Spotbox 1.0.6

HTTP API

Table of Contents

Introduction

The Spotbox provides an API for managing the device through HTTP requests. The API methods are implemented as a set of http routes. Each method take parameters as part of the URL or as a JSON object in the body of the request.

All calls will return a JSON object with one of two properties:

```
{"result": <result object> }    // Successful result
```

or

```
{"error": "ERROR_CODE", "msg": "message"}    // Error result
```

The API is protected with a basic authorisation mechanism which requires the "/api/login" method is called before any other methods are used. On successful login a cookie is returned which must be provided for subsequent calls to be successful. The cookie will be removed by the "/api/logout" method.

POST /api/login

Login to session

Sample request

```
POST /api/login HTTP/1.1
Accept: application/json

{"user": "admin", "password": "password"}
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: text/javascript
```

Input JSON Parameters	user password	[str] username [str] password
Status Codes	200 401	Operation performed normally Not authorized

POST /api/logout

Logout of a session

Sample request

```
POST /api/logout HTTP/1.1
Accept: application/json
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: text/javascript
```

Input JSON Parameters		
Status Codes	200 401	Operation performed normally Not authorized

GET /api/view

Get the list of possible view modes.

The valid view modes are:

fullscreen – Full screen display
grid_2x2 - Grid of 2 rows and 2 colums
grid_3x3 - Grid of 3 rows and 3 colums
grid_4x4 - Grid of 4 rows and 4 colums

Sample request

GET /api/view HTTP/1.1
Accept: application/json

Sample response

HTTP/1.1 200 OK
Content-Type: application/json

{"result": ["grid_3x3", "fullscreen", "grid_2x2", "grid_4x4"]}

Output JSON Parameter	result	[str array] possible video modes
Status Codes	200 401	Operation performed normally Not authorized

GET /api/view/mode

Get the current view mode.

Sample request

```
GET /api/view/mode HTTP/1.1
Accept: application/json
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"result": "grid_2x2"}
```

Output JSON Parameters	result	[**grid_mode**] Current view mode
Status Codes	200 401	Operation performed normally Not authorized

PUT /api/view/mode

Change current view mode. Mode can be one of the valid modes returned by 'GET /api/view'

Sample request

```
PUT /api/view/mode HTTP/1.1
Accept: application/json

{"mode":"grid_2x2"}
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"result": "OK"}
```

JSON Input parameters	mode	[str] New view mode
JSON output Parameters	result	[str] OK if mode accepted
Status Codes	200 401	Operation performed normally Not authorized

GET /api/view/<view number>

Get list of cameras playing on view <view number>.

Sample request

```
GET /api/view/0 HTTP/1.1
Accept: application/json
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "result": [
    {
      "profile": {
        "name": "2_PROFILE",
        "uri": "rtsp://10.0.5.103/video1",
        "res_width": "1920",
        "codec": "H264",
        "res_height": "1080",
        "fps": "25"
      },
      "url": "rtsp://admin:1234@10.0.5.103/video1",
      "cnum": 0
    }
  ]
}
```

JSON Input parameters		
JSON output Parameters	result	**[object array]** Each element consist of: **profile [profile]** – Camera profile details **url [str]** –Full URL of rtsp stream to be played **cnum [int]** – camera index number
Status Codes	200 401	Operation performed normally Not authorized

PUT /api/view/<view number>/play

Play list of cameras on specific view <view number>.

Sample request

```
PUT /api/view/0/play HTTP/1.1
Accept: application/json

{"cameras": [0,1], "period": 120}
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"result": "OK"}
```

JSON Input parameters	cameras period	[int array] – List of camera index numbers [int] – Length of time in seconds to display each camera A period of zero will play continuously
JSON output Parameters	result	[str] – OK
Status Codes	200 401	Operation performed normally Not authorized

GET /api/cameras

Get list of camera settings

Sample request

```
GET /api/cameras HTTP/1.1
Accept: application/json
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "result": [
    {
      "username": "admin",
      "idx": 0,
      "grids": {
        "grid_4x4": null,
        "grid_3x3": null,
        "grid_2x2": null
      },
      "uri": "onvif://10.0.5.103",
      "profiles": [
        {
          "name": "2_PROFILE",
          "uri": "rtsp://10.0.5.103/video1",
          "res_width": "1920",
          "codec": "H264",
          "res_height": "1080",
          "fps": "25"
        }
      ],
      "label": "NEW",
      "period": 0,
      "lab_position": 0,
      "props": {
        "latency": 300,
        "timeout": 0
      },
      "password": "1234"
```

```
    }
  ]
}
```

