CTOUCH®
The power to engage

Product & Innovation

Neo JSON API specification

Version	Date	Changes		Author(s)
1.0	30-08-2022	First release	P&I	Hans Crijns
1.1	13-01-2023	Updated commands table	P&I	Hans Crijns
1.2	22-03-2023	Updated commands table	P&I	Hans Crijns

Authors: Hans Crijns

Introduction

This document describes the CTOUCH 'Remote Management API' and it's details. This document primary focusses on the Neo screen, but is written with applicability to future products in mind.

API Overview

The API consists of a HTTP server running on the screen which reacts to JSON-requests. The JSON requests contain a hash, timestamp and command, the response is JSON as well and consist of version, type and result field. In case the server cannot respond in the right way, it will send an error message. All these are described in more detail in the next few chapters.

Interface

- Request should be sent via a HTTP POST

- Request URL is: <ip-address>:<port>/managementapi

- With port being 8110

- Normally the response should be within 500ms.

Response with HTTP Status Code:

HTTP Status Code	Situation
400	If not able to process request at all (malformed request)
401	If token is not valid
200	All other cases

The response should always contain a body text. See below for the different type of responses.

- Timestamp: current time, in ISO 8601 format

- Hash: SHA-256 of timestamp + token

- Type api_request commands: get / set

Next to checking the hash, also the timestamp can be validated. If timestamp differs more than X hours for current time of screen, the request will be replied with an error message.

Enable JSON API and get secret token

In order to enable the JSON API, it needs to be enabled in the Settings menu under the Management submenu. After enabling the JSON API, a security token will be generated and shown.

The screen's IP address can be found in the Settings' Device submenu by clicking Details.

Security

The API requests run over http with the transmitted data not secret nor sensitive. An authentication mechanism is used to make sure the request comes from an authorized source.

The API provides a basic authorization/authentication mechanism using a authentication token based hash, generated at random by the screen and consisting of 8 characters. The token is be generated by the screen and stored on the mainboard. All API requests need to contain a hash based on this token and data from within the request.

Resetting of the token is not possible from a remote location: The user has to be physically present at the screen. When Remote Management API is disabled (in the dealer menu) and re-enabled later, a new token will be generated (this is also the way to "reset" the token)

Read / Get

Example of reading value

Read requests (for status or configuration information):

```
{"api_request": {
        "hash": "ba059253dd5b2878f4dca2427af4edd20b373accf33afa43b68d2f46c0044c20",
        "timestamp": "2019-08-14T13:56:32.427Z",
        "command": {
                "type":"get",
                "value_of": "Source"
        }
}}
```

```
# In these examples 6wfx9j1t is used as token to calculate the hash value
# above hash is SHA256("2019-08-14T13:56:32.427Z6wfx9j1t")
```

Response (in body):

```
{"api_response": {
        "version" : "1.0",
        "type" : "get",
        "result": {
                "Source" : "HDMI1"
        }
}}
```

Example of reading object

A example with ConfigExport object:

```
{"api_request": {
```

```
        "hash": "ba059253dd5b2878f4dca2427af4edd20b373accf33afa43b68d2f46c0044c20",
        "timestamp": "2019-08-14T13:56:32.427Z",
         "command": {
                "type":"get",
                "value_of": "ConfigExport"
        }
}}
```

Response: (in body)

```
{"api_response": {
        "version" : "1.0",
        "type" : "get",
        "result": {
                "ConfigExport" : {
                        "PictureMode": "Dymamic","Contrast": 80,"Brightness": 65, etc
                }
        }
}}
```

Write / Set

Example of setting single value

Write requests for configuration settings:

```
{"api_request": {
        "hash": "cf1dafdc67bcb4904be45f020b059b17977177cdcae24bfa90e00c25ba184675",
        "timestamp": "2019-08-14T14:16:09.835Z",
        "command": {
                "type":"set",
                "Source": "HDMI1"
        }
}}
```

