DiGidot

PUSHING CONTROL BOUNDARIES

API documentation

TABLE OF CONTENT

Introduction

This documentation is about the API of the DiGidot C4. The DiGidot C4 is a compact device that is capable of converting Art-Net to SPI signals. There are two versions of the DiGidot C4. The ‘live’ version is the best option if you would like to control a show live with a computer that can send Art-Net. The DiGidot C4 Extended can work as a standalone device without the necessity for an additional computer. You can record effects, playback scenes and use a wide range of input triggers to execute actions.

With the Api you can communicate to the DiGidot C4 device to create, read, update or delete data. This can be helpful if you want to create your own application for the DiGidot C4.

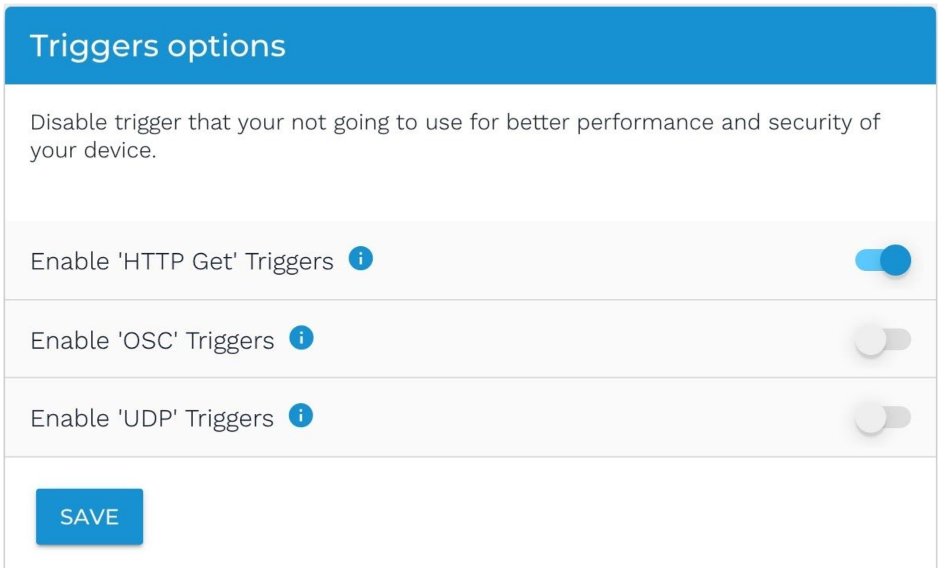
How to connect to the API

There are two ways to connect to the API. It is possible to use HTTP post or HTTP get. For simple applications we recommend using HTTP get. For HTTP get it's required to enable the HTTP Get option in the interface. If you want to use all the available API calls, we recommend using the HTTP post.

HTTP get

To enable the HTTP get API use the following steps:

1. Load the DiGidot C4 interface by going to the IP address of the DiGidot C4 or use the global IP address of http://10.254.254.254.
2. Click on the 'Triggers' button.
3. Click on the three dots at the top right and then "HTTP, UDP and OSC".
4. Enable the HTTP Get option.
5. Click the 'Save' button


Triggers options

Disable trigger that your not going to use for better performance and security of your device.

Enable 'HTTP Get' Triggers [✓ on]

Enable 'OSC' Triggers [off]

Enable 'UDP' Triggers [off]

After enabling the HTTP get option, it is possible to send HTTP get requests to the DiGidot C4. In the next chapter 'API calls', we show some examples of HTTP get requests.

HTTP post

Using the HTTP post method to connect to the API of the DiGidot C4 needs some preparations. Follow the steps below to get started:

1. Create a valid HTTP post request
2. Get nonce from DiGidot C4 for communication
3. Send requests
4. Send requests with specific user account

Create a valid HTTP post request

A proper authenticated request contains a SHA-1 of the request and the request. If accounts is configurted a object auth object and SHA-256 HMAC authentication is also required.

Request definition

A request contains three parts, a SHA-1 of the object, the request message and a SHA-256 HMAC. The SHA-256 is only needed when having an account created.

