTP=0,
        TRANSPORT_TYPE_RTP_ONLY=1,
        TRANSPORT_TYPE_HTTP=2,
        TRANSPORT_TYPE_MSN=3,
};

typedef struct _channelSetting {
        int enabled;
        int index;                      // (Unique) 0: reserved. 1+: valid index
        char name[16];
        int transportType;      // enum _TransportType
        SVideoSessionSetting video;
        SAudioSessionSetting audio;
        SMetaSessionSetting meta;
} SChannelSetting;


typedef struct _SChannelSetList {
        int size;
        SChannelSetting channels[5];
}SChannelSetList;


/* SChannelSetSetting */
typedef struct _channelSetSetting {
         SChannelSetList channelList;
} SChannelSetSetting;
```

ActionEvents

ActionEvent	Description
getChannels	Get all available channels
getChannel	Get a channel info
addChannel	Add a new channel
updateChannel	Update an existing channel
updateChannels	Update all existing channels
deleteChannel	Delete a channel
getStream	Request to receive a RTSP streaming session

1.1 getChannels

ActionEvent: getChannels

Request	http://<IP>/cgi-bin/channels.cgi&action=get
Response	size = CH1.index=1 CH1.enabled= CH1.name= CH1.transportType= CH1.video.enabled= CH1.video.format.sourceDevice= CH1.video.format.codecType= CH1.video.format.codecSubType= CH1.video.format.constantBitrate= CH1.video.format.bitrateInKbps= CH1.video.format.resolutionWidth= CH1.video.format.resolutionHeight= CH1.video.format.frameRate= CH1.video.format.gop= CH1.video.format.quality= CH1.video.transport.multicastEnabled= CH1.video.transport.multicastAddress= CH1.video.transport.multicastPort= CH1.video.transport.ttl= CH1.audio.enabled= CH1.audio.format.codecType= CH1.audio.format.codecSubType= CH1.audio.transport.multicastEnabled= CH1.audio.transport.multicastAddress= CH1.audio.transport.multicastPort= CH1.audio.transport.ttl= CH1.meta.enabled= CH1.meta.format.mdAlarmEnabled= CH1.meta.transport.multicastEnabled= CH1.meta.transport.multicastAddress= CH1.meta.transport.multicastPort= CH1.meta.transport.ttl=

	Ch2.index=2 ….
Comment	
Method	GET

1.2 getChannel

ActionEvent: getChannel

Request	http://<IP>/cgi-bin/channels.cgi?action=getChannel&index=<index>
Response	enabled= name= transportType= video.enabled= video.format.codecType= video.format.codecSubType= video.format.constantBitrate= video.format.bitrateInKbps= video.format.resolutionWidth= video.format.resolutionHeight= video.format.frameRate= video.format.gop= video.format.quality= video.transport.multicastEnabled= video.transport.multicastAddress= video.transport.multicastPort= video.transport.ttl= audio.enabled= audio.format.codecType= audio.format.codecSubType= audio.transport.multicastEnabled= audio.transport.multicastAddress= audio.transport.multicastPort= audio.transport.ttl= meta.enabled= meta.format.mdAlarmEnabled= meta.transport.multicastEnabled= meta.transport.multicastAddress= meta.transport.multicastPort= meta.transport.ttl=
Comment	
Method	GET

1.3 addChannel

ActionEvent: addChannel

Request	http://<IP>/cgi-bin/channels.cgi action=add index=<index> enabled= name= transportType= video.enabled= video.format.codecType= video.format.codecSubType= video.format.constantBitrate= video.format.bitrateInKbps= video.format.resolutionWidth= video.format.resolutionHeight= video.format.frameRate= video.format.gop= video.format.quality= video.transport.multicastEnabled= video.transport.multicastAddress= video.transport.multicastPort= video.transport.ttl= audio.enabled= audio.format.codecType= audio.format.codecSubType= audio.transport.multicastEnabled= audio.transport.multicastAddress= audio.transport.multicastPort= audio.transport.ttl= meta.enabled= meta.format.mdAlarmEnabled= meta.transport.multicastEnabled= meta.transport.multicastAddress= meta.transport.multicastPort= meta.transport.ttl=
Response	
Comment	
Method	POST

1.4 updateChannel

ActionEvent: updateChannel

Request	http://<IP>/cgi-bin/channels.cgi action=update index=<index> enabled= name= transportType= video.enabled= video.format.codecType= video.format.codecSubType= video.format.constantBitrate= video.format.bitrateInKbps= video.format.resolutionWidth= video.format.resolutionHeight= video.format.frameRate= video.format.gop= video.format.quality= video.transport.multicastEnabled= video.transport.multicastAddress= video.transport.multicastPort= video.transport.ttl= audio.enabled= audio.format.codecType= audio.format.codecSubType= audio.transport.multicastEnabled= audio.transport.multicastAddress= audio.transport.multicastPort= audio.transport.ttl= meta.enabled= meta.format.mdAlarmEnabled= meta.transport.multicastEnabled= meta.transport.multicastAddress= meta.transport.multicastPort= meta.transport.ttl=
Response	
Comment	
Method	POST

1.5 updateChannels

ActionEvent: updateChannels

Request	http://<IP>/cgi-bin/channels.cgi action=updateAll c1Enable=& c1Name=& c1TransportType=& c1VideoEnabled=& c1VideoFormatCodecType=& c1VideoFormatCodecSubType=& c1VideoFormatConstantBitrate=& c1VideoFormatBitrateInKbps =& c1VideoFormatResolutionWidth=& c1VideoFormatResolutionHeight=& c1VideoFormatFrameRate=& c1VideoFormatGop=& c1VideoFormatQuality =& c1VideoTransportMulticastEnabled=& c1VideoTransportMulticastAddress=& c1VideoTransportMulticastPort=& c1VideoTransportTtl=& c1AudioEnabled=& c1AudioFormatCodecType=& c1AudioFormatCodecSubType =& c1AudioTransportMulticastEnabled=& c1AudioTransportMulticastAddress=& c1AudioTransportMulticastPort=& c1AudioTransportTtl=& c1MetaEnabled=& c1MetaFormatMdAlarmEnabled =& c1MetaTransportMulticastEnabled=& c1MetaTransportMulticastAddress=& c1MetaTransportMulticastPort=& c1MetaTransportTtl=& c2Enable=&…….
Response	
Comment	
Method	POST

ActionEvent: deleteChannel

Request	http://<IP>/cgi-bin/channels.cgi action=delete&index=<index>
Response	
Comment	
Method	POST

1.6 getStream

ActionEvent: getStream

Request	rtsp://<IP>/channel<index>
Response	
Comment	<Index> is the index number of the SChannelSetting.
Method	

Camera API

The camera API allows applications to set/get the Camera/lens setting.

Data structures

Data Structure	Description
SWhiteBalanceSetting	White balance setting of the Camera
SBrightnessSetting	Brightness setting of the Camera
SColorSaturationSetting	Color Saturation setting of the Camera
SMirrorFlipSetting	MirrorFlip setting of the Camera
SSharpnessSetting	Sharpness setting of the Camera
SContrastSetting	Contrast setting of the Camera
SFrequencySetting	50Hz / 60Hz switching
SEffectSetting	Special Effect switching
SEnvModeSetting	Indoors / Outdoor switching
SIRCutFilterSetting	IR cut-off filter setting
SIRLEDSetting	IR LED setting
SVideoOverlaySetting	Video overlay setting

```
/* SWhiteBalanceSetting */
enum WhiteBalanceMode {
      WB_MODE_OFF=0,
      WB_MODE_SIMPLE,
      WB_MODE_ADVANCED,

};
/* SAutoExposureSetting */
enum AutoExposureMode {
      AE_MODE_OFF=0,
      AE_MODE_AEC,
      AE_MODE_AGC,
};




/* SExposureSetting */
typedef struct _ExposureSetting   {
      int mode;                              // enum AutoExposureMode
} SExposureSetting;

/* SWhiteBalanceSetting */
typedef struct _whiteBalanceSetting   {
      int mode;                         // enum WhiteBalanceMode
      int level;                     //
} SWhiteBalanceSetting;

/* SBrightnessSetting */
```

```
typedef struct _brightnessSetting {
	int level;				//
} SBrightnessSetting;

/* SColorSaturationSetting */
typedef struct _colorSaturationSetting {
	int level;				//
} SColorSaturationSetting;

/* MirrorFlipSetting */
typedef struct _MirrorFlipSetting {
	int mirror_enabled;
	int flip_enabled;
} SMirrorFlipSetting;

/* SSharpnessSetting */
typedef struct _sharpnessSetting {
	int level;				//
} SSharpnessSetting;