Response: (in body)

```
{"api_response": {
        "version" : "1.0",
        "type":"set",
        "result": {
                "Source": "HDMI1" – echo request
        }
}}
```

Error response

Above examples show the response in case of a valid request. If the request cannot be carried out an error message will be replied: (in body)


```
{"api_response": {
        "version" : "1.0",
        "type":"error",
        "result": {
                "message": "Unknown key",
                "error-code": 30
        }
}}
```


Error-codes:

10 – Not authorized - also return HTTP-Status Code 401 instead of 200
11 – Request outdated - also return HTTP-Status Code 401 instead of 200
20 – Invalid document - also return HTTP-Status Code 400 instead of 200
30 – Unknown key
40 – Invalid key value combination

Command overview

Note: *Grey* items may not have been implemented (correctly)

Key	RD	WR	Values
Picture			
Energy Mode	X	X	Auto/Normal/Ecofriendly
PictureMode	X	X	Dynamic/Standard/Soft
ColorTemp	X	X	Cool/Normal/Warm
Backlight	X	X	0-100
BlueLightFilter	X	X	On/Off
EyeProtect	X	X	On/Off
Sound			
SoundMode	X	X	Standard/Music/Movie/Sports/User
MediaOut	X	X	Speaker/SPDIF/ARC/Bluetooth/Speaker+SPDIF
Balance	X	X	-50 … 50
Treble	X	X	0-100
Mid	X	X	0-100
Bass	X	X	0-100
AVC	X	X	On/Off
SPDIFMode	X	X	RAW/PCM
MaxVolume	X	X	0-100
Microphone	*X*	*X*	*On/Off*
Input			
HDMI1	X	X	On/Off
HDMI2	X	X	On/Off
HDMI3	X	X	On/Off
USB-C	X	X	On/Off
DP	X	X	On/Off
OPS	X	X	On/Off
Audio	X	X	On/Off

AutoSwitchSource	X	X	On/Off
HDMIEDID	X	X	1.4/2.0
HDMICEC	X	X	On/Off
HDMIARC	X	X	On/Off
HDMIOut	X	X	1080p60Hz/2160p30Hz/2160p60Hz (only with external display connected)
Touch	X	X	On/Off
Power			
NoSignalPowerOff	X	X	Off/1m/3m/5m/10m/15m/30m/45m/60m
StartUpByExternal	X	X	On/Off
OnTimerPeriod	X	X	Once/Everyday/MonToFri/MonToSat/SatToSun /Mo/Tu/We/Th/Fr/Sa/Su
OnTimerTime	X	X	00:00 – 23:59
OffTimerPeriod	*X*	*X*	Once/Everyday/MonToFri/MonToSat/SatToSun /Mo/Tu/We/Th/Fr/Sa/Su
OffTimerTime	X	X	00:00 – 23:59
PowerOnMode	X	X	Direct/Stand-by/Memory
PowerOnSource	X	X	Memory/HDMI1/HDMI2/HDMI3/USB-C/DP/Android
PowerOnVolume	X	X	Memory/Preset/0-100
PowerOnOPSDirect	X	X	On/Off
CTOUCHButton	X	X	On/Off
General			
Android	X	X	On/Off
Home	X	X	ANDROID/HDMI1/HDMI2/HDMI3/USB-C/DP/OPS/Audio
DateTime	X	X	ISO8601-format (example: 2021-05-15T11:12:48.535Z)
AutoDateTime	X	X	On/Off
Language	X	X	UK/NL/D
IRLock	X	X	On/Off
SettingsMenuPincode	X	X	On/Off
OTA	X	X	On/Off/Auto
SourceLabel	X	X	{"Source":"", "Label": "New source name"}