Structure:

```
<SHA1><JSON MESSAGE><SHA256-HMAC>
```

Example request:

```
834d715104354a842419b6b550dabaf35ffc43d5{
        "action":"status",
        “subaction”: [“Time”,"settings", "scene_manager","analog",
"performance"],
        "auth":{
                "nonce" : "d43821556a65430ebb9aa613c16de55fabc31e31",
                "user" : "admin”,
                "sessionid" : 865992
        }
}1edeaa8b1ca3c2ae251cec8642ec482b4cc648b9d7d56c5da7626aab1e65bc0c:
```

SHA1: A hash that is used to verify the integrity of the message. This will be placed at the front of the JSON message.
SHA256: Keyed-hash message authentication hash (SHA-256) of the json message and password as the key. This is only needed when working with accounts. The SHA-256 will be placed after the JSON message.

Example code

Below are some links of examples on how to connect to the API with different code languages.

Javascript/Jquery:

https://digidot.eu/example-code/javascript/jquery_api_example.zip

Python 3:

https://digidot.eu/example-code/python/python_api_example.py

API calls

The API calls examples in this chapter will not include the SHA-1, the auth object and the SHA-256 HMAC, but note this will still be required. This way the examples will stay cleaner and don't repeat every time.

Please note: all values in the requests are examples and need to be replaced by device- and user specific values.

Device settings

Get status

This will get the current status of the DiGidot C4.

Message:

```
{
        "action": "status",
        "get": ["time", "master_fader", "license", "settings", "wifi", "scene_manager", "mappings",
         "analog", "performance"]
}
```

Response:

```
{
        "time": {

                "sec": 34,
                "min": 25,
                "hour": 15,
                "dom": 9,
                "dow": 6,
                "month": 1,
                "year": 2020,
                "dst": 1
        },
        "master_fader": 65535,
        "analog": {
                "0": {
                        "value": 3097,
                        "minimum": 0,
                        "maximum": 4095
                },
                    "1": {
                              "value": 3161,
                              "minimum": 0,
                              "maximum": 4095
                    },
                    "2": {
                              "value": 3171,
                              "minimum": 0,
                              "maximum": 4095
                    }
          },
          "button": 0,
          "performance": {
                    "memory_used": 7984,
                    "memory_total": 46960,
                    "artnet_fps": 16,
                    "artnet_operations": 240,
                    "scene_fps": 24,
                    "scene_operations": 120
          }
}
```

Get settings

Get the status LED, device and groups settings.

Message:

```
{
          "action": "settings",
          "subaction": "get"
}
```

Response:

```
{

          "result": "ok",
          "settings": {
                    "version": 253,
                    "name": "DiGidot C4",
                    "status_leds": true,
                    "ethernet_leds": true,
                    "listen_group": [
                              {
                                        "unique_id": 2,
                                        "name": "group"
                              }
                    ],
                    "order": 0,
                    "udp_triggering": false,
                    "http_triggering": false,
                    "osc_triggering": false,
                    "sd_trigger_logging": false
          }

}
```

Set settings

Set the status LED, device and groups settings.

Message:

```
{
        "action": "settings",
        "subaction": "set",
        "name": "DiGidot C4",
        "status_leds": true,
        "ethernet_leds": true,
        "listen_group": [],
        "order": 0,
        "udp_triggering": true,
        "http_triggering": true,
        "osc_triggering": true
}
```

Response:

```
{

        "result": "ok",
        "Settings": {
                "action": "settings",
                "subaction": "set",
                "name": "DiGidot C4",
                "status_leds": true,
                "ethernet_leds": true,
                "listen_group": [],
                "order": 0,
                "udp_triggering": true,
                "http_triggering": true,
                "osc_triggering": true
        }

}
```

Get Time

Get the time of a device.

Message:

```
{
	"action": "time"
}
```

Response:

```
{
	"result": "ok",
	"time": {
		"sec": 0,
		"min": 42,
		"hour": 17,
		"dom": 9,
		"dow": 6,
		"month": 1,
		"year": 2020,
		"dst": 1
	}
}
```

Set Time

Set the time of a device.

Message:

```
{
	"action": "time",
	"sec": 0,
	"min": 42,
	"hour": 17,
	"dom": 9,
	"dow": 6,
	"month": 1,
	"year": 2020,
	"dst": 1
}
```

Response:

```
{
	"time": {
		"sec": 45,
                "min": 34,
                "hour": 12,
                "dom": 6,
                "dow": 3,
                "month": 1,
                "year": 2020,
                "dst": 1
        },
        "result": "ok"
}
```

Discover devices

The device will perform a “discover” action in the network in order to find other DiGidot C4 devices. If a new request with parameter “clear_discovered”: ‘false’ needs to be sent to get the other devices. This can take up multiple requests to get all the devices in the network.