/* SContrastSetting */
typedef struct _contrastSetting	{
	int level;				//
} SContrastSetting;

enum Frequency {
	FREQ_60HZ=0,
	FREQ_50HZ,
};

/* SFrequencySetting */
typedef struct _frequencySetting	{
	int freq;				// 60Hz : 0 , 50Hz : 1
} SFrequencySetting;


enum SpecialEffectMode {
	EFFECT_MODE_DISABLED=0,
	EFFECT_MODE_NEGATIVE,
	EFFECT_MODE_BLACKWHITE,
};

enum IndoorOutdoorMode {
	MODE_OUTDOOR=0,
	MODE_INDOOR,
};


typedef struct _effectSetting	{
	int effectMode;					// enum SpecialEffectMode
} SEffectSetting;
```

```
typedef struct _EnvModeSetting      {
        int envMode;                                    // enum IndoorOutdoorMode
} SEnvModeSetting;


/* SIRCutFilterSetting */
enum IRCutMode {
        IRCUT_MODE_OFF=0,
        IRCUT_MODE_ON,
        IRCUT_MODE_AUTO,
};

typedef struct _IRCutFilterSetting    {
        int mode;                               // enum IRCutMode
        int thresholdLevel;                     // (reserved) 0-100
} SIRCutFilterSetting;


/* SIRLEDSetting */
enum IRLEDMode {
        IRLED_OFF=0,
        IRLED_ON,
        IRLED_MODE_AUTO,

};


typedef struct _IRLEDSetting    {
        int mode;                               // enum IRCutMode
        int thresholdLevel;                     // (reserved) 0-100
} SIRLEDSetting;


/*SAutoIris*/
enum AutoIrisMode {
        AUTOIRIS_DISABLED=0,
        AUTOIRIS_ENABLED,
};
typedef struct _autoIris {
        int enabled;                        //enum AutoIrisMode
}SAutoIris;


/* SVideoOverlaySetting */
enum TimeStampMOde{
        TimeStamp_off=0,
        TimeStamp_on,
};
enum UseImage{
        NO_IMAGE = 0,
        UPLOAD_IMAGE,
};
```

```
typedef struct _OsdPalette {
	int y;		//Range:0~255
	int Cb;		//Range:0~255
	int Cr;		//Range:0~255
} SOsdPalette;
typedef struct _OsdWindow {
	int x;		//Range:depends on resolution
	int y;		//Range:depends on resolution
	int transparent;//Range:0~3
} SOsdWindow;

/* SVideoOverlaySetting */
typedef struct _VideoOverlaySetting {
	int	useTimestamp;			// 0: no timestamp, 1: use timestamp
	char displayString[50];
	int useImage;					// 0: no image, 1: use uploaded image.
	int enabled;
	SOsdPalette osdPalette1;
	SOsdPalette osdPalette2;
	SOsdWindow osdWindow1;
	SOsdWindow osdWindow2;

} SVideoOverlaySetting;
```

ActionEvents

ActionEvent	**Description**
setWhiteBalance	Set white balance
getWhiteBalance	Get white balance
setBrightness	Set brightness
getBrightness	Get brightness
setColorSaturation	Set Color Saturation
getColorSaturation	Get Color Saturation
setMirrorFlip	Set MirrorFlip
getMirrorFlip	Get MirrorFlipof
setSharpness	Set Sharpness
getSharpness	Get Sharpness
setContrast	Set Contrast
getContrast	Get Contrast
setFrequency	Set Frequency
getFrequency	Get Frequency
setEffect	Set Effect
getEffect	Get Effect
setEnvMode	Set EnvMode
getEnvMode	Get EnvMode
setIRCutFilter	Set IR cut Filter
getIRCutFilter	Get IR cut filter
setIRLED	Set IR LED

getIRLED	Get IR LED
setVideoOverlay	Set video overlay
getVideoOverlay	Get video overlay
setCameraSetting	Set all camera setting.
getCameraSetting	Get all camera setting.

2.1 setWhiteBalance

ActionEvent: setWhiteBalance

Request	http://<IP>/cgi-bin/camera.cgi action=**setWhiteBalance** mode= level=
Response	
Comment	
Method	POST

2.2 getWhiteBalance

ActionEvent: getWhiteBalance

Request	http://<IP>/cgi-bin/camera.cgi?action=**getWhiteBalance**
Response	mode= level=
Comment	
Method	GET

2.3 setBrightness

ActionEvent: setBrightness

Request	http://<IP>/cgi-bin/camera.cgi action= **setBrightness** level=
Response	
Comment	
Method	POST

2.4 getBrightness

ActionEvent: getBrightness

Request	http://<IP>/cgi-bin/camera.cgi?action=**getBrightness**
Response	level=
Comment	
Method	GET

2.5 setColorSaturation

ActionEvent: setColorSaturation

Request	http://<IP>/cgi-bin/camera.cgi action= **setColorSaturation** level=
Response	
Comment	
Method	POST

2.6 getColorSaturation

ActionEvent: getColorSaturation

Request	http://<IP>/cgi-bin/camera.cgi?action=**getColorSaturation**
Response	level=
Comment	
Method	GET

2.7 setMirrorFlip

ActionEvent: setMirrorFlip

Request	http://<IP>/cgi-bin/camera.cgi action= **setMirrorFlip** mirrorEnabled = flipEnabled=
Response	
Comment	
Method	POST

2.8 getMirrorFlip

ActionEvent: getMirrorFlip

Request	http://<IP>/cgi-bin/camera.cgi?action= **getMirrorFlip**
Response	flipEnabled= mirrorEnabled =
Comment	
Method	GET

2.9 setSharpness

ActionEvent: setSharpness

Request	http://<IP>/cgi-bin/camera.cgi action= **setSharpness** level=
Response	
Comment	
Method	POST

2.10 getSharpness

ActionEvent: getSharpness

Request	http://<IP>/cgi-bin/camera.cgi?action=**getSharpness**
Response	level=
Comment	
Method	GET

2.11 setContrast
ActionEvent: setContrast

Request	http://<IP>/cgi-bin/camera.cgi action=**setContrast** level=
Response	
Comment	
Method	POST

2.12 getContrast
ActionEvent: getContrast

Request	http://<IP>/cgi-bin/camera.cgi?action=**getContrast**
Response	level=
Comment	
Method	GET

2.13 setFrequcny
ActionEvent: setFrequcny

Request	http://<IP>/cgi-bin/camera.cgi action=**setFrequency** freq =
Response	
Comment	
Method	POST

2.14 getFrequency
ActionEvent: getFrequency

Request	http://<IP>/cgi-bin/camera.cgi?action=**getFrequency**
Response	freq=
Comment	
Method	GET

2.15 setEffect
ActionEvent: setEffect

Request	http://<IP>/cgi-bin/camera.cgi action=**setEffect** effectMode =
Response	
Comment	
Method	POST

2.16 getEffect
ActionEvent: getEffect

Request	http://<IP>/cgi-bin/camera.cgi?action=**getEffect**
Response	effectMode=
Comment	
Method	GET

2.17 setEnvMode

ActionEvent: setEnvMode

Request	http://<IP>/cgi-bin/camera.cgi action=**setEnvMode** envMode =
Response	
Comment	
Method	POST

2.18 getEnvMode

ActionEvent: getEnvMode

Request	http://<IP>/cgi-bin/camera.cgi?action=**getEnvMode**
Response	envMode=
Comment	
Method	GET

2.19 setIRCutFilter

ActionEvent: setIRCutFilter

Request	http://<IP>/cgi-bin/camera.cgi action=**setIRCutFilter** mode= thresholdLevel=
Response	
Comment	
Method	POST

2.20 getIRCutFilter

ActionEvent: getIRCutFilter

Request	http://<IP>/cgi-bin/camera.cgi?action=**getIRCutFilter**
Response	mode= thresholdLevel=
Comment	
Method	GET

2.21 setIRLED

ActionEvent: setIRLED

Request	http://<IP>/cgi-bin/camera.cgi action=**setIRLED** mode= thresholdLevel=
Response	
Comment	
Method	POST

2.22 getIRLED

ActionEvent: getIRLED

Request	http://<IP>/cgi-bin/camera.cgi?action=**getIRLED**
Response	mode= thresholdLevel=
Comment	
Method	GET

2.23 setVideoOverlay

ActionEvent: setVideoOverlay

Request	http://<IP>/cgi-bin/camera.cgi action=**setVideoOverlay** useTimestamp= displayString= useImage= useText= osdPalette1.y= osdPalette1.Cb= osdPalette1.Cr= osdPalette2.y= osdPalette2.Cb= osdPalette2.Cr= osdWindow1.x= osdWindow1.y= osdWindow1.transparent= osdWindow2.x= osdWindow2.y= osdWindow2.transparent=
Response	
Comment	
Method	POST

2.24 getVideoOverlay

ActionEvent: getVideoOverlay

Request	http://<IP>/cgi-bin/camera.cgi?action=**getVideoOverlay**
Response	useTimestamp= displayString= useImage= useText= osdPalette1.y= osdPalette1.Cb= osdPalette1.Cr= osdPalette2.y= osdPalette2.Cb= osdPalette2.Cr= osdWindow1.x= osdWindow1.y= osdWindow1.transparent= osdWindow2.x= osdWindow2.y= osdWindow2.transparent=
Comment	
Method	GET

2.25 setAutoIris

ActionEvent: setAutoIris

Request	http://<IP>/cgi-bin/camera.cgi action= **setAutoIris** enabled
Response	
Comment	
Method	POST

2.26 getAutoIris

ActionEvent: getAutoIris

Request	http://<IP>/cgi-bin/camera.cgi?action= **getAutoIris**
Response	enabled=
Comment	
Method	GET

2.27 setCameraSetting

ActionEvent: setCameraSetting

Request	http://<IP>/cgi-bin/camera.cgi action=**setCameraSetting** whiteBalance.mode=0 whiteBalance.level=0 brightness.level=1 colorSaturation.level=-1 flipEnabled=0 mirrorEnabled=0 sharpness.level=2 contrast.level=0 freq=0 effectMode=0 envMode=1 IRCutFilter.mode=2 IRCutFilter.thresholdLevel=0 IRLED.mode=2 IRLED.thresholdLevel=0 autoIris.enabled=1 videoOverlay.useTimestamp=1 videoOverlay.displayString=HELLO videoOverlay.useImage=0 videoOverlay.useText= videoOverlay.osdPalette1.y=255 videoOverlay.osdPalette1.Cb=128 videoOverlay.osdPalette1.Cr=128 videoOverlay.osdPalette2.y=16 videoOverlay.osdPalette2.Cb=128 videoOverlay.osdPalette2.Cr=128 videoOverlay.osdWindow1.x=0 videoOverlay.osdWindow1.y=13 videoOverlay.osdWindow1.transparent=0 videoOverlay.osdWindow2.x=0 videoOverlay.osdWindow2.y=0 videoOverlay.osdWindow2.transparent=0
Response	
Comment	
Method	POST

2.28 getCameraSetting

ActionEvent: getCameraSetting

Request	http://<IP>/cgi-bin/camera.cgi?action=**getCameraSetting**
Response	whiteBalance.mode=0 whiteBalance.level=0 brightness.level=1 colorSaturation.level=-1 flipEnabled=0 mirrorEnabled=0 sharpness.level=2 contrast.level=0 freq=0 effectMode=0 envMode=1 IRCutFilter.mode=2 IRCutFilter.thresholdLevel=0 IRLED.mode=2 IRLED.thresholdLevel=0 autoIris.enabled=1 videoOverlay.useTimestamp=1 videoOverlay.displayString=HELLO videoOverlay.useImage=0 videoOverlay.useText= videoOverlay.osdPalette1.y=255 videoOverlay.osdPalette1.Cb=128 videoOverlay.osdPalette1.Cr=128 videoOverlay.osdPalette2.y=16 videoOverlay.osdPalette2.Cb=128 videoOverlay.osdPalette2.Cr=128 videoOverlay.osdWindow1.x=0 videoOverlay.osdWindow1.y=13 videoOverlay.osdWindow1.transparent=0 videoOverlay.osdWindow2.x=0 videoOverlay.osdWindow2.y=0 videoOverlay.osdWindow2.transparent=0
Comment	
Method	GET

Audio API

Audio API allows applications to
1) set/get the audio device setting
2) set/get the audio volume of the device

Data structures

Data Structure	Description
SAudioDeviceSetting	Basic audio device setting