Smart On/Off			
SmartOnOff.Wakeup	X	X	On/Off
SmartOnOff.Standby	X	X	On/Off
SmartOnOff.WakeUpType	X	X	LowDistraction/FullColor
SmartOnOff.StandbyTimer	X	X	1m/3m/5m/10m/15m/30m
Connectivity			
UARTBaudRate	X	X	4800/9600/19200/38400/57600/115200
UARTID	X	X	0-255
SettingsMenuPincode	X	X	On/Off
OPSEthernet	X	X	On/Off
WakeOnLan	X	X	On/Off
USB	X	X	On/Off/TouchOnly
MainboardCon.Ethernet	X	X	On/Off
MainboardCon.Wifi	X	X	On/Off
MainboardCon.Hotspot	X	X	On/Off (Note: disables MainboardCon.Wifi)
MainboardCon.Bluetooth	X	X	On/Off
Control			
Source	X	X	HDMI1/HDMI2/HDMI3/USB-C/DP/PC
Freeze	X	X	On/Off
Volume	X	X	0-100
Volume_Mute	X	X	On/Off
Backlight_Mute	X	X	On/Off
Power	X	X	Off
Floatbar buttons			
QuickControlButtons.Source	X	X	On/Off
QuickControlButtons.Volume	X	X	On/Off

QuickControlButtons.Mute	X	X	On/Off
QuickControlButtons.BacklightMute	X	X	On/Off
QuickControlButtons.Brightness	X	X	On/Off
QuickControlButtons.Shutdown	X	X	On/Off
QuickControlButtons.Settings	X	X	On/Off
QuickControlButtons.Capture	X	X	On/Off
QuickControlButtons.Annotate	X	X	On/Off
QuickControlButtons.Home	X	X	On/Off
QuickControlButtons.Back	X	X	On/Off
QuickControlButtons.AllApps	X	X	On/Off
Info			
ProductName	X	-	CTOUCH Neo
API_Version	X	-	0-255
DisplayInfo	X	-	Model, Serial Number, Resolution, MCU firmware, Main firmware
DisplayUsage	X	-	TotalOperation, LastBootUpTime, PowerUpCycles, SourceSwitches, VolumeChanges, ConfigurationChanges
Sensors	X	-	Motion, LightLevel
ConfigExport	X	-	All parameters as JSON
Execute command			{"type": "exectute"}
COSOATUpdate			

Python example test code

```
import datetime, hashlib, requests

# Install and use:
# 1. Download Python at https://www.python.org/downloads/
# 2. Install requests from commandline: pip install requests
# 3. Copy script and execute from commandline(example): python jsontest.py

## Send get or set command to screen
def getset(ipval, tokenval, cmdval, parameterval, value = ''):
	timestamp = datetime.datetime.utcnow().strftime('%Y-%m%dT%H:%M:%S.%f')[:3] + 'Z'
	hv = (timestamp + tokenval).encode('utf-8')
	hashval = hashlib.sha256(hv).hexdigest()

	if cmd == 'get':
		payload =
{'api_request':{'hash':hashval,'timestamp':timestamp,'command':{'type':'get','value_of':parameterval}}}
	else:
		payload =
{'api_request':{'hash':hashval,'timestamp':timestamp,'command':{'type':'set', parameterval: value}}}

	try:
		r = requests.post(url = 'http://' + ipval + ':8110/managementapi', json = payload, timeout = 3)
	except Exception as e:
		sys.exit("Error: No (good) reponse from screen; Possibly wrong IP address used?")
	return(r.json())

## Example execution of getset-function
if __name__ == "__main__":
	ip = '192.168.108.70'
	token = 'bz7gm494'
	parameter = 'Source'
	setvalue = 'HDMI2'

	# Get command example
	cmd = 'get'
	gvalue = getset(ip, token, cmd, parameter)
	print(gvalue)

	# Get command example
	cmd = 'set'
	svalue = getset(ip, token, cmd, parameter, setvalue)
	print(svalue)
```

Notes:

- Indentation is important in Python; Copy indentation exactly from above
- Some lines (payload and requests lines) are overflowing into a next line in this document, when using the code code: place them on one line! (just copy and paste into .py file)
- Make sure you update ‘ip’ and ‘token’ to values from your screen