JSON Input parameters		
JSON output Parameters	result	[**camera** array] Array of camera settings
Status Codes	200 401	Operation performed normally Not authorized

GET /api/camera/<idx>

Get settings for camera <idx>

Sample request

```
GET /api/camera/1 HTTP/1.1
Accept: application/json
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "result": {
    "idx": 1,
    "username": "admin",
    "password": "1234",
    "uri": "onvif://10.0.5.100",
    "label": "onvif://10.0.5.100",
    "lab_position": 0,
    "grids": {
      "grid_4x4": 4,
      "grid_3x3": 2,
      "grid_2x2": null
    },
    "period": 0,
    "profiles": [
      {
        "name": "2_PROFILE",
        "uri": "rtsp://10.0.5.100/video1",
        "res_width": "1920",
        "codec": "H264",
        "res_height": "1080",
        "fps": "25"
      },
      {
        "name": "3_PROFILE",
        "uri": "rtsp://10.0.5.100/video1s",
        "res_width": "640",
        "codec": "H264",
        "res_height": "480",
```

```
        "fps": "25"
       }
      ],
      "props": {
       "latency": 300,
      }
     }
    }
```


JSON Input parameters		
JSON output Parameters	result	[**camera**] Camera setting for camera
Status Codes	200 401	Operation performed normally Not authorized

PUT /api/camera/<cnum>

Update camera setting for camera <cnum> .

Sample request

```
GET /api/camera/1 HTTP/1.1
Accept: application/json

{
  "camera": {
    "label": "NEW",
    "username": "admin",
    "password": "password",
    "uri": "onvif://10.0.5.103",
    "profiles": [
      {
        "name": "2_PROFILE",
        "uri": "rtsp://10.0.5.103/video1",
        "codec": "H264",
        "res_width": "1920",
        "res_height": "1080",
        "fps": "25"
      }
    ]
  }
}
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"result": cnum}
```

JSON Input parameters	camera	[**camera**] – camera setting
JSON output Parameters	result	[int] - camera index for updated camera setting
Status Codes	200 401	Operation performed normally Not authorized

PUT /api/camera

Add new camera to settings

Sample request

```
GET /api/camera/1 HTTP/1.1
Accept: application/json

{
  "camera": {
    "uri": "onvif://10.0.5.103",
    "username": "admin"
    "password": "password",
    "label": "NEW",
    "profiles": [
      {
        "codec": "H264",
        "fps": "25",
        "name": "2_PROFILE",
        "res_height": "1080",
        "res_width": "1920",
        "uri": "rtsp://10.0.5.103/video1"
      }
    ]
  }
}
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"result": 6}
```

JSON Input parameters	camera	[**camera**] – camera setting
JSON output Parameters	result	[int] - camera index for new camera setting
Status Codes	200 401	Operation performed normally Not authorized

DELETE /api/camera/<idx>

Delete camera to settings

Sample request

```
DELETE /api/camera/1 HTTP/1.1
Accept: application/json
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"result": 6}
```

JSON Input parameters		
JSON output Parameters	result	[int] - Number of cameras in settings
Status Codes	200 401	Operation performed normally Not authorized

DELETE /api/cameras

Delete all camera settings

Sample request

```
DELETE /api/cameras HTTP/1.1
Accept: application/json
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"result": 0}
```

JSON Input parameters		
JSON output Parameters	result	[int] - 0 successfully deleted all cameras
Status Codes	200 401	Operation performed normally Not authorized

PUT /api/cameras/restart

Restart playing cameras in current view mode

Sample request

```
PUT /api/cameras/restart HTTP/1.1
Accept: application/json
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{"result": "OK"}
```

JSON Input parameters		
JSON output Parameters	result	[str] - OK
Status Codes	200 401	Operation performed normally Not authorized

GET /network/eth0

Get network configuration.

Sample request

```
GET /network/eth0/ HTTP/1.1
Accept: application/json
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "result": {
    "NTP": "10.0.0.1",
    "MAC": "00:16:2A:81:CA:1D",
    "Gateway": ""10.0.0.1",
    "Netmask": "255.255.0.0",
    "DNS": ""10.0.0.1",
    "Address": "10.0.1.133",
    "Method": "dhcp"
  }
}
```

JSON Input parameters		
JSON output Parameters	result	[**network_interface**] – Network interface configuration
Status Codes	200 401	Operation performed normally Not authorized

