I/O

Hanwha

SUNAPI

v2.6.2
2023-04-07

Contact Information

Hanwha Vision Co., Ltd.
Hanwha Vision 6, Pangyo-ro 319beon-gil, Bundang-gu, Seongnam-si, Gyeonggi-do, 13488, KOREA
www.hanwhavision.com

Hanwha Vision America
500 Frank W. Burr Blvd. Suite 43 Teaneck, NJ 07666
hanwhavisionamerica.com

Hanwha Vision Europe
Heriot House, Heriot Road, Chertsey, Surrey, KT16 9DT, United Kingdom
hanwhavision.eu

Hanwha Vision Middle East FZE
Jafza View 18, Office 2001-2003, Po Box 263572, Jebel Ali Free Zone, Dubai, United Arab Emirates
www.hanwhavision.com/ar

Table of Contents

Chapter 1. Overview

1.1. Description

io.cgi configures the alarm output settings. It gives manual control to the end user to perform control operations on alarm output at any time.

The following submenus are used for I/O functionalities:

- **alarmoutput**: Sets and controls the alarm output, state and duration.
- **aux**: Controls the auxiliary equipment state.
- **userinput**: Controls the on/off state of the manual trigger event.
- **alarmreset**: Resets the alarm.
- **ioport**: Sets the configurable alarm IO modes.
- **ledcontrol**: It can check current LED’s status or turn off LED.

Chapter 2. Alarm Output

2.1. Description

The **alarmoutput** submenu configures the alarm output settings.

> **NOTE**
> Attribute to check for alarm outputs support: "attributes/IO/Support/AlarmOutput"
> Attribute to check for maximum alarm outputs: "attributes/IO/Limit/MaxAlarmOutput"

Access level

Action	Camera	NVR
view	Suser	User
control	Suser	User
set	Suser	User

2.2. Syntax

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=
alarmoutput&action=<value>[&<parameter>=<value>...]
```

2.3. Parameters

Action	Parameters	Request/ Response	Type/ Value	Description
view				Reads the alarm output values.
	AlarmOutput.#.Type	RES	<enum> Alarmout, Beep	Alarm output type (read-only)
	AlarmOutput.#.IOPortIndex	RES	<int>	IO port index It shows the real index of IO port. User can match logical and real index with this parameter.. CAMERA ONLY
control	AlarmOutput.#.State	REQ	<enum> On, Off	Turns the alarm on or off. > **Note** **AlarmOutput.#.State** must be sent together with the **control** action.

Action	Parameters	Request/ Response	Type/ Value	Description
	AlarmOutput.#.ManualDuration	REQ	<enum> Always, 1s, 2s, 3s, 4s, 5s, 6s, 7s, 8s, 9s, 10s, 11s, 12s, 13s, 14s, 15s	Alarm output duration **AlarmOutput.#.ManualDuration** is only valid if **AlarmOutput.#.State** is set as On. CAMERA ONLY
set	AlarmOutput.#.IdleState	REQ, RES	<enum> NormallyOpen, NormallyClose	Alarm output state (read only for NVR)
	AlarmOutput.#.<dddh>	REQ, RES	<enum> Off, On, EventSync	Alarm output time <dddh> stands for the day of the week and time in hours. e.g. SUN1 means 1:00 AM on Sunday. and MON2 means 2:00 AM on Monday. NVR ONLY
	AlarmOutput.#.ManualDuration	REQ, RES	<enum> Always, 1s, 2s, 3s, 4s, 5s, 6s, 7s, 8s, 9s, 10s, 11s, 12s, 13s, 14s, 15s	Alarm output default duration

NOTE # represents the index number of the alarm output.

2.4. Examples

2.4.1. Getting the current alarm output settings

REQUEST

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=alarmoutput&action=view
```

CAMERA TEXT RESPONSE

```
HTTP/1.0 200 OK
```

```
Content-type: text/plain
<Body>
```

```
AlarmOutput.1.Type=Alarmout
AlarmOutput.1.IdleState=NormallyOpen
AlarmOutput.1.ManualDuration=Always
AlarmOutput.1.IOPortIndex=2
```

CAMERA JSON RESPONSE

```
HTTP/1.0 200 OK
Content-type: application/json
<Body>
```

```
{
    "AlarmOutputs": [
        {
            "AlarmOutput": 1,
            "Type": "Alarmout",
            "IdleState": "NormallyOpen",
            "ManualDuration": "Always",
            "IOPortIndex": 2
        }
    ]
}
```