Message:

```
{

        "action": "discover",
        "clear_discovered": true,
        "timeout": 1200,
}
```

Response:

```
{
        "devices": [
        {
                "result": "ok",
                "mac_address": "D0:76:50:A1:05:50",
                "name": "DiGidot C4",
                "version": 1577724899,
                "fw_date": "Dec 30 2019 17:53:42",
                "fw_type": "production",
                "fw_version": "2.1.2",
                "order": 0,
                "ip_address[0]": "10.0.0.2",
                "ip_address[1]": "0.0.0.0",
                "ip_address[2]": "0.0.0.0",
                "ip_address[3]": "0.0.0.0",
                "revision": 2,
                "has_io_config_file": 0,
                "master_fader": 65535
        }
        ],
        "result": "ok”
}
```

Get master fader (brightness)

Get the master brightness of a device. This is a 16 bit value (0-65535).

Message:

```
{
        "action": "fader",
        "type": "master_fader"
}
```

Response:

```
{
        "master_fader": {
                "value": 65535
        },
        "result": "ok"
}
```

Set master fader (brightness)

Set the master brightness of the device. This is a 16 bit value (0-65535).

Message:

```
{
        "action": "fader",
        "type": "master_fader",
        "value": 65535
}
```

Response:

```
{
        "master_fader": {
                "value": 65535
        },
        "result": "ok”
}
```

Reset device

Reboot a device. This can be needed when adding new triggers. The triggers will only be loaded at the startup.

Message:

```
{
        "action": "reset"
}
```

Response:

```
{
        "result": "ok"
}
```

Input / output configuration

The input and output configuration contains inputs, mappings and outputs. The mapping connects inputs and outputs to each other. This way you can easily re-use inputs.

Get IO config unique ID's

Get all the unique ID's of the IO config items. This is necessary to get detailed information from an IO config item.

Message:
```
{
        "action": "io_manager",
        "type": "io_config"
}
```

Response:

```
{
        "unique_id": [
                1001,
                1002,
                1003,
                3001,
                3002,
                3003,
                4001,
                4002,
                4003
                ],

        "result": "ok"
}
```

Get IO config item

Get detailed information of an IO config item.

Message:

```
{
        "action": "io_manager",
        "type": "io_config",
        "unique_id": [
                1001,
                1002
```

```
	]
}
```

Response:

```
{
	"unique_id": {
		"1001": {
			"address": 8192,
			"io_config": {
				"type": "artnet_input",
				"unique_id": 1001,
				"flags": 0,
				"port": "network",
				"channel": 512,
				"timeout": 0,
				"universe": 0,
				"ip_address": "0.0.0.0"

			},
			"has_io_config_file": 0
		},
		"1002": {
			"address": 8960,
			"io_config": {
				"type": "artnet_input",
				"unique_id": 1002,
				"flags": 0,
				"port": "network",
				"channel": 512,
				"timeout": 0,
				"universe": 1,
				"ip_address": "0.0.0.0"
			},
			"has_io_config_file": 0
		}
	},
	"result": "ok"
}
```

Create IO config item

Create a new IO config item. This can be an input, mapping or output item.

Message:

```
{
	"action":"io_manager",
	"subaction":"add",
        "type":"io_config",
        "io_config":{
                "type": "artnet_input",
                "unique_id": 1002,
                "flags": 0,
                "port": "network",
                "channel": 512,
                "timeout": 0,
                "universe": 1,
                "ip_address": "0.0.0.0"
        }
}
```

Response:

```
{
"io_config": {
        "type": "artnet_input",
        "unique_id": 1002,
        "flags": 0,
        "port": "network",
        "channel": 512,
        "timeout": 0,
        "universe": 1,
        "ip_address": "0.0.0.0"
        },
        "result": "ok"
}
```

Clear IO config

Clear certain IO config items by unique ID of the config item.

Message:

```
{
        "action": "io_manager",
        "type": "io_config",
        "subaction": "remove",
        "unique_id": []
}
```

Response:

```
{
        "result": "ok"
}
```

Art-Net listener

Get the signal information of an input item. De data is returned in a base64. To get the RGB(W) value, decode the ‘data’ value. Last_revew gives you information about how long ago the signal changed in milliseconds.