```
/* SAudioDeviceSetting */
typedef struct _audioDeviceSetting  {
        int muted;                       // True (muted), False (un-muted)
        int level;                       // volume level 1-100
        int mediaType;                   // (reserved) Full=0, Half duplex=1
        int voiceSource;                 // voice MIC/Line in =>0/1 =>0
} SAudioDeviceSetting;
```

ActionEvents

ActionEvent	Description
setAudioDevice	Set audio device setting
getAudioDevice	Get audio device setting
setAudioMuteState	Mute or un-mute audio
getAudioMuteState	Get the mute state of audio
setAudioVolume	Set audio volume setting
getAudioVolume	Get audio volume setting

3.1 setAudioDevice

ActionEvent: setAudioDevice

Request	http://<IP>/cgi-bin/audio.cgi action=**setAudioDevice** muted= level = voiceSource =
Response	
Comment	
Method	POST

3.2 getAudioDevice

ActionEvent: getAudioDevice

Request	http://<IP>/cgi-bin/ audio.cgi?action=**getAudioDevice**
Response	muted = level = voiceSource =
Comment	
Method	GET

3.3 setAudioMuteState

ActionEvent: setAudioMuteState

Request	http://<IP>/cgi-bin/audio.cgi action=**setAudioMuteState** muted=
Response	
Comment	
Method	POST

3.4 getAudioMuteState

ActionEvent: getAudioMuteState

Request	http://<IP>/cgi-bin/audio.cgi?action=**getAudioMuteState**
Response	muted=
Comment	
Method	GET

3.5 setAudioVolume

ActionEvent: setAudioVolume

Request	http://<IP>/cgi-bin/audio.cgi action=**setAudioVolume** level=
Response	
Comment	
Method	POST

3.6 getAudioVolume

ActionEvent: getAudioVolume

Request	http://<IP>/cgi-bin/audio.cgi?action=**getAudioVolume**
Response	level=
Comment	
Method	GET

Network API

Network API allows applications to set/get the network-related settings including IP address, WIFI network, etc.

Data structures

Data Structure	Description
SBasicNetworkSetting	Basic network setting such as IP address, netmask, etc.
SUPnPSetting	UPnP setting for SSDP advertisement
SDDNSSetting	DDNS setting
SEthernetSetting	Ethernet (802.3?) setting
SWIFISetting	802.11 WLAN setting
SIPFilterSetting	IPFilter setting

/* SBasicNetworkSetting */

```
enum NetAddressType {
	NET_ADDRESS_TYPE_STATIC=0,
	NET_ADDRESS_TYPE_DHCP,
	NET_ADDRESS_TYPE_PPPOE,
};


typedef struct _DHCPSetting {
	// Currently reserved
} SDHCPSetting;

typedef struct _PPPoESetting {
	char username[128];
	char password[128];
} SPPPoESetting;

typedef struct _BasicNetworkSetting	{
	int addressType;			// enum NetAddressType
	char ipv4Address[16];
	char subnetMask[16];
	char gatewayAddress[16];
	char dnsAddress1[16];
	char dnsAddress2[16];
	SDHCPSetting				dhcp;
	SPPPoESetting				pppoe;

	// TBD: IPv6, ….
} SBasicNetworkSetting;
```

/* SUPnPSetting */

```
typedef struct _UPnPSetting {
	int enabled;
```

```
        char upnpName[128];
} SUPnPSetting;
```

/* SDDNSSetting */

```
enum ddnsServerType{
        DYNDNS = 0,
        TZO,
};

typedef struct _SDDNSEntry{
        int wildcardEnabled;//0:disable 1:enable
        char username[128];
        char password[128];
        char hostname[128];
}SDDNSEntry;


typedef struct _DDNSSetting {
        int dyndnsEnabled;
        int tzodnsEnabled;
        SDDNSEntry dyndns;
        SDDNSEntry tzodns;
} SDDNSSetting;




/* SEthernetSetting */
enum EthernetMediaType {
        ETHER_MEDIA_TYPE_AUTO=0,
        ETHER_MEDIA_TYPE_10_HALF_DUPLEX,
        ETHER_MEDIA_TYPE_10_FULL_DUPLEX,
        ETHER_MEDIA_TYPE_100_HALF_DUPLEX,
        ETHER_MEDIA_TYPE_100_FULL_DUPLEX,
        ETHER_MEDIA_TYPE_1000_FULL_DUPLEX,
};

typedef struct _EthernetSetting {
        Int mediaType;      // enum EthernetMediaType
} SEthernetSetting;


