CHROMATEQ

Communication Development Kit
User Manual (V2.1)
2020/07

Chromateq UDP JSON Protocol

Overview

This document describes how to drive the Chromateq softwares or Chromateq CQSA-E 1024 Ethernet Interfaces from external software using either mobile application (Android, IOS, …) or desktop application with common programming languages (C++, Java …). The following describes how to communicate with Chromateq software/interface server (smartphone server).

Third party software can establish **UDP** communication with the Chromateq software/interface and drive it using pre-defined JSON commands. JSON is a popular well supported format that makes it easy to communicate with Chromateq software/interface from any programming language and platform.

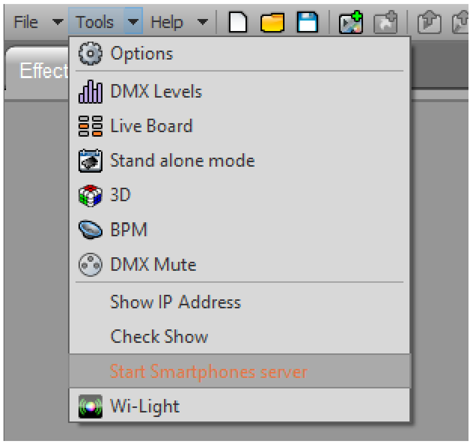
Then you will be able to On/Off buttons of the software, interface scenes and use commands by the Ethernet network.

Protocol

The UDP protocol contains different messages that allow interacting with Scenes, Triggers, Commands, Colors and DMX channels.

<u>Messages are **JSON formatted** and encoded using **UTF-8** to support internationalization.</u>

1. You must send a **"Get Serial"** or **"Get SW Serial "** message after opening the **UDP** port of your application to establish the communication with Chromateq software/interface server (**port 8093 to write or send JSON command messages / 8094 to read or receive command messages).** The ports used to drive Chromateq Ethernet Interfaces are different than the softwares ones (**port 8011 to write or send JSON command messages / 8012 to read or receive command messages**). You will open the UDP port of the software when you turn On the smartphone server with the "Start Smartphone server" option from the tools menu. Interface ports are always open.



2. After receive the serial number of the software or the interface, you have to send an **“Get State”** initialization message (type = **”Get State”**, Highly recommended to know the state of your scene, see what is playing, record the software and make sure you are communicating with the right software).

3. The Chromateq software/interface will return a “State” message (type = **”State”**) that contains the current state of the software/interface and should be used to mimic the state of the software/interface in your application.

If the communication is accepted, you software can drive Chromateq software/interface using the list of messages described in page 5 and 6.