Message:

```
{
        "action": "input",
        "unique_id": 1002
}
```

Response:

```
{
        "total_length": 1,
        "unique_id": {
                "1002": {
                        "last_renew": 11799,
                        "data":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=",
                "locked": false
                }
        },
        "result": "ok"

}
```

Ethernet and Wi-Fi settings

Get ethernet settings

Get the information about the ethernet settings.

Message:

```
{
        "action": "network",
        "subaction": "settings",
        "type": "ethernet"
}
```

Response:

```
{
        "settings": {
                "mac_address": "D0:76:50:A1:05:50",
                "ip_address[0]": "10.0.0.2",
                "ip_address[1]": "0.0.0.0",
                "ip_address[2]": "0.0.0.0",
                "ip_address[3]": "0.0.0.0",
                "netmask": "255.0.0.0",
                "gateway": "0.0.0.0",
                "dhcpserver": true
        },
        "result": "ok"
}
```

Set ethernet settings

Set certain ethernet settings.

Message:

```
{
        "action": "network",
        "subaction": "settings",
        "type": "ethernet",
        "item": {
                "dhcp_server": true,
                "ip_address[0]": "10.0.0.2",
                "ip_address[1]": "0.0.0.0",
                "ip_address[2]": "0.0.0.0",
		"ip_address[3]": "0.0.0.0",
		"netmask": "255.0.0.0",
		"gateway": "0.0.0.0"
	}
}
```

Response:

```
{
	"settings": {
		"mac_address": "D0:76:50:A1:05:50",
		"ip_address[0]": "10.0.0.2",
		"ip_address[1]": "0.0.0.0",
		"ip_address[2]": "0.0.0.0",
		"ip_address[3]": "0.0.0.0",
		"netmask": "255.0.0.0",
		"gateway": "0.0.0.0",
		"dhcpserver": true
	},
	"result": "ok"
}
```

Get Wi-Fi settings

Get the Wi-Fi settings information.

Message:

```
{
	"action": "network",
	"subaction": "settings",
	"type": "wifi"
}
```

Response:

```
{
	"settings": {
		"enabled": true,
		"mode": "client",
		"ip_mode": "dhcp",
		"dhcp_server": true,
		"use_bssid": false,
		"ssid_hidden": false,
		"ip_address[0]": "0.0.0.0",
		"netmask": "255.255.0.0",
		"gateway": "172.16.0.1",
		"auto_channel": false,
		"channel": 11,
                "bssid": "00:00:00:00:00:00",
                "ssid": "wifipoint",
                "security": "open",
                "dhcp_start": "172.16.0.2",
                "dhcp_end": "172.16.0.50"
        },
        "result": "ok"
}
```

Set Wi-Fi settings

Set certain Wi-Fi settings.

Message:

```
{
        "action": "network",
        "subaction": "settings",
        "type": "wifi",
        "wifi_settings": {
                "enabled": true,
                "mode": "ap",
                "dhcp_server": false,
                "useb_ssid": false,
                "ssid_hidden": false,
                "ip_address[0]": "172.16.0.1",
                "ip_address[1]": "0.0.0.0",
                "ip_address[2]": "0.0.0.0",
                "ip_address[3]": "0.0.0.0",
                "netmask": "255.0.0.0",
                "gateway": "0.0.0.0",
                "channel": 11,
                "bssid": "f4:ec:38:dd:69:3c",
                "ssid": "digidot network",
                "password": "123456789",
                "security": "wpa",
                "dhcp_start": "172.16.0.2",
                "dhcp_end": "172.16.0.50"
        }

}
```

Response:

```
{
        "settings": {
                "enabled": true,
                "mode": "client",
                "ip_mode": "dhcp",
                "dhcp_server": true,
                "use_bssid": false,
                "ssid_hidden": false,
                "ip_address[0]": "0.0.0.0",
                "netmask": "255.255.0.0",
                "gateway": "172.16.0.1",
                "auto_channel": true,
                "channel": 11,
                "bssid": "00:00:00:00:00:00",
                "ssid": "wifi point",
                "security": "open",
                "dhcp_start": "172.16.0.2",
                "dhcp_end": "172.16.0.50"
        },
        "result": "ok"
}
```

Get Wi-Fi state

Get the information about the current Wi-Fi state. If the device is connected to another Wi-Fi point or if the device is used as an access point.