/* SWIFISetting */
enum WIFIWPA_algorithmType {
        WL_TKIP=0,
        WL_AES,
        WL_TKIP_AES,
};

enum WIFIWEP__authenticationType {
        WL_OPEN=0,
        WL_SHARED,
```

```
        WL_WEPAUTO,
};

enum WIFISecurityMode {
        WL_NONE=0,
        WL_WEP,
        WL_WPAPSK,
        WL_WPA2PSK,
        //WL_WPA_ENTERPRISE,
        //WL_WPA2_ENTERPRISE,
};

enum WIFIAccessMode {
        WIFI_ACCESS_MODE_INFRASTRUCTURE=0,
        WIFI_ACCESS_MODE_ADHOC,
};

enum WIFIOperationMode {
        WIFI_OP_MODE_AUTO=0,
        WIFI_OP_MODE_11G_ONLY,
        WIFI_OP_MODE_11B_ONLY,
        WIFI_OP_MODE_11N_ONLY,
        WIFI_OP_MODE_11BG_MIXED,
        WIFI_OP_MODE_11GN_MIXED,
        WIFI_OP_MODE_11BGN_MIXED,
};

enum WIFIPreambleType {
        WIFI_PREAMBLE_TYPE_LONG=0,
        WIFI_PREAMBLE_TYPE_SHORT,
};

enum WIFIAuthenticationType {
        WIFI_AUTHENTICATION_TYPE_OPEN=0,
        WIFI_AUTHENTICATION_TYPE_SHARED_KEY,
};

enum WIFIchannelBandWidth {
        FORTY_MHZ=0,
        TWENTY_MHZ,
};


enum WIFIWPSMode {
        NONE=0,
        PIN,
        PBC,
};
```

```
typedef struct _SSWPS {
        int WPSMode;                // enum WIFIWPSMode
        char PINCode[64];
}SWPS;


typedef struct _SSWPA {
        int algorithmType;                // enum WIFIWPA_algorithmType
        char sharedKey[64];
}SWPA;


typedef struct _SSKeyentry {
      char encryptionKey[64];
}SKeyentry;

typedef struct _SSEncryptionKeyList {
        int size;
      SKeyentry keyEntry[4];
}SEncryptionKeyList;

typedef struct _SSWEP {
        int authenticationType;               // enum WIFIWEP__authenticationType
        int defaultTransmitKeyIndex;
        int wepKeyLength;
        SEncryptionKeyList encryptionKeyList;
}SWEP;

//================= IEEE 802.1X ======================
//authenticationProtocolType
enum IEEE_802_1x_authenticationProtocolType {
      WL_EAP_TLS=0,
      WL_EAP_TTLS,
      WL_EAP_PEAP,
      WL_EAP_FAST,
      WL_EAP_LEAP,
};
//authenticationMethod
enum IEEE_802_1x_authenticationMethod {
      WL_MSCHAP=0,
      WL_MSCHAPV2,
      WL_PAP,
      WL_EAP_MD5,
};

//innerEAPProtocolType
enum IEEE_802_1x_innerEAPProtocolType {
      WL_INNER_EAP_TLS=0,
      WL_EAP_OTP,
};
```

```
typedef struct _IEEE_802_1xSetting {
        int enabled;
        int authenticationProtocolType; //enum authenticationProtocolType
        int innerTTLSAuthenticationMethod; //enum authenticationMethod
        int innerEAPProtocolType;//enum innerEAPProtocolType
        int validateServerEnabled;
        char userName[65];
        char password[65];
        char anonymousID[65];
        int autoPACProvisioningEnabled;
        int caline;
        int clientline;
        int PACline;
} SIEEE_802_1xSetting;

typedef struct _WIFISetting {
      int enabled;
      int mode;                      // enum WIFIAccessMode
      int operationMode;             // WIFIOperationMode
      int channel;                   // (0) Auto,
      int wmm;                       // 0:disabled 1:enabled
      char SSID[31];
      int preamble;              // enum WIFIPreambleType
      int    rtsThreshold;           //
      int fragmentationThreshold;
      int authentication;        // enum WIFIAuthenticationType
      int channelBandWidth;      // enum WIFIchannelBandWidth
      int securityMode;              // enum WIFISecurityMode
      SWEP WEP;
      SWPA WPA;
      SWPS WPS;
      SIEEE_802_1xSetting wl_802_1x;
} SWIFISetting;

enum IPFilterPermissionType {
      Deny=0,
      Allow,
};

typedef struct _SSFilterAddressEntry {
      int enabled;
      char startIP[16];
      char endIP[16];
}SFilterAddressEntry;

typedef struct _SSFilterAddressList {
        int size;
        SFilterAddressEntry filterEntry[16];
}SFilterAddressList;

typedef struct _SSIPFilterSetting {
```

```
        int enabled;
        int permissionType;
        SFilterAddressList allowList;
        SFilterAddressList denyList;
}SIPFilterSetting;
```

ActionEvents

ActionEvent	Description
setBasicNetwork	Set the basic network setting
getBasicNetwork	Get the basic network setting
setUPnP	Set UPnP setting
getUPnP	Get UPnP setting
setDDNS	Set DDNS setting
getDDNS	Get DDNS setting
setEthernet	Set Ethernet setting
getEthernet	Get Ethernet setting
setWIFI	Set WIFI setting
getWIFI	Get WIFI setting
setIPFilter	Set IPFilter setting
getIPFilter	Get IPFilter setting

4.1 setBasicNetwork

ActionEvent: setBasicNetwork

Request	http://<IP>/cgi-bin/basicNetwork.cgi
	action= **set**
	//STATIC
	addressType=0
	ipv4Address=
	subnetMask=
	gatewayAddress=
	dnsAddress1=
	dnsAddress2=
	// DHCP,
	addressType=1
	// PPPOE
	addresssType=2
	pppoe.username=
	pppoe.password=

Response	
Comment	
Method	POST

4.2 getBasicNetwork

ActionEvent: getBasicNetwork

Request	http://<IP>/cgi-bin/basicNetwork.cgi?action=**get**
Response	addressType= (0=Static,1=DHCP, 2=PPPoE) ipv4Address= subnetMask= gatewayAddress= dnsAddress1= dnsAddress2= pppoe.username= pppoe.password=
Comment	
Method	GET

4.3 setUPnP

ActionEvent: setUPnP

Request	http://<IP>/cgi-bin/upnp.cgi action=**set** enabled= name=
Response	
Comment	
Method	POST

4.4 getUPnP

ActionEvent: getUPnP

Request	http://<IP>/cgi-bin/upnp.cgi?action=**get**
Response	enabled= name=
Comment	
Method	GET

4.5 setDDNS

ActionEvent: setDDNS

Request	http://<IP>/cgi-bin/ddns.cgi action=set dyndnsEnabled= dyndns.wildcardEnabled= dyndns.username= dyndns.password= dyndns.hostname= tzodnsEnabled= tzodns.wildcardEnabled= tzodns.username= tzodns.password= tzodns.hostname=
Response	
Comment	
Method	POST

4.6 getDDNS

ActionEvent: getDDNS

Request	http://<IP>/cgi-bin/ddns.cgi? action=**get**
Response	dyndnsEnabled=0 dyndns.wildcardEnabled= dyndns.username= dyndns.password= dyndns.hostname= tzodnsEnabled= tzodns.wildcardEnabled= tzodns.username= tzodns.password= tzodns.hostname=
Comment	
Method	GET

4.7 setEthernet

ActionEvent: setEthernet

Request	http://<IP>/cgi-bin/ethernet.cgi action=**set** mediaType=
Response	
Comment	
Method	POST

4.8 getEthernet

ActionEvent: getEthernet

Request	http://<IP>/cgi-bin/ethernet.cgi?action=**get**
Response	mediaType=
Comment	
Method	GET

4.9 setWIFI

ActionEvent: setWIFI

Request	http://<IP>/cgi-bin/wifi.cgi action=**set** enabled= mode= operationMode= channel= SSID= preamble= rtsThreshold= fragmentationThreshold= authentication= channelBandWidth= securityMode= WEP. authenticationType= WEP. defaultTransmitKeyIndex = WEP. wepKeyLength = WEP. encryptionKeyList. Keyentry1.encryptionKey= WEP. encryptionKeyList. Keyentry2.encryptionKey= WEP. encryptionKeyList. Keyentry3.encryptionKey= WEP. encryptionKeyList. Keyentry4.encryptionKey= WPA. algorithmType= WPA.sharedKey= WPS.WPSMode= WPS.PINCode=
Response	
Comment	
Method	POST

4.10 getWIFI

ActionEvent: getWIFI

Request	http://<IP>/cgi-bin/wifi.cgi? action=**get**
Response	enabled= mode= operationMode= channel= SSID= preamble= rtsThreshold= fragmentationThreshold= authentication= channelBandWidth= securityMode= (a.) securityMode=0 return Nothing!! (b.) securityMode=1 WEP. authenticationType= WEP. defaultTransmitKeyIndex = WEP. wepKeyLength= WEP. encryptionKeyList.Keyentry1.encryptionKey= WEP. encryptionKeyList.Keyentry2.encryptionKey= WEP. encryptionKeyList.Keyentry3.encryptionKey= WEP. encryptionKeyList.Keyentry4.encryptionKey= (c.) securityMode=2 WPA. algorithmType= WPA.sharedKey= (d.) securityMode=3 WPA. algorithmType= WPA.sharedKey= WPS.WPSMode= WPS.PINCode
Comment	
Method	GET

4.11 setIPFilter

ActionEvent: setIPFilter

Request	http://<IP>/cgi-bin/IPFilter.cgi action=**set** permissionType= enabled= allow.enabled1= allow.startIP1= allow.endIP1= allow.enabled2= allow.startIP2= allow.endIP2= ……. deny.enabled1= deny.startIP1= deny.endIP1= deny.enabled2= deny.startIP2= deny.endIP2=
Response	
Comment	
Method	POST

4.12 getIPFilter

ActionEvent: getIPFilter

Request	http://<IP>/cgi-bin/ IPFilter.cgi? action=**get**
Response	enabled= permissionType= allow.size= allow.enabled1= allow.startIP1= allow.endIP1= allow.enabled2= allow.startIP2= allow.endIP2= ……. deny.size= deny.enabled1= deny.startIP1= deny.endIP1= deny.enabled2= deny.startIP2= deny.endIP2=
Comment	
Method	GET

Storage API (TBD)

Storage API allows applications to configure the storage devices reachable by the IPCAM unit.

Data structures

Data Structure	Description

ActionEvents

ActionEvent	Description

ActionEvent:

Request	http://<IP>/cgi-bin/stream. l?action=
Response	
Comment	
Method	

System API

System API allows applications to configure miscellaneous system settings not covered by any other category. These settings include Time, Syslog, and etc.

// NOTE: In the future, we may switch to rsyslog instead of syslogd.

Data structures

Data Structure	Description
SDeviceInfo	IP Camera device info
STimeSetting	Time setting
SSyslogSetting	Syslog setting
SSystemStatus	Structure containing system status info