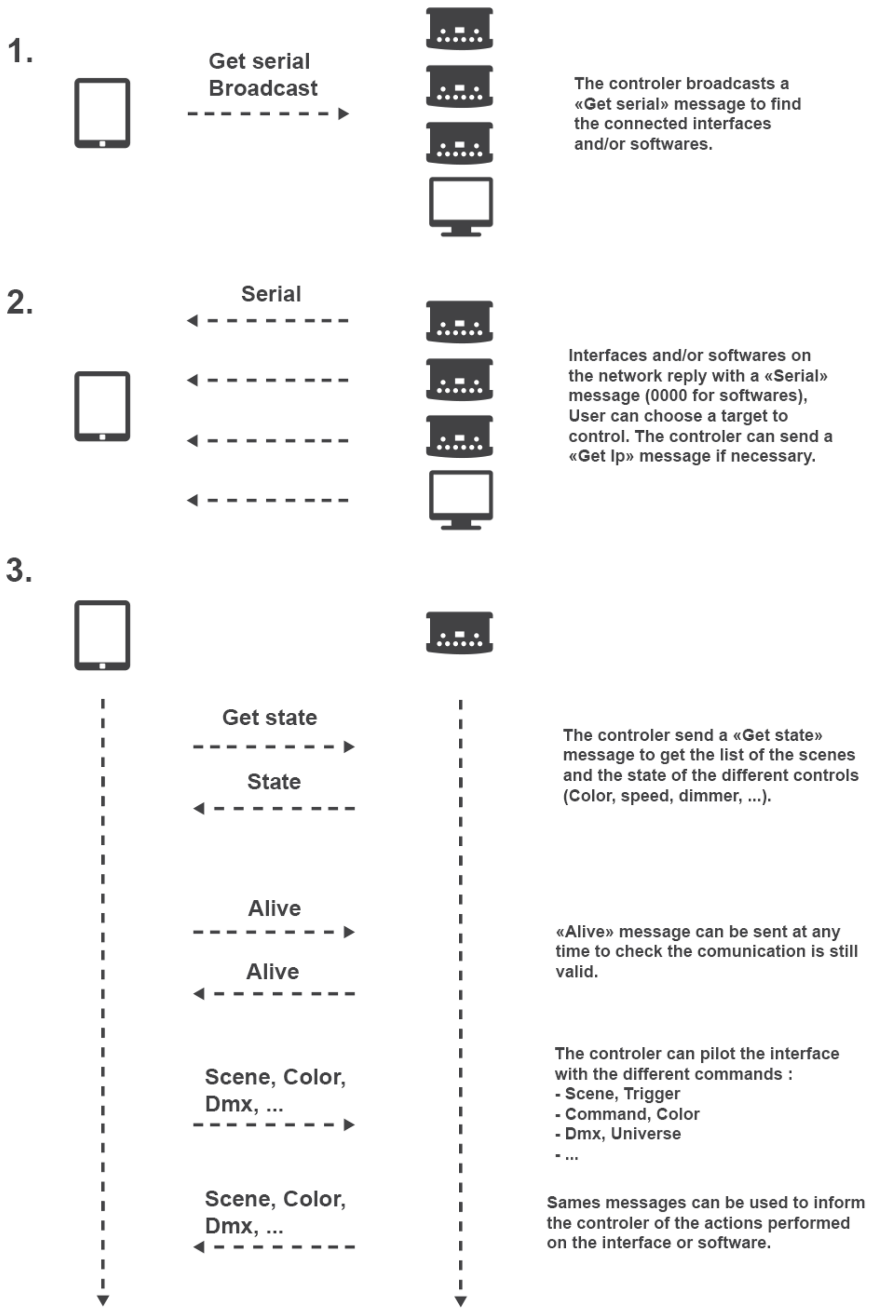
List of the possible serial software numbers received :
Pro DMX, LED Player, PIXXEM = 0000
CQSA-E 1024 = 5 digits

Software ports :
- 8093 to write or send JSON command messages
- 8094 to read or receive command messages

CQSA-E interface ports :
- 8011 to write or send JSON command messages
- 8012 to read or receive command messages

Connexion Lifecycle

Multiple connections can be established with the Chromateq softwares. When the number max of users is reached, a “limit” message is sent instead of “state” message in response at a “getstate” message and your software can no longer drive the Chromateq software.
Here is a picture showing the expecting behavior :



The controler broadcasts a «Get serial» message to find the connected interfaces and/or softwares.

Interfaces and/or softwares on the network reply with a «Serial» message (0000 for softwares), User can choose a target to control. The controler can send a «Get Ip» message if necessary.

The controler send a «Get state» message to get the list of the scenes and the state of the different controls (Color, speed, dimmer, ...).

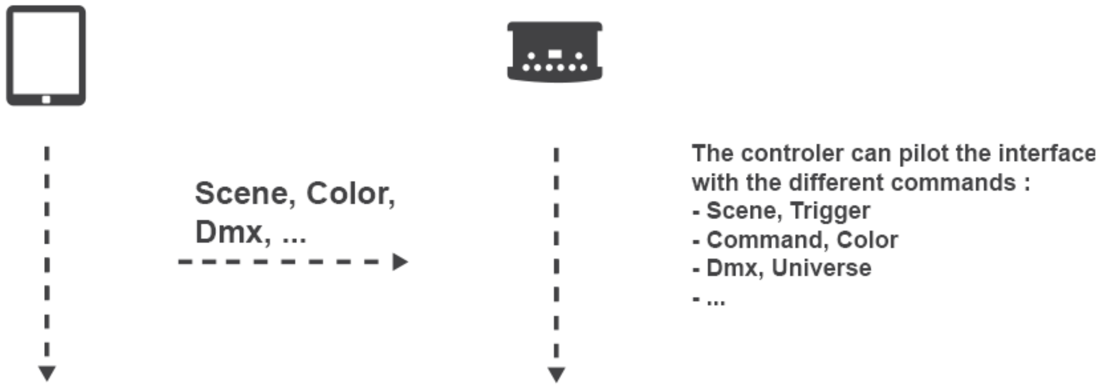
«Alive» message can be sent at any time to check the comunication is still valid.

The controler can pilot the interface with the different commands :
- Scene, Trigger
- Command, Color
- Dmx, Universe
- ...