Message:

```
{
        "action": "network",
        "type": "wifi",
        "subaction": "state"
}
```

Response:

```
{
        "enabled": true,
        "state": "idle",
        "mode": "client",
        "connected": true,
        "network": {
                "rssi": -85,
                "dhcp_info": {
                        "lwip_status": 10,
                        "status": "successful",
                        "ip_address": "192.168.1.220",
                        "netmask": "255.255.255.0",
                        "gateway": "192.168.1.1"
                }
        },
        "result": "ok"
}
```

Request Wi-Fi network scan

Give the device a task to do a Wi-Fi network scan. The results of this scan can be obtained by the ‘Get scan result’ command.

Message:

```
{
        "action": "network",
        "type": "wifi",
        "subaction": "scan"
}
```

Response:

```
{
        "state": "scan requested",
        "result": "ok"
}
```

Get Wi-Fi network scan results

Get the available Wi-Fi networks of the Wi-Fi scan. If more then five networks are returned you need to do another request with the index parameter + 1 to get the other results.

Message:

```
{
        "action": "network",
        "type": "wifi",
        "subaction": "scan_result",
        "index": 0
}
```

Response:

```
{
        "state": "idle",
        "network": [
                {
                        "bssid": "d0:76:50:a9:0b:cc",
                        "rssi": -70,

                        "security": "open",
                        "ssid": "Washers"
                },
                {
                        "bssid": "1c:3a:de:fd:a3:63",
                        "rssi": -83,
                        "security": "wpa2",
                        "ssid": "Fallscreenshore"
                },
                {
                        "bssid": "86:a9:3e:52:e3:e1",
                        "rssi": -76,
                        "security": "wpa2",
                        "ssid": "DIRECT-e1-HP M15 LaserJet"
                },
                {
                        "bssid": "78:8a:20:d1:8c:ef",
                        "rssi": -83,
                        "security": "wpa2",
                        "ssid": "InventDesign Guest"
                },
                {
                        "bssid": "d0:76:50:a9:14:de",
                        "rssi": -89,
                        "security": "open",
                        "ssid": "DiGidot C4 - 14:DE"
                }
        ],
        "size": 5,
        "result": "ok"
}
```

Wi-Fi check client connection details

Try to connect to a network with the given Wi-Fi credentials. The result of this request can be obtained by the Wi-Fi result client connection details' command.

Message:

```
{
    "action": "network",
    "type": "wifi",
    "subaction": "client_connection",
    "mode": "check",
    "ssid": "inventdesign",
    "password": "password"
}
```

Response:

Coming soon

Wi-Fi result client connection details

Get the result of the 'Wi-Fi check client connect details.' command.

Message:

```
{
        "action": "network",
        "type": "wifi",
        "subaction": "client_connection",
        "mode": "result"
}
```

Response:

```
{
        "successful": true,
        "result": "ok",
}
```

microSD card

Get info

Get information about the SD card.

Message:

```
{
	"action": "sd_card",
	"subaction": "lsdir"
}
```

Response:

```
{
	"free": 122870,
	"csize": 64,
	"size": 512,
	"sectors": 15802368,
	"block_size": 1,
	"media_type": 3,
	"result": "ok",
}
```

Get directory

Get the files a certain directory of the SD card.