Schemas

The following provides details of the types for the Input and Output JSON parameters using JSON Schema notation:

```
{
"grid_modes": {
   "enum": ['fullscreen',
            'grid_2x2',
            'grid_3x3',
            'grid_4x4']
},

"period": {
   "type": "integer", "minimum": 0
},

"camera_idx": {
   "type": "integer",
   "minimum": 0
},

"profile": {
       "type": "object",
       "properties": {
          "uri": {"type": "string"},
          "name": {"type": "string"},
          "codec": {"type": "string"},
          "res_width": {"type": "string"},
          "res_height": {"type": "string"},
          "fps": {"type": "string"}
       },
       "required": ["uri",
                    "name",
                    "codec",
                    "res_width",
                    "res_height",
                    "fps"]
},

"grid": {
   "type": "object",
```

```
    "properties": {
  "fullscreen": {"type": ["integer", "null"]}),
  "grid_2x2": {"type": ["integer", "null"]}),
  "grid_3x3": {"type": ["integer", "null"]}),
  "grid_4x4": {"type": ["integer", "null"]})
    }
},



"props": {
    "type": "object",
    "properties": {
        "latency": {"type": "integer"},
        "timeout": {"type": "integer"},
        "drop_on_latency": {"type": "integer"},
        "debug": {"type": "integer"}
    }
},

"camera": {
    "type": "object",
    "properties": {
        "grids": grid,
        "label": {"type": "string"},
        "username": {"type": "string"},
        "password": {"type": "string"},
        "uri": {"type": "string"},
        "period": {"type": "integer"},
        "lab_position": {"type": "integer"},
        "props": { "$ref": "#/props"},
        "profiles": {
            "type": "array",
            "items": { "$ref": "#/profile" },
        },
        "idx": { "$ref": "#/camera_idx" }
    },
    "required": ["label", "uri"],
},

"network_interface": {
    "type": "object",
          "properties": {
             "Method": {"enum": ["manual", "dhcp"]},
```

```
                'Netmask': {"type": "string", "format": "ipv4"},
                'Address': {"type": "string", "format": "ipv4"},
                'Gateway': {"type": "string", "format": "ipv4"},
                'DNS': {"type": "string", "format": "ipv4"},
                'NTP': {"type": "string", "format": "ipv4"}
            },
    "required": ["Method"]
},

"network": {
    "type": "object",
    "properties": {
        "eth0": { "$ref": "#/network_interface"}
    },
},

 "view_cameras": {
    "type": "object",
    "properties": {
        "cameras": {
            "type": "array", "minitems": 0,
            "items": { "type": "integer", "minimum": 0}
        },
        "period": { "type": "integer", "minimum": 0}
    },
    "required": ["cameras"]
}
}
```