Sames messages can be used to inform the controler of the actions performed on the interface or software.

Quick and Easy example

If you already know the IP address of your DMX interface(s) and the scenes ID written in memory, you can simply send trigger commands from your application to the interface without the discovery part. Just send a JSON commands as follow directly to the hardware's IP address or by broadcasting on the network to one or several hardware:



The controler can pilot the interface with the different commands :
- Scene, Trigger
- Command, Color
- Dmx, Universe
- ...

Messages Description

MESSAGE	TYPE	JSON CONTENT	DESCRIPTION
ALIVE	1	none	Enables to know if the communication is valid.
LIMIT	2	none	Returned by the server after a “Get State” message when the number max of users has been reached (10 max).
QUIT	3	none	Informs the server that a client closes.
GET IP	4	none	Asks the server to return its IP address.
IP	5	'ip'	Returns the IP address of the server.
GET STATE	6	none	Asks the server to return its state and start the communication.
STATE	7	'scn'	JSON Array of scenes (see SCENE message below)
		'trg'	JSON Array of Triggers (see PROGRAM message below)
		'clr' or 'clr list'	Live color (see COLOR message below)
		'cmd' ot 'cmd list'	Live commands (see COMMAND message below)
SCENE	8	‘n’	Name of the scene **[STATE only]**
		‘Id’	Identifier of the scene
		‘a’	area (Zone) of the scene
		‘l’	Number of loops **[STATE only]**
		‘s’	Status: true means activated
TRIGGER	9	‘n’	Name of the trigger **[STATE only]**
		‘Id’	Identifier of the trigger
		‘a’	area (Zone) of the trigger
		‘s’	Status: true means activated
COLOR	10	‘rgbw’	Activate/Deactivate rgbw color mode
		‘a’	area (Zone) of the color
		‘s’	Status: true means activated
		‘color’	Rgbw values of color
COMMAND	11	‘a’	area (Zone) of the command
		‘bo’	Blackout
		‘next’	Next program/scene
		‘prev’	Previous program/scene
		‘white’	Full white
		‘hold’	Hold execution
		‘dim’	Set dimmer value from 0 to 100
		‘speed’	Set speed value from 0 to 100
	12	‘level’	DMX level from 0 to 255
		‘s’	Status: true means activated
	13	'universe'	Index of the DMX Universe
		'channels'	Array of channels

MESSAGE	TYPE	JSON CONTENT	DESCRIPTION
DMX	12	‘chan’	DMX Channel (if set to -1, the 512 channels of the given universe will be set to the same DMX level)
		‘univ’	DMX Universe
GET SERIAL	14	none	Asks the server to return its serial number
SERIAL	15	'serial'	Returns the serial number of the server (0000 if it's the software, serial number of the device otherwise).
UNIVERSE COLOR	16	'universe'	DMX Universe
		'chan'	1st color channel
		'number'	Number of colors
		'color'	color (see COLOR message)
		'rgbw'	If true, color is RGBW, simple RGB otherwise
GET SCENES STATE	17	none	Asks the server to return the scenes or Triggers currently playing.
SCENES STATE	18	'scn'	JSON Array of scenes (see SCENE message)
		'trg'	JSON Array of Triggers (see TRIGGER message)
GET COMMANDS STATE	19	none	Asks the server to return the state of the live commands. The server will answer with a COMMAND message.
GET CHANNELS STATE	20	none	Asks the server to return the state of the live channels. The server will answer with UNIVERSE message(s) if there are active channels.
COLOR LIST	21	'clr list'	List of live colors (see COLOR message)
COMMANDS LIST	22	'cmd list'	List of live commands (see COMMAND message)
TIMELINE	23	'play'	Play timeline
		'pause'	Pause timeline
		'stop'	Stop timeline
		'go'	Go timeline
		'prev'	Previous marker timeline
		'next'	Next marker timeline

- Alive message :

It enables to know if the communication is valid and can be sent by your software at any time. The Chromateq software will reply with the same message, if not it means that the communication is broken. In Chromateq applications, we check every 2 seconds. This is an optional message.

Example:

```
{
	"typ" : 1
}
```

- Limit message :

Returned by the server after a “Get State” message has been sent when the number max of users has been reached. It means that your software can not drive the the software until another user leaves.

Example:

```
{
	"typ" : 2
}
```

- Quit message :

Must be sent by your software when the application closes. It enables the server to know the communication ended and free the port.

Example:

```
{
	"typ" : 3
}
```

- Get IP message :

Sent by your software to get the IP address of the server. Optional but useful if you can not know the address of the sender when you receive UDP messages.