```
/* SDeviceInfo */
typedef struct _SSDeviceInfo {
        char chipVersion[65];
        char sensorID[65];
        char macAddress[17];
        char firmwareVersion[65];
        char firmwareReleasedDate[65];
        char InternalName[65];
        char ProductName[65];
        char ModelNumber[16];
        char CompanyName[32];
        char Comments[128];
} SDeviceInfo;
```

```
/* STimeSetting */
enum TimeConfigType {
        TIME_CONFIG_TYPE_NONE=0,
        TIME_CONFIG_TYPE_MANUAL,
        TIME_CONFIG_TYPE_NTP,
};

// TODO: TBD.
enum TimeZoneID    {
         TIME_ZONE_MIN,
              TIME_ZONE_KWAJALEIN,
              TIME_ZONE_SAMOA,
              TIME_ZONE_HAWAII,
              TIME_ZONE_ALASKA,
              TIME_ZONE_LOS_ANGELES,
              TIME_ZONE_PHOENIX,
              TIME_ZONE_MEXICO_CITY,
              TIME_ZONE_NEW_YORK,
              TIME_ZONE_SANTIAGO,
```

```
            TIME_ZONE_SAO_PAULO,
            TIME_ZONE_NORONHA_ISLAND,
            TIME_ZONE_PRAIA,
            TIME_ZONE_LONDON,
            TIME_ZONE_PARIS,
            TIME_ZONE_CAIRO,
            TIME_ZONE_MOSCOW,
            TIME_ZONE_DUBAI,
            TIME_ZONE_KARACHI,
            TIME_ZONE_DHAKA,
            TIME_ZONE_JAKARTA,
            TIME_ZONE_HONG_KONG,
            TIME_ZONE_TOKYO,
            TIME_ZONE_SYDNEY,
            TIME_ZONE_NOUMEA,
            TIME_ZONE_NewZealand,
            TIME_ZONE_MAX
};

 // Reserved for internal use...
typedef struct _TimeZone {
      int id;           // Time zone id.
      Char TZSyntax[128];
} STimeZone;

typedef struct _TimeZoneList {
      int size;
      STimeZone   timezone[60];
} STimeZoneList;


typedef struct _ManualTimeSetting {
      int year;
      int month;
      int day;
      int hour;
      int minute;
      int second;
} SManualTimeSetting;


typedef struct _NTPTimeSetting {
      char ntpServerLoc1[100];  // IP address or FQDN of NTP server
      char ntpServerLoc2[100];
} SNTPTimeSetting;


typedef struct _TimeSetting
{
      int type;                 // enum TimeConfigType
      int enableDST;                  // Daylight saving.   (0: disabled, 1: enabled)
      int timezoneID;                 // enum TimeZoneID
```

```
        SManualTimeSetting   manual;
        SNTPTimeSettingntp;
} STimeSetting;
```

/* SSyslogSetting */

```
// Note, these values are taken from manpage for syslog (3).
enum LogPriority {
        SLOG_EMERG=0,           // system is unusable
        SLOG_ALERT,         // action must be taken immediately
        SLOG_CRIT,          // critical conditions
        SLOG_ERR,           // error conditions
        SLOG_WARNING,       // warning conditions
        SLOG_NOTICE,        // normal, but significant, condition
        SLOG_INFO,              // informational message
        SLOG_DEBUG,         // debug-level message
};
enum AddressFormatType {
        IP_TYPE,
        HOSTNAME_TYPE,
};

Typedef struct _SyslogSetting {
        int localLogLevel;      // Log with LogPriority value smaller than this is logged to local
file.
        Int useRemoteLog;                   // 0: disabled, 1: enabled
        int addressingFormatType;
        char remoteServerAddress[128];      // IP address or FQDN of the syslog server
        int remoteServerPort;               // Port number of the syslog server
} SSyslogSetting;

Typedef struct _systemStatus
{
        // TBD
} SSystemStatus;
```

ActionEvents

ActionEvent	Description
getDeviceInfo	Get device info
setTimeSetting	Set time setting
getTimeSetting	Get time setting
setSyslogSetting	Set syslog setting
getSyslogSetting	Get syslog setting
getSyslogFile	Get syslog file.
SyslogClear	Clear syslog.
getSystemStatus	Get system status

5.1 getDeviceInfo

ActionEvent: getDeviceInfo

Request	http://<IP>/cgi-bin/system.cgi?action=**get**
Response	chipVersion= sensorID= macAddress= firmwareVersion= firmwareReleasedDate= InternalName= ProductName= ModelNumber= CompanyName= Comments=
Comment	
Method	GET

5.2 setTimeSetting

ActionEvent: setTimeSetting

Request	http://<IP>/cgi-bin/time.cgi action=**set** type=0 or == type=1 enableDST= timezoneID= manual.year= manual.month= manual.day= manual.hour= manual.minute= manual.second= or == type=2 enableDST= timezoneID= ntp.ntpServerLoc1= ntp.ntpServerLoc2=

Response	
Comment	
Method	POST

5.3 getTimeSetting

ActionEvent: getTimeSetting

Request	http://<IP>/cgi-bin/time.cgi?action=**get**
Response	type= enableDST= timezoneID= manual.year= manual.month= manual.day= manual.hour= manual.minute= manual.second= enableDST= timezoneID= ntp.ntpServerLoc1= ntp.ntpServerLoc2=
Comment	
Method	GET

5.4 setSyslogSetting

ActionEvent: setSyslogSetting

Request	http://<IP>/cgi-bin/syslog.cgi action=**set** localLogLevel= useRemoteLog= addressingFormatType= remoteServerAddress= remoteServerPort=
Response	
Comment	
Method	POST

5.5 getSyslogSetting

ActionEvent: getSyslogSetting

Request	http://<IP>/cgi-bin/syslog.cgi ?action=**get**
Response	localLogLevel= useRemoteLog= addressingFormatType= remoteServerAddress= remoteServerPort=
Comment	
Method	GET

5.6 getSyslogFile

ActionEvent: getSyslogFile

Request	http://<IP>/syslog.dump
Response	Content of syslog.
Comment	
Method	GET

5.7 syslogClear

ActionEvent: syslogClear

Request	http://<IP>/cgi-bin/syslog.cgi?action=clear
Response	
Comment	Clear syslog.
Method	GET

ActionEvent: getSystemStatus

Request	**http://<IP>/cgi-bin/systemStatus.cgi?action=get**
Response	
Comment	
Method	**GET**

Admin API

Admin API enables applications to perform administrative tasks on the IPCAM unit. The tasks include add/delete users, upgrade firmware, etc.

Data structures

Data Structure	Description
SUserSetting	Setting for a user account
SUserSetSetting	All user accounts
SHTTPSetting	HTTP setting
SHTTPSSetting	HTTPS setting

ActionEvents

ActionEvent	Description
addUser	Add a user to the system
deleteUser	Delete a user from the system
updateUser	Update the account of user <username>
getUsers	Get all user accounts
setHTTP	Set HTTP setting
setHTTP/HTTPS	Set HTTP/HTTPS in one request.
getHTTP	Get HTTP setting
setHTTPS	Set HTTPS setting
getHTTPS	Get HTTPS setting
resetToDefault	Reset the IPCamera setting to factory default.
upgradeFirmware	Upgrade firmware
Reboot	Reboot the system.
importConfigFile	This function is used to upload configuration to the device.
exportConfigFile	This function is used to get the configuration from the device.
setPWDComplexity	Set password Complexity.
getPWDComplexity	Get password Complexity.