EXAMPLE PYTHON CODE

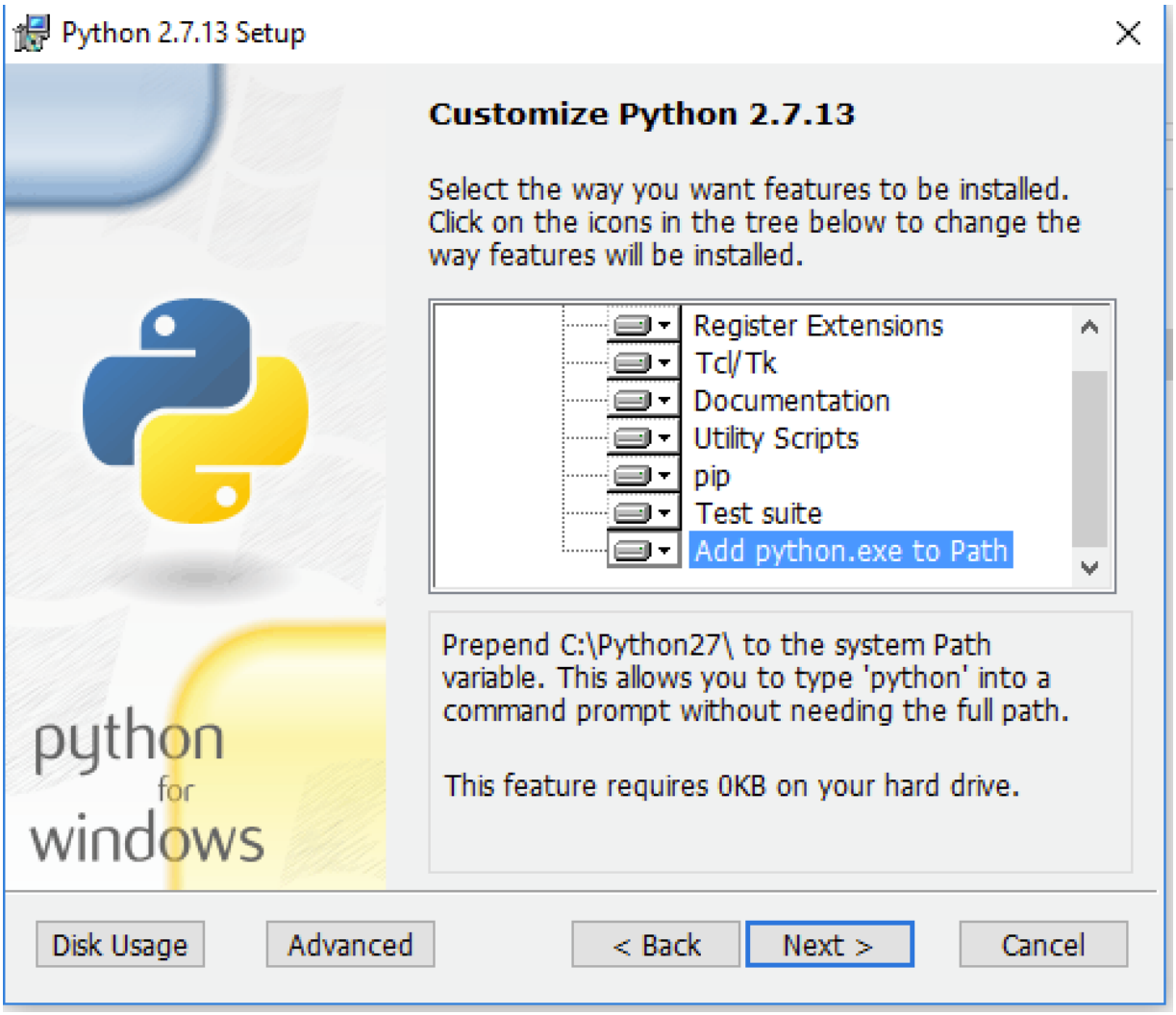
There is a simple python application "spotbox_api.py" provided to demonstrate sending API commands to the Spotbox.

Installing on Windows

The application require python and an extra python module to be installed. To install python on you host computer download and install the following file:

https://www.python.org/ftp/python/2.7.13/python-2.7.13.msi

NOTE: Make sure the option to add python.exe to Path is selected:



The open a command prompt and confirm python is installed correctly:

```
c:\spotbox>python -V
Python 2.7.13

c:\spotbox>
```

Next install the "requests" module:

```
c:\spotbox>pip install  -sprequests
Collecting requests
  Downloading requests-2.18.1-py2.py3-none-any.whl (88kB)
    100% |################################| 92kB 1.5MB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests)
  Downloading chardet-3.0.4-py2.py3-none-any.whl (133kB)
    100% |################################| 143kB 1.3MB/s
Collecting certifi>=2017.4.17 (from requests)
  Downloading certifi-2017.4.17-py2.py3-none-any.whl (375kB)
    100% |################################| 378kB 1.6MB/s
Collecting idna<2.6,>=2.5 (from requests)
  Downloading idna-2.5-py2.py3-none-any.whl (55kB)
    100% |################################| 61kB 1.3MB/s
Collecting urllib3<1.22,>=1.21.1 (from requests)
  Downloading urllib3-1.21.1-py2.py3-none-any.whl (131kB)
    100% |################################| 133kB 1.1MB/s
Installing collected packages: chardet, certifi, idna, urllib3, requests
Successfully installed certifi-2017.4.17 chardet-3.0.4 idna-2.5 requests-2.18.1 urllib3-1.21.1

c:\spotbox>
```

You should now be able to send commands to the Spotbox.

Running the command provides an interactive prompt. To send and API request type in the HTTP command ('get', 'put', 'post', 'delete') followed by the api route then the JSON parameter if required.

NOTE: Don't forget to always log into the spotbox as the first command.

For example:

```
c:\spotbox>spotbox_api.py -h 10.0.1.133
> post /api/login {"user":"admin", "password":"admin"}
request URL:
http://10.0.1.133/api/login

response:

> get /api/view/mode
request URL:
http://10.0.1.133/api/view/mode

response:
{"result": "grid_2x2"}
> put /api/view/mode {"mode":"grid_4x4"}
request URL:
http://10.0.1.133/api/view/mode

response:
{"result": "OK"}
> get /api/view/mode
request URL:
http://10.0.1.133/api/view/mode

response:
{"result": "grid_4x4"}
>
```