When you start the communication, you can send the “Get State” message to the server at its IP address (You can see it in Tools menu / Show IP address) or send it in broadcast. If you broadcast your first message, you have to use the server address for other messages. If this information is not provided by your socket, use “Get Ip” message.

Example:

```
{
	"typ" : 4
}
```

<u>- IP message :</u>

Returned by the server after a “Get IP” message has been sent.

Example:

```
{
	"typ" : 5,
	"content" :
		{"ip" : "192.168.0.11"}
}
```

<u>- Get State message :</u>

Must be sent by your software to initialize the communication and get the the software state. It can be sent at any time, even if the communication has started if you want to refresh your application.

You won't be able to drive the software until you send this message.

Example:

```
{
	"typ" : 6
}
```

- State message :

Returned by the server after a “Get State” message has been sent.

“State” message can be sent in multiple sub-messages since some system of sockets don't manage long messages correctly (when it gets over MTU). In that case, your software must read until full JSON message is received. All sub-messages end with a marker : !#(. It has to be removed from each sub message to build the full “Get State” message.

Example:

```
{
        "typ" : 7,
        "content" :
        {
                "scn" :
                {
                        "typ" : 8,
                        "content" :
                                [
                                {"n" : "scn1", "id" : 1, "a" : "0", "l" : "0", "s" : true},
                                {"n" : "scn2", "id" : 2, "a" : "0", "l" : "4", "s" : false},
                                {"n" : "scn3", "id" : 3, "1" : "0", "l" : "2", "s" : false}
                                ]
                },!#(
                "trg":
                {
                        "typ" : 9,
                        "content" :
                                [
                                {"n" : "trg1", "id" : 1, "a" : "0", "s" : true},
                                {"n" : "trg2", "id" : 2, "a" : "1", "s" : false}
                                ]
                },!#(
                "clr" :  (or “clr list”)
                {
                        "typ": 10,
                        "content" :
                                {"rgbw" : false, "a" : "0", "s" : false, "color" : {"r" : 255, "g" : 255, "b" :
                                255, "w" : 0}}
                },!#(
                "cmd" : (or “cmd list”)
                {
                        "typ" : 11,
                        "content" :
                                {"bo" : false, "next" : false, "prev" : false, "white" : false, "hold" :
                                false,  "dim" : 50, "speed" : 50, "a" : "0"}
                }
        }
}!#(
```

- Scene message :

Sent by your software to trigger a scene in the Chromateq softwares.

Example:

```
{
        "typ" : 8,
        "content" :
                {"id" : 1, "a" : 0, "s" : true}
}
```

Trigger message :

Sent by your software to trigger a Trigger button or Program Button in the Chromateq softwares.

Example:

```
{
        "typ" : 9,
          "content" :
                {"id" : 1, "a" : "0", "s" : true}
}
```

- Color message :

Sent by your software to set a Live Color in the Chromateq softwares.

Example:

```
{
        "typ" : 10,
        "content" :
                {"rgbw" : false, "a" : "0", "s" : false, "color" : {"r" : 255, "g" : 255, "b" : 255, "w" : 0}}
}
```

- Command message :

Sent by your software to trigger different commands available in the Chromateq softwares.

Example:

```
{
        "typ": 11,
        "content" :
                {"bo" : false, "next" : false, "prev" : false, "white" : false, "hold" : false, "dim" : 50,
                "speed" : 50, "a" : "0"}
}
```

- DMX message :

Sent by your software to set a DMX level on a single channel.

Example:

```
{
        "typ" : 12,
        "content" :
                {"chan" : 10, "univ" : 0, "level" : 127, "s" : true}
}
```

Universe message :

Sent by your software to set multiple DMX levels on a given DMX universe.

Example:

```
{
        "typ" : 13,
        "content" :
        {
                "universe : 0",
                "channels" :
                        [
                        {"c" : "0",  "l" : 255, "s" : true},
                        {"c" : "12", "l" : 127, "s" : true},
                        {"c" : "57", "l" : 0,   "s" : false}
                        ]
        }
}
```

- Get Serial message :

Sent by your software to get the serial number of the server, Chromateq software or interface. Optional but useful, it can be used before sending a “getsate” message to choose witch server you want to work with if there are several Ethernet devices.

Example:

```
{
        "typ" : 14
}
```

- Serial message :

Returned by the server after a “Get Serial” message has been sent.