```
enum UserPrivilegeType {
        USER_PRIVILEGE_VIEW=0,
        USER_PRIVILEGE_ADMIN,
        USER_PRIVILEGE_REMOTE_VIEW,
};

/* SUserSetting */
typedef struct _userSetting {
        int index;
        char username[30];   // Unique key.
        char password[30];
        int privilege;          // Administration, Viewer
} SUserSetting;
\
/* SUserSetSetting */
typedef struct _userSetList {
        int size;
        SUserSetting users[10];
} SUserSetList;

typedef struct _userSetSetting {
          SUserSetList userList;
}SUserSetSetting;


enum ProtocolMode{
        PROTOCOL_HTTP=0,
        PROTOCOL_HTTPS,
        PROTOCOL_HTTP_HTTPS
};

/* SHTTPSetting */
typedef struct _HTTPSetting {
        int enabled;
        int port;
} SHTTPSetting;


/* SHTTPSSetting */
typedef struct _HTTPSSetting {
        int enabled;
        int port;
} SHTTPSSetting;


typedef struct _FWUPGRADE{
      char filename[64];
      int status;
} SFWUPGRADE;

typedef struct _ConfigFile{
```

```
    char filename[64];
} SConfigFile;

/* SComplexityPWDSetting */
typedef struct _SSComplexityPWDSetting {
        int pwdRule1Enabled;
        int pwdRule2Enabled;
        int pwdRule3Enabled;
}SComplexityPWDSetting;
```

6.1 addUser

ActionEvent: addUser

Request	http://<IP>/cgi-bin/users.cgi action=**add** index= username=<username> password=<password> privilege=<privilege>
Response	
Comment	
Method	POST

6.2 deleteUser

ActionEvent: deleteUser

Request	http://<IP>/cgi-bin/users.cgi action=**delete** username=<username>
Response	
Comment	
Method	POST

6.3 getUsers

ActionEvent: getUsers

Request	http://<IP>/cgi-bin/users.cgi?action=**getUsers**
Response	Size= User1.index= User1.username= User1.password= User1.privilege= … User2.username= User2.password= User2.privilege=
Comment	
Method	GET

6.4 updateUser

ActionEvent: updateUser

Request	http://<IP>/cgi-bin/users.cgi action= **update** index= username=<xxxx> password= privilege=
Response	
Comment	
Method	POST

6.5 setHTTP

ActionEvent: setHTTP

Request	http://<IP>/cgi-bin/http.cgi action= **set** enabled= port=
Response	
Comment	
Method	POST

6.6 setHTTP/HTTPS

ActionEvent: setHTTP/HTTPS

Request	http://<IP>/cgi-bin/http.cgi action= **setAll** enabled= port= httpsEnabled= httpsPort=
Response	
Comment	
Method	POST

6.7 getHTTP

ActionEvent: getHTTP

Request	http://<IP>/cgi-bin/http.cgi?action= **get**
Response	enabled= port=
Comment	
Method	GET

6.8 setHTTPS

ActionEvent: setHTTPS

Request	http://<IP>/cgi-bin/https.cgi action= **set** enabled= port=
Response	
Comment	
Method	POST

6.9 getHTTPS

ActionEvent: getHTTPS

Request	http://<IP>/cgi-bin/https.cgi?action= **get**
Response	enabled= port=
Comment	
Method	GET

6.10 resetToDefault

ActionEvent: resetToDefault

Request	http://<IP>/cgi-bin/reset.cgi?action= **reset**
Response	
Comment	Reset all settings to factory default
Method	GET

6.11 upgradeFirmware

ActionEvent: upgradeFirmware

Request	http://<IP>/cgi-bin/upgradeFirmware.cgi action= **upgrade** **Followed by the IPCam firmware**
Response	
Comment	Upgrade the system firmware upon this request
Method	POST

6.12 reboot

ActionEvent: reboot

Request	http://<IP>/cgi-bin/reboot.cgi?action= **reboot**
Response	
Comment	Reboot the system
Method	GET/POST

6.13 importConfigFile

ActionEvent: importConfigFile

Request	http://<IP>/cgi-bin/ConfigFile.cgi action= **set** filename =
Response	
Comment	
Method	POST

6.14 exportConfigFile

ActionEvent: exportConfigFile

Request	http://<IP>/cgi-bin/ConfigFile.cgi?action= **get**
Response	
Comment	
Method	get

6.15 setPWDComplexity

ActionEvent: setPWDComplexity

Request	http://<IP>/cgi-bin/complexity.cgi action= **set** pwdRule1Enabled = pwdRule2Enabled = pwdRule3Enabled =
Response	
Comment	
Method	POST

6.16 getPWDComplexity

ActionEvent: getPWDComplexity

Request	http://<IP>/cgi-bin/complexity.cgi?action= **get**
Response	pwdRule1Enabled = pwdRule2Enabled = pwdRule3Enabled =
Comment	
Method	GET

Capability API (TBD)

ActionEvents

ActionEvent	Description
getCapability	Get camera **Capability.**

7.1 getCapability

ActionEvent: getCapability

Request	http://<IP>/cgi-bin/**Capability**.cgi?action= **get**
Response	Streaming.VideoCodec.size=2 Streaming.VideoCodec1=h264 Streaming.VideoCodec2=mjpeg Streaming.name1=h264 Streaming.name1.resolution.size=3 Streaming.name1. resolutionWidth1=320 Streaming.name1. resolutionHeight1=192 Streaming.name1. resolutionWidth2=640 Streaming.name1. resolutionHeight2=400 Streaming.name1. resolutionWidth3=1280 Streaming.name1. resolutionHeight3=800 Streaming.name2=mjpeg Streaming.name2.resolution.size=3 Streaming.name2. resolutionWidth1=320 Streaming.name2. resolutionHeight1=192 Streaming.name2. resolutionWidth2=640 Streaming.name2. resolutionHeight2=400 Streaming.name2. resolutionWidth3=1280 Streaming.name2. resolutionHeight3=800 Audio.codec.size=3 Audio.codec1=PCMA Audio.codec2=PCMU Audio.codec3=G.726 Network.Type.size=2 Network.Type1=Wire Network.Type2=Wireless
Comment	
Method	GET

Motion detection API

Motion detection API allows applications to
1) set/get the motion detection setting

Data structures

Data Structure	Description
SMotionDetectionSetting	Basic motion detection setting.
SMDList	List of detection channels.
SChannelMotionDetection	Keep the information of detection channels.
SMDRegionList	List of detection regions.
SMDRegion	Keep the information of detection regions.