NVR TEXT RESPONSE

```
HTTP/1.0 200 OK
Content-type: text/plain
<Body>
```

```
AlarmOutput.1.Type=Alarmout
AlarmOutput.1.IdleState=NormallyOpen
AlarmOutput.1.SUN0=EventSync
AlarmOutput.1.SUN1=EventSync
AlarmOutput.1.SUN2=EventSync
AlarmOutput.1.SUN3=EventSync
```

```
...
```

NVR JSON RESPONSE

```
HTTP/1.0 200 OK
Content-type: application/json
<Body>
```

```
{
    "AlarmOutputs": [
        {
            "AlarmOutput": 1,
            "Type": "Alarmout",
            "IdleState": "NormallyOpen",
            "SUN": [
                "Off",
                "Off",
                "Off",
                "Off",
                "Off",
                "Off",
                "Off",
                "Off",
                "Off",
                "On",
                "On",
                "On",
                "On",
                "On",
                "On",
                "On",
                "On",
                "EventSync",
                "EventSync",
                "EventSync",
                "EventSync",
                "EventSync",
                "EventSync",
                "EventSync"
            ],
            "MON": [
```

```
                ...
            ],
            ...
            "FRI": [...]
        }
    ]
}
```

2.4.2. Turning on Alarm Output 1 with continuous output

REQUEST

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=alarmoutput&action=control&AlarmOutput.1.State=On&AlarmOutput.1.ManualDuration=Always
```

The following request example is for NVR only.

REQUEST

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=alarmoutput&action=control&AlarmOutput.1.State=On
```

2.4.3. Setting the duration of Alarm Output 1 to 10 seconds

REQUEST

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=alarmoutput&action=set&AlarmOutput.1.IdleState=NormallyOpen&AlarmOutput.1.ManualDuration=10s
```

Chapter 3. Auxiliary Devices

3.1. Description

The **aux** submenu controls the auxiliary device on/off state.

NOTE

This chapter applies to the network cameras only.
Attribute to check for auxiliary devices support: "attributes/IO/Support/Aux"
Attribute to check for maximum auxiliary devices: "attributes/IO/Limit/MaxAux"

Access level

Action	Camera
view	Suser
control	Suser

3.2. Syntax

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=
aux&action=<value>[&<parameter>=<value>...]
```

3.3. Parameters

Action	Parameters	Request/ Response	Type/ Value	Description
view				Reads the auxiliary device settings.
	Aux.#.State	RES	<enum> On, Off	Auxiliary device state
control	Aux.#.State	REQ	<enum> On, Off	Auxiliary device state **Note** **Aux.#.State** must be sent together with the **control** action.

NOTE

represents the index number of the auxiliary device.

3.4. Examples

3.4.1. Getting the current auxiliary device settings

REQUEST

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=aux&action=view
```

TEXT RESPONSE

```
HTTP/1.0 200 OK
Content-type: text/plain
<Body>
```

```
Aux.1.State=On
Aux.2.State=Off
```

JSON RESPONSE

```
HTTP/1.0 200 OK
Content-type: application/json
<Body>
```

```
{
    "AuxDevices": [
        {
            "Aux": 1,
            "State": "On"
        },
        {
            "Aux": 2,
            "State": "Off"
        }
    ]
}
```

3.4.2. Deactivating the auxiliary device 1

REQUEST

```
http://<Device IP>/stw-
cgi/io.cgi?msubmenu=aux&action=control&Aux.1.State=Off
```

Chapter 4. User Input State

4.1. Description

The **userinput** submenu controls the User Input on/off state. When User Input is 'on', an event can be manually triggered by a user.

NOTE

This chapter applies to the network cameras only.
Attribute to check for User Input support: "attributes/Eventsource/Support/UserInput"

Access level

Action	Camera
view	Admin
control	Admin

4.2. Syntax

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=
userinput&action=<value>[&<parameter>=<value>...]
```

4.3. Parameters

Action	Parameters	Request/ Response	Type/ Value	Description
view				Reads the user input state.
	State	RES	<enum> On, Off	User input state
control	State	REQ	<enum> On, Off	User input state **Note** **State** must be sent together with the **control** action.