Message:

```
{
	"action": "sd_card",
	"subaction": "ls",
	"dir": "."
}
```

Response:

```
{

	"files": [
		{
```

```
            "info": 16,
            "date": 20332,
            "time": 27747,
            "size": 0,
            "name": "FIRMWARE"
    },
    {
            "info": 16,
            "date": 20332,
            "time": 27747,
            "size": 0,
            "name": "SCENES"
    },
    {
            "info": 16,
            "date": 20332,
            "time": 27748,
            "size": 0,
            "name": "HTTP"
    },
    {
            "info": 16,
            "date": 20332,
            "time": 27748,
            "size": 0,
            "name": "SYSTEM"
    },
    {
            "info": 16,
            "date": 20332,
            "time": 27748,
            "size": 0,
            "name": "TMP"
    },
    {
            "info": 16,
            "date": 20332,
            "time": 27748,
            "size": 0,
            "name": "STATICIN"
    },
    {
            "info": 16,
            "date": 20332,
            "time": 27748,
            "size": 0,
            "name": "ZONES"
    },
    {
```

```
                    "info": 32,
                    "date": 20343,
                    "time": 32023,
                    "size": 83,
                    "name": "C-4"
                },
                {
                    "info": 16,
                    "date": 20332,
                    "time": 32504,
                    "size": 0,
                    "name": "TRASH"
                },
                {
                    "info": 32,
                    "date": 20535,
                    "time": 31212,
                    "size": 10000026,
                    "name": "LOG0.TXT"
                },
                {
                    "info": 32,
                    "date": 20518,
                    "time": 28230,
                    "size": 7723,
                    "name": "LOG1.TXT"
                }
        ],
        "result": "ok"
}
```

Write to microSD card

Write a file to the SD card. If the file is bigger than 1kb it’s required to send in multiple requests. The data value should be sent base64 encoded.

Message:

```
{
        "action": "sd_card",
        "subaction": "nano",
        "filename": "SYSTEM/TRIGGER/123.JSN",
        "create": true,
        "seek": 0,
        "data": "[base64-data]",
        "length": 1024
}
```

Response:

```
{
        Result: "ok"
}
```

Read microSD card

Read a file form the SD card. If the file is bigger than 2kb it's required to send multiple requests to get the whole file. To get the readable file you have to decode the base64 data from the response.

Message:

```
{
        "action": "sd_card",
        "subaction": "cat",
        "filename": "./SYSTEM/TRIGGERS/2754ACD.JSN",
        "seek": 0,
        "length": 2048
}
```

Response:

```
{
"data":
"eyJ1bmlxdWVfaWQiOjQxMjQxMjkzLCJncm91cCI6NDEyNDEyOTMsIm5hbWUiOiJhZXJiZXJiIiwidHlwZSI6MS
wiYWN0aXZlIjp0cnVlLCJyZXRyaWdnZXIiOjY1NTM1LCJvZmZzZXQiOjIwMDAsImxlbmd0aCI6MjIwMCwiZXZlbn
QiOjEsImR1cmF0aW9uIjoxNSwiaW5kZXgiOjAsImFuYWxvZ190eXBlIjoiYnV0dG9uIn0=",
        "length": 182,
        "result": "ok"
}
```

Format the SD card

This will clear all the files and will create a basic DiGidot C4 folder structure. By formatting the SD card the scenes and triggers will be lost.

Message:

```
{
        "action":"sd_card",
        "subaction":"format"
}
```

Response:

```
{
        "result": "ok"
}
```

Scenes/playlist

Get scenes

Get a list of scenes from a device. If five scenes are returned another request needs to be sent with the offset set to 5.

Message:

```
{
	"action": "scene_manager",
	"subaction": "list",
	"length": 5,
	"offset": 0
}
```

Response:

```
{
	"list": [
		{
			"filename": "5DCBDA6F.SCN",
			"size": 5632,
			"name": "Falling drops",
			"order": 0,
			"time": 1125
		},
		{
			"filename": "5DF8EE91.SCN",
			"size": 9216,
			"name": "rood",
			"order": 0,
			"time": 1000
		},
		{
			"filename": "5E0E19DE.SCN",
			"size": 9216,
			"name": "white",
			"time": 1000
		},
		{
			"filename": "5E0EF7C1.SCN",
			"size": 9216,
			"name": "4 channels",
			"time": 1000
```

```
                },
                {
                        "filename": "5E0EF963.SCN",
                        "size": 9216,
                        "name": "testje",
                        "time": 1000
                }
        ],
        "result": "ok"
}
```

Current playing information

Get information about the current scene or playlist that is active.

Message:

```
{
        "action": "scene_manager",
        "subaction": "state"
}
```

Response:

```
{
        "state": 1,
        "playlist": [],
        "scene": [
                {
                        "frame": {
                                "current": 2,
                                "total": 4
                        },
                        "unique_id": 1578039233,
                        "priority": 2,
                        "alpha": 65535,
                        "speed": 100,
                        "paused": false
                }
        ],
        "result": "ok"
}
```

Play scene

Play a scene with certain parameters.