```
/* SMotionDetection */
// Upper left coordinte (x,y), bottom right coordinate (x1, y1)
typedef struct _MDRegionEntry {
        int enabled;
        int sensitivity;   // 1-100.   (low->high)
        int threshold;   // 1-100.   (low->high)
        int x;
        int y;
        int x1;
        int y1;
} SMDRegionEntry;


   /*SMDRegionList*/
typedef struct _MDRegionList          {
        int size;
        SMDRegionEntry regionEntry[5];
}SMDRegionList;


typedef struct _MDEntry {
        int enabled;
        int channelIndex; //match stream channel index , (Unique) 0: reserved. 1+: valid
index
        int detectionInterval;       // The time interval to carry out another MD after previous
one.
        SMDRegionList MDRList;
} SMDEntry;


typedef struct _MDList {
          int size;
          SMDEntry MDEntry[5];//match stream
}SMDList;
```

```
typedef struct _MotionDetectionSetting {
	SMDList MDList;
}SMotionDetectionSetting;
```

ActionEvents

ActionEvent	Description
setMotionDetection	Set motion detection setting
getMotionDetection	Get motion detection setting
getMotionDetections	Get all motion detections setting

8.1 setMotionDetection

ActionEvent: setMotionDetection

Request	http://<IP>/cgi-bin/motiondetection.cgi action=**set** enabled=1 channelIndex detectionInterval= region1.enabled= region1.sensitivity= region1.threshold= region1.x= region1.y= region1.x1= region1.y1= region2.enabled= region2.sensitivity= region2.threshold= region2.x= region2.y= region2.x1= region2.y1= region3.enabled= region3.sensitivity= region3.threshold= …….
Response	
Comment	
Method	POST

8.2 getMotionDetection

ActionEvent: getMotionDetection

Request	http://<IP>/cgi-bin/ motiondetection.cgi?action=**getMD&index=<index>**
Response	enabled=1 detectionInterval= region.size region1.enabled= region1.sensitivity= region1.threshold= region1.x= region1.y= region1.x1= region1.y1= region2.enabled= region2.sensitivity= region2.threshold= region2.x= region2.y= region2.x1= region2.y1= region3.enabled= region3.sensitivity= region3.threshold= ……
Comment	
Method	GET

8.3 getMotionDetections

ActionEvent: getMotionDetections

Request	http://<IP>/cgi-bin/ motiondetection.cgi?action=**get**
Response	size= MD1.enabled=1 MD1.channelIndex MD1.detectionInterval= MD1.region.size MD1.region1.enabled= MD1.region1.sensitivity= MD1.region1.threshold= MD1.region1.x= MD1.region1.y= MD1.region1.x1= MD1.region1.y1= MD1.region2.enabled= MD1.region2.sensitivity= MD1.region2.threshold= MD1.region2.x= MD1.region2.y= MD1.region2.x1= MD1.region2.y1= MD1.region3.enabled= MD1.region3.sensitivity= MD1.region3.threshold= MD1.region3.x= MD1.region3.y= MD1.region3.x1= MD1.region3.y1= ………….
Comment	
Method	GET

Event API

Event API allows applications to
1) set/get the event setting
2) set/get the notification setting

Data structures

Data Structure	**Description**
SEventPolicySetting	General setting for events.
SEventRuleSettingList	List of event rules.
SEventRuleSetting	Details the setting of each event.
SEventScheduleSettin g	Set up the schedule for triggering events
SEmailSetting	Details the setting of email.
SMailingServerList	List of email servers.
SMailingServer	Details the email servers.
SFTPSetting	Details the setting of ftp.
SFTPServerList	List of ftp servers.
SFTPServer	Details the ftp servers.
SMediaInfo	Specify the format of media.
SambaServer	Details the samba servers.

```
enum _eventScheduleType {
        EVENT_SCHEDULE_ALWAYS=0,
        EVENT_SCHEDULE_WEEKLY=1,                // TODO: TBD.
        EVENT_SCHEDULE_NEVER=2,
};

typedef struct _eventScheduleSetting {
        int    type;      /* type of schedule */
        char time[128];
/*
Weekly schedule:
Mon:0900-1700,Tue:0900-1700,Wed:0900-1700,Thu:0900-1700,Fri:0900-1700,Sat:0900
-1700,Sun:0900-1700
*/
} SEventScheduleSetting;

#define ACTION_NAME_FTP          "ftp"
#define ACTION_NAME_EMAIL        "smtp"
#define ACTION_NAME_SAMBA        "samba"

typedef struct _eventRuleSetting {
        int     index;                      //unique id
        int             enabled;
        char  name[10];
        unsigned int     eventID;                    /* type of event */
        SEventScheduleSetting sched;
```

```
	char actions[128];		/* list of references to action names separated by comma
',' */
} SEventRuleSetting;

typedef struct _eventRuleSettingList {
	int	size;
	SEventRuleSetting	rule[10];
} SEventRuleSettingList;

typedef struct _eventPolicySetting {
	SEventRuleSettingList ruleList;
} SEventPolicySetting;


enum AuthMOde{
	PLAIN=0,
	LOGIN=1,
	LOGIN_TLS=2
};


typedef struct _mailingServer {
	unsigned int authenticationMode;// => enum { PLAIN , LOGIN , TLS_LOGIN }
	unsigned int portNo; //=> 25
	unsigned char smtpServerHostName[64]; //=> smtp.gmail.com
	unsigned char accountName[64]; //=> XXXXXX
	unsigned char password[64]; //=> XXXXXX
} SMailingServer;


/* SEmailSetting */
typedef struct _emailSetting {
	unsigned char senderAddress[64]; //=> XXX@gmail.com
	unsigned char receiverAddress1[64]; //=> XXX@brickcom.com.tw // if NULL, disable
	unsigned char receiverAddress2[64]; //=> YYY@brickcom.com.tw // if NULL, disable
	unsigned char senderName[64]; //=> IPCAM
	unsigned char subject[64]; //=> "IPCAM Alert"
	unsigned int attachedVideoURLEnabled; //=> 0/1
	unsigned int attachedSnapShotEnabled; //=> 0/1
	unsigned int attachedVideoClipEnabled; //=> 0/1
	SMailingServer primary;
	SMailingServer secondary;
} SEmailSetting;


/* SFTPServer */
typedef struct _ftpServer	{
	unsigned int addressType;
	unsigned char hostname[64];
	unsigned char ipAddress[32];
	unsigned char ipv6Address[48];
	unsigned int portNo;
```

```
        unsigned char accountName[64];
        unsigned char password[64];
        unsigned int passiveModeEnabled;
} SFTPServer;


/* SFTPSetting */
typedef struct _ftpSetting   {
        unsigned int uploadSnapShotEnabled;
        unsigned int uploadVideoClipEnabled;
        SFTPServer primary;
        SFTPServer secondary;
} SFTPSetting;



/* SAlarmMediaInfo */
typedef struct _mediaInfo   {
        unsigned int snapShotEnabled;
        unsigned int videoClipEnabled;
        unsigned int preAlarmInterval;
        unsigned int postAlarmInterval;
} SAlarmMediaInfo;

enum EVENT_TYPE_DATA {
        EVENT_NONE,
        EVENT_MD,
        EVENT_IO,
        EVENT_NETWORK,
        EVENT_RESOURCE,
        EVENT_DAEMON,
};
enum NOTIFICATION_METHOD_DATA{
        NOTIFICATION_NONE,
        NOTIFICATION_FTP,
        NOTIFICATION_MAIL,
        NOTIFICATION_SAMBA,
};
enum NOTIFICATION_RECURRENCE_DATA{
        RECURRENCE_START,
        RECURRENCE_START_AND_END,
        RECURRENCE,
};

typedef struct _SambaServer {
       unsigned char HostDns[32];
       unsigned char IpAddress[32];
       unsigned char Ipv6Address[48];
       unsigned char UserName[16];
       unsigned char Password[16];
       unsigned int AddressType;
       unsigned char Preserve[12];
```

```
        unsigned char workGroup[32];
        unsigned char shareDIR[32];
} SambaServer;


/////////////////////
// Event notification      //
/////////////////////

/* Event subscription */
enum _eventTransportMode {
        EVENT_TRANSPORT_MODE_PUSH=0,
        EVENT_TRANSPORT_MODE_PULL=1,
};

/* Event transport type */
enum _eventTransportProtocol {
        EVENT_TRANSPORT_PROTOCOL_RESERVED=0,
        EVENT_TRANSPORT_PROTOCOL_UDP=1,
        EVENT_TRANSPORT_PROTOCOL_TCP=2,
        EVENT_TRANSPORT_PROTOCOL_HTTP=3,
};

enum _eventTransportDataFormat {
        EVENT_TRANSPORT_DATA_FORMAT_BINARY=0,
        EVENT_TRANSPORT_DATA_FORMAT_TEXT=1,
        EVENT_TRANSPORT_DATA_FORMAT_XML=2,
};

typedef struct _eventTransportSetting {
        int mode;               /* Binary (host byte order) or text */
        int protocol;           /* UDP, TCP, HTTP */
        int dataFormat;
        char destIPv4Address[16];
        unsigned short destPort;
} SEventTransportSetting;

typedef struct _eventSubscriptionSetting {
        unsigned int id;                /* Subscription ID (unique across system) */
        unsigned int leaseTime;         /* 0: always active, lease time in second */
                                        // TODO: How to represent time..
        SEventTransportSetting          transport;
} SEventSubscriptionSetting;