4.4. Examples

4.4.1. Getting the current User Input state

REQUEST

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=userinput&action=view
```

TEXT RESPONSE

```
HTTP/1.0 200 OK
Content-type: text/plain
<Body>
```

```
State=On
```

JSON RESPONSE

```
HTTP/1.0 200 OK
Content-type: application/json
<Body>
```

```
{
    "State": "On"
}
```

4.4.2. Deactivating the User Input

REQUEST

```
http://<Device IP>/stw-
cgi/io.cgi?msubmenu=userinput&action=control&State=Off
```

Chapter 5. Alarm Reset

5.1. Description

The **alarmreset** submenu resets the alarm.

> **NOTE** This chapter applies to NVR only.

Access level

Action	NVR
control	User

5.2. Syntax

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=
alarmreset&action=<value>[&<parameter>=<value>...]
```

5.3. Parameters

Action	Parameters	Request/ Response	Type/ Value	Description
control	Reset	REQ		Alarm reset No value is required for this parameter.

5.4. Examples

5.4.1. Getting the current User Input state

REQUEST

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=alarmreset&action=control
```

Chapter 6. IO Ports Configuration

6.1. Description

The **ioport** submenu configures the configurable alarm IO port.

> **NOTE**
>
> This chapter applies to network cameras only.
> Attribute to check for configurable alarm IO support:
> "attributes/IO/Support/ConfigurableIO"
> Attribute to check for maximum configurable alarm IO:
> "attributes/IO/Limit/MaxConfigurableIO"

Access level

Action	Camera
view	Admin
set	Admin

6.2. Syntax

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=
ioport&action=<value>[&<parameter>=<value>...]
```

6.3. Parameters

Action	Parameters	Request/ Response	Type/ Value	Description
view				Reads the configurable alarm IO settings
	Port	REQ	<csv>	Physical port number
set	Port.#.Mode	REQ, RES	<enum> Input, Output	Port operation mode

6.4. Examples

6.4.1. Getting the configurable alarm IO settings

REQUEST

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=ioport&action=view
```

TEXT RESPONSE

```
HTTP/1.0 200 OK
Content-type: text/plain
<Body>
```

```
Port.1.Mode=Input
Port.2.Mode=Output
```

JSON RESPONSE

```
HTTP/1.0 200 OK
Content-type: application/json
<Body>
```

```
{
    "Ports": [
        {
            "Port": 1,
            "Mode": "Input"
        },
        {
            "Port": 2,
            "Mode": "Output"
        }
    ]
}
```

6.4.2. Setting the operation mode of the configurable alarm IO port 1 to "Output"

REQUEST

```
http://<Device IP>/stw-
cgi/io.cgi?msubmenu=ioport&action=set&Port.1.Mode=Output
```

Chapter 7. LED Control

7.1. Description

The **ledcontrol** submenu controls LED Mode(turning off) and checks LED status.

Access level

Action	Camera	LEDBox
check	Admin	Admin
control	Admin	Admin

7.2. Syntax

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=
ledcontrol&action=<value>[&<parameter>=<value>...]
```

7.3. Parameters

Action	Parameters	Request/ Response	Type/ Value	Description
check				Reads the LED status.
	LED.#.Color	RES	<enum> Green, Blue, Red, Pink, SkyBlue, Purple	User input state
	LED.#.LightMode	RES	<enum> On, Off	
	LED.#.LEDPresetIndex	RES	<int>	By using "eventrules"cgi "ledpreset" submenu "control" action, you can apply LEDPresetIndex to LED. It Shows which index is being used. * 0: Not set, * #: Being used preset index
control	LightMode	REQ	<enum> Off	Control Light Mode * Off: Turn off

Action	Parameters	Request/ Response	Type/ Value	Description
	LEDIndex	REQ	<csv>	Select which LEDUsageIndex(eventrules.cgi ledpreset submenu) to apply. If hardware has 1 LED, only 1 is available.

7.4. Examples

7.4.1. Getting the current LED Status

REQUEST

```
http://<Device IP>/stw-cgi/io.cgi?msubmenu=ledcontrol&action=check
```

JSON RESPONSE

```
HTTP/1.0 200 OK
Content-type: application/json
<Body>
```

```
{
    "ledcontrol": [
        {
            "LED": 1,
            "LightMode": "Off",
            "LEDPresetIndex": 0
        },
        {
            "LED": 2,
            "LightMode": "Off",
            "LEDPresetIndex": 0
        }
    ]
}
```

7.4.2. Turn off LEDUsageIndex 1

REQUEST

```
http://<Device IP>/stw-
```

```
cgi/io.cgi?msubmenu=ledcontrol&action=control&LightMode=Off&LEDIndex=1
```

JSON RESPONSE

```
HTTP/1.0 200 OK
Content-type: application/json
<Body>
```

```
{
    "Response": "Success"
}
```