Example:

```
{
        "typ" : 15,
        "content" :
                {"serial" : "0123"}
}
```

- Universe Color message :

Sent by your software to set a color on a given DMX universe.

Example:

Write consecutively 100 RGB Colors (R:255, G:255, B:255) from the 1st channel of the 1st DMX Universe.

```
{
        "typ" : 16,
        "content" :
        {
                "universe" : 0,
                "chan" : 0,
                "number" : 100
                "color" : {"r" : 255, "g" : 255, "b" : 255, "w" : 0}
                "rgbw" : false
        }
}
```

- Get Scenes State message :

Sent by your software to get the the list of the scenes that are currently playing.

Example:

```
{
        "typ" : 17
}
```

Scenes State message :

Returned by the server after a “Get Scenes State” message has been sent.

Example:

```
{
        "typ" : 18,
        "content" :
        {
                "scn" :
                {
                        "typ" : 8,
                        "content" :
                        [
                        {"id" : 1, "a" : 0},
                        {"id" : 2, "a" : 1}
                        ]
                },
                "prg":
                {
                        "typ" : 9,
                        "content" :
                        [
                        {"id" : 1},
                        {"id" : 2},
                        {"id" : 3}
                        ]
                }
        }
}
```

- Get Commands State message :

Sent by your software to get the state of Live Commands.

Example:

```
{
        "typ" : 19
}
```

Get Channels State message :

Sent by your software to get the state of Live Channels.
If no active channels, the server doesn't reply.

Example:

```
{
        "typ" : 20
}
```

- Color list message :

Sent by the server to update the live color for each zone/area (Zone).

Example:

```
{
         "typ":21,
        "content":
        [
        {"a":0,"rgbw":true,"s":false,"color":{"r":0,"g":0,"b":0,"w":0}},
        {"a":1,"rgbw":true,"s":false,"color":{"r":0,"g":0,"b":0,"w":0}},
        {"a":2,"rgbw":true,"s":false,"color":{"r":0,"g":0,"b":0,"w":0}},
        {"a":3,"rgbw":true,"s":false,"color":{"r":0,"g":0,"b":0,"w":0}},
        {"a":4,"rgbw":true,"s":false,"color":{"r":0,"g":0,"b":0,"w":0}}
        ]
}
```

- Commands list message :

Sent by the server to update the live commands for each zone/area (Zone).

Example:

```
{
         "typ":22
        "content":
        [
        {"a":0,"bo":false,"next":false,"prev":false,"white":false,"hold":false,"dim":50,"speed":50},
        {"a":1,"bo":false,"next":false,"prev":false,"white":false,"hold":false,"dim":50,"speed":50},
        {"a":2,"bo":false,"next":false,"prev":false,"white":false,"hold":false,"dim":50,"speed":50},
        {"a":3,"bo":false,"next":false,"prev":false,"white":false,"hold":false,"dim":50,"speed":50},
        {"a":4,"bo":false,"next":false,"prev":false,"white":false,"hold":false,"dim":50,"speed":50}
        ]
}
```

- Timeline message :

Sent by your software trigger the timeline in Chromateq softwares.

Example:

```
{
        "typ" : 23,
        "content" :
        {
                "play" : 1,
                "pause" : 0,
                "stop" : 0,
                "go" : 0,
                "prev" : 0,
                "next" : 0
        }
}
```

Chromateq Ethernet Interface

The ports used to drive Chromateq Ethernet Interfaces are different than the softwares ones:
port 8011 to write or send JSON command messages / 8012 to read or receive command message.

If you want to drive an Ethernet Interface and have to send consecutive messages (to set a color for example), we recommend to send the messages every 40ms max (25Hz). If you exceed that limit, you can crash the Ethernet module.

Since we don't use triggers button in our interfaces (Trigger buttons are only use in the Chromateq softwares), you can not play triggers buttons on a Ethernet Interface, only scenes can be triggered. For the same reason, you will receive only scenes in the “State” message.

CONTACTS

TECHNICAL SUPPORT:
https://www.chromateq.com
support (at) chromateq.com
You can visit our web site and use the online form for technical support.
Chromateq will reply shortly.

CRASH REPORT:
https://www.chromateq.com
You can visit our web site and use the online form for a crash report.
Chromateq will reply shortly and could supply an update file quickly.

GENERAL INFORMATION AND PURCHASE:
https://www.chromateq.com
info (at) chromateq.com
Tel: +33 (0)952210755
https://www.chromateq.com
CHROMATEQ, Lighting and Video Control Solutions.