Message:

```
{
        "action": "playback",
        "subaction": "play",
        "type": "scene",
        "fade": {
                "in": {
                        "fade": 100,
                        "delay": 0,
                        "curve": 1
                },
                "out": {
                        "fade": 100,
                        "delay": 0,
                        "curve": 1
                }
        },
        "unique_id": 1578039233,
        "alpha": 65535,
        "priority": 2
}
```

Response:

```
{
        "result": "ok"
}
```

Play playlist

Play a playlist.

Message:

```
{
        "action": "playback",
        "subaction": "play",
        "type": "playlist",
        "start_index": 0,
        "unique_id": 42004932,
        "alpha": 65535,
        "priority": 2
}
```

Response:

```
{
        "result": "ok"
}
```

Stop scene or playlist

Stop a scene or playlist.

Message:

```
{
        "action": "playback",
        "subaction": "stop"
}
```

Response:

```
{
        "result": "ok"
}
```

Delete scene

Delete a scene.

Message:

```
{
        "action":"sd_card",
        "subaction":"rm",
        "filename":"SCENES/5E0F0058.SCN"
}
```

Response:

```
{
        "result": "ok"
}
```

HSV

Get HSV

Get the current HSV value of a device.

Message:

Response:

Set HSV

Set the HSV value of a device.

Message:

```
{
        "action": "hsv",
        "subaction": "offset",
        "hue": 0,
        "saturation": 0,
        "value": 0
}
```

Response:

```
{

        "unique_id": {}
        "result": "ok"
}
```

Speed

Get speed

Get the current speed of the scene.

Message:

```
{
	"action": "scene_manager",
	"subaction": "state"
}
```

Response:

```
{
	"state": 1,
	"playlist": [],
	"scene": [
		{
			"frame": {
				"current": 2,
				"total": 4
			},
			"unique_id": 1578039233,
			"priority": 2,
			"alpha": 65535,
			"speed": 100,
			"paused": false
		}
	],
	"result": "ok"
}
```

Set speed

Set the speed of the current scene. Speed value is in percentages. The transition duration is the time it takes to go to the given speed in milliseconds.

Message:

```
{
	"action": "playback",
	"subaction": "speed",
	"speed": {
		"value": 161,
                "transition_duration": 1000
        }
}
```

Response:

```
{
        "speed": {
                "value": 161,
                "transition": true
        },
        "result": "ok"
}
```

Accounts

Authenticate

Check if the user can login to the device.

Message:

```
{

        "action": "authenticate",
        "subaction": "check",
        "auth":{

                "nonce" : "d43821556a65430ebb9aa613c16de55fabc31e31",
                "user" : "admin”,
                "sessionid" : 865992
          }
}
```

Response:

```
{
        "authentication_level": "Admin",
        "result": "ok",
}
```

Get accounts

Get the current account of a device.

Message:

```
{
        "action": "authenticate",
        "subaction": "list"
}
```

Response:

```
{
        "user": [
                {
                        "name": "admin",
                        "username": "admin",
                        "authentication_level": "Admin"
```

```
                    }
          ],
          "result": "ok",
}
```

Triggers

Execute a trigger

This will execute the action of a trigger.

Message:

```
{
	"action": "trigger",
	"subaction": "fire",
	"unique_id": [
		23352342
	]
}
```

Response:

```
{
	"result": "ok"
}
```

Activate triggers

This will enable a trigger. This option will not be stored after restart.

Message:

```
{
	"action": "trigger",
	"subaction": "active",
	"unique_id": [
		82551268
	],
	"active": false
}
```

Response:

```
{
	"trigger": [
		{
			"active": false,
			"unique_id": 82551268
		}
```

```
        ],
        "result": "ok"
}
```

Deactivate triggers

This will disable a trigger. This option will not be stored after restart.

Message:

```
{
        "action": "trigger",
        "subaction": "active",
        "unique_id": [
                82551268
        ],
        "active": true
}
```

Response:

```
{
        "trigger": [
                {
                        "active": true,
                        "unique_id": 82551268
                }
        ],
        "result": "ok"
}
```

License

Get license

Get the license of a device.

Message:

```
{
        "action": "license",
        "subaction": "get"
}
```

Response:

```
{
        "sd_card_enabled": 1,
        "wifi_allowed": 1,
        "max_artnet_channels": 16384,
        "serial_number": "C4--20170803-4AFAE-9288B",
        "result": "ok"
}
```