typedef struct _eventSubscriptionSettingList {
        int size;
        SEventSubscriptionSetting subscription[10];
} SEventSubscriptionSettingList;
```

ActionEvent	Description
setEventSetting	Set event setting
getEventPolicy	Get event policy
getEventRule	Get event rule
addEventSetting	Add event setting
updateEventSetting	Update event setting
removeEventSetting	Remove event setting
setEmailSetting	Set Email setting
getEmailSetting	Get Email setting
setFTPSetting	Set FTP setting
getFTPSetting	Get FTP setting
setAlarmMediaInfo	Set alarm media info
getAlarmMediaInfo	Get alarm media info
setSamba	Set samba server setting.
getSamba	Get samba server setting.

9.1 setEventSetting

ActionEvent: setEventSetting

Request	http://<IP>/cgi-bin/event.cgi action= **setEventSetting** R1index= R1enabled= R1name= R1eventID= R1sched.type= R1sched.time= R1actions= R2index=… …
Response	
Comment	
Method	POST

9.2 addEventSetting

ActionEvent: addEventSetting

Request	http://<IP>/cgi-bin/event.cgi action= **addEventSetting** index= enabled= name= eventID= sched.type= sched.time= actions=
Response	
Comment	
Method	POST

9.3 updateEventSetting

ActionEvent: updateEventSetting

Request	http://<IP>/cgi-bin/event.cgi action= **updateEventSetting** index= enabled= name= eventID= sched.type= sched.time= actions=
Response	
Comment	
Method	POST

9.4 removeEventSetting

ActionEvent: removeEventSetting

Request	http://<IP>/cgi-bin/event.cgi action= **removeEventSetting** index=
Response	
Comment	
Method	POST

9.5 getEventPolicy

ActionEvent: getEventPolicy

Request	http://<IP>/cgi-bin/event.cgi?action=**getEventPolicy**
Response	size= R1index= R1enabled= R1name= R1eventID= R1sched.type= R1sched.time= R1actions= R2index=…
Comment	
Method	GET

9.6 getEventRule

ActionEvent: getEventRule

Request	http://<IP>/cgi-bin/event.cgi?action=**getEventRule**
Response	index=0 enabled=0 name= eventID=0 sched.type=0 sched.time= actions=
Comment	
Method	GET

9.7 setEmailSetting

ActionEvent: setEmailSetting

Request	http://<IP>/cgi-bin/event.cgi action=**setEmailSetting** senderAddress= receiverAddress1= receiverAddress2= senderName= subject= attachedVideoURLEnabled= attachedSnapShotEnabled= attachedVideoClipEnabled= authenticationMode1= port1= smtpServerHostName1 accountName1= password1= authenticationMode2= port2= smtpServerHostName2= accountName2= password2=
Response	
Comment	
Method	POST

9.8 getEmailSetting

ActionEvent: getEmailSetting

Request	http://<IP>/cgi-bin/event.cgi?action=**getEmailSetting**
Response	senderAddress= receiverAddress1= receiverAddress2= senderName= subject= attachedVideoURLEnabled= attachedSnapShotEnabled= attachedVideoClipEnabled= authenticationMode1= port1= smtpServerHostName1 accountName1= password1= authenticationMode2= port2= smtpServerHostName2= accountName2= password2=
Comment	
Method	GET

9.9 setFTPSetting

ActionEvent: setFTPSetting

Request	http://<IP>/cgi-bin/event.cgi action= **setFTPSetting** uploadSnapShotEnabled= uploadVideoClipEnabled= addressType1= hostName1= ipAddress1= ipv6Address1= port1= accountName1= password1= passiveMode1= addressType2= hostName2= ipAddress2= ipv6Address2= port2= accountName2= password2= passiveMode2=
Response	
Comment	
Method	POST

9.10 getFTPSetting

ActionEvent: getFTPSetting

Request	http://<IP>/cgi-bin/event.cgi?action= **getFTPSetting**
Response	uploadSnapShotEnabled= uploadVideoClipEnabled= addressType1= hostName1= ipAddress1= ipv6Address1= port1= accountName1= password1= passiveMode1= addressType2= hostName2= ipAddress2= ipv6Address2= port2= accountName2= password2= passiveMode2=
Comment	
Method	GET

9.11 setAlarmMediaInfo

ActionEvent: setAlarmMediaInfo

Request	http://<IP>/cgi-bin/event.cgi action= **setAlarmMediaInfo** snapShotEnabled = videoClipEnabled = timeBeforeEvent= timeAfterEvent=
Response	
Comment	
Method	POST

9.12 getAlarmMediaInfo

ActionEvent: getAlarmMediaInfo

Request	http://<IP>/cgi-bin/event.cgi?action= **getAlarmMediaInfo**
Response	snapShotEnabled = videoClipEnabled = timeBeforeEvent= timeAfterEvent=
Comment	
Method	GET

9.13 setSamba

ActionEvent: setSamba

Request	http://<IP>/cgi-bin/event.cgi action= **setSamba** hostDns= IpAddress= Ipv6Address= UserName= Password= workgroup= shareDIR= addressTyep= Preserve=
Response	
Comment	
Method	POST

9.14 getSamba

ActionEvent: getSamba

Request	http://<IP>/cgi-bin/event.cgi?action= **getSamba**
Response	addressType= hostDns= ipAddress= ipv6Address= userName= password= preserve= shareDIR= workGroup=
Comment	
Method	GET

I/O Control API

I/O Control API allows applications to

1) set/get the GPIO setting

Data structures

Data Structure	Description
SGPIO	General I/O setting.

```
/*GOPI */
enum{
      GPIO_DIR_IN,
      GPIO_DIR_OUT,
      };
enum{
      GPIO_STATUS_LOW,
      GPIO_STATUS_HIGH,
      };
```

ActionEvents

ActionEvent	Description
setGPIOSetting	Set GPIO setting
getGPIOSetting	Get GPIO setting
getGPIOStatus	Get GPIO status

10.1 setGPIOSetting

ActionEvent: setGPIOSetting

Request	http://<IP>/cgi-bin/gpio.cgi
Response	
Comment	
Method	POST

10.2 getGPIOSetting

ActionEvent: getGPIOSetting

Request	http://<IP>/cgi-bin/event.cgi?action= **get**
Response	
Comment	
Method	GET

10.3 getGPIOStatus

ActionEvent: getGPIOStatus

Request	http://<IP>/cgi-bin/event.cgi?action= **getStatus**
Response	
Comment	
Method	GET

MSN API

MSN API allows applications to
1) set/get the IP Camera MSNBot setting

Data structures

Data Structure	Description
SMsnbot	Details the setting of MSNBot.
SMsnBuddyList	List of msn buddy.
MsnBuddy	Details the buddy information.

```
/*MSNbot */
typedef struct _MsnBuddy{
	int enabled;
	char account[128];			//msn account
	int    isNotifiedAcnt;		//0:no 1:yes
}MsnBuddy;


/*SMsnBuddyList */
typedef struct _MsnBuddyList	{
	int size;
	MsnBuddy buddy[5];
}SMsnBuddyList;


typedef struct _msnbotSetting{
	char account[128];
	char passwd[128];
	char msnOpPasswd[128];
	char friendlyName[128];
	int webcamEnabled;			//0:disable 1:enable
	int alarmNotifyEnabled;	//0:disable 1:enable
	SMsnBuddyList bList;

}SMsnbot;
```

ActionEvents

ActionEvent	Description
setMSNBot	Set MSNBot setting
getMSNBot	Get MSNBot setting

11.1 setMSNBot

ActionEvent: setMSNBot

Request	http://<IP>/cgi-bin/msn.cgi action=set account= passwd= msnOpPasswd= friendlyName= buddy0.enabled= buddy0.account= buddy0.isNotifiedAcnt= buddy1.enabled= buddy1.account= buddy1.isNotifiedAcnt= buddy2.enabled= buddy2.account= buddy2.isNotifiedAcnt= buddy3.enabled= buddy3.account= buddy3.isNotifiedAcnt= buddy4.enabled= buddy4.account= buddy4.isNotifiedAcnt= webcamEnabled= alarmNotifyEnabled=
Response	
Comment	
Method	POST

11.2 getMSNBot

ActionEvent: getMSNBot

Request	http://<IP>/cgi-bin/msn.cgi?action= **get**
Response	account= passwd= msnOpPasswd= friendlyName= buddy0.enabled= buddy0.account= buddy0.isNotifiedAcnt= buddy1.enabled= buddy1.account= buddy1.isNotifiedAcnt= buddy2.enabled= buddy2.account= buddy2.isNotifiedAcnt= buddy3.enabled= buddy3.account= buddy3.isNotifiedAcnt= buddy4.enabled= buddy4.account= buddy4.isNotifiedAcnt= webcamEnabled= alarmNotifyEnabled=
Comment	
Method	GET