FIBARO System

REST API Developer Documentation

Note:This documentation describes features available in Home Center version 4.0 Beta or newer.

Introduction

Welcome to the Fibaro Home Center Rest API,

FHCR API is a tool to help developers everywhere create amazing applications using unlimited capabilities of our system. Fibaro provides a simple RESTful API where each type of resource has a URL that you can interact with. All resources are encoded as JSON objects.

Using this website, you will get to know what kind of functions are available, find out how to use them and what are their responses. We hope this will help you to truly use Fibaro as you want it, by making new apps, websites and many others, integrating our system into something else or just playing around.

The software, trademarks documentation, and any other materials we provide to help you develop Fibaro Home Center, including especially the interface specifications "API" belong to Fibar Group. It may happen that working on an app suggests an idea to you for an improvement in the API or our materials. If you suggest any improvements to us and we adopt them, they become part of the platform used by everyone, and belong to us.

The interface between your apps and the Home Center will evolve over time, but we will do our best to maintain backwards compatibility and will inform you timely before we roll out updates.

You may refer to Fibaro in plain text but you are not allowed to use Home Center name and branding or to use Fibaro in any logo or graphics. What you are allowed to do is to experiment and have fun.

Conditions of use

You are free to develop any kind of application you can imagine with Fibaro system. There are just a few rules and restrictions you need to keep in mind:

You may not distribute the documentation shown in this document except by links to the site itself or, if you use another method, these conditions of use must be attached.

We want all your apps to work with our API to form a rich ecosystem of interoperable applications, so it is a condition of access to our API documentation that you do not use it to develop or distribute any systems or devices which interpret the Fibaro API.

It may happen that working on an app suggests an idea to you for an improvement in the API. If you suggest any improvements to us and we adopt them, they become part of the platform used by everyone, and belong to Fibaro, you will make no claims in this respect.

Make sure it is very clear from all you do that your app belongs to you and not to Fibaro. Do not use any Home Center or Fibaro branding trademarks or trade dress in any logo or graphics.

If you receive our API developer materials, you cannot claim ownership or IP rights in any improvements of that material or in any of the APIs either with respect to us or to other Fibaro app developers.

Please don't make any applications that are obscene, not compliant with laws and regulations, offensive or discriminatory or that infringe someone else's rights.

So, just as a reminder, before you can start having fun, you agree that by using the API provided here to you, you accept these terms of use.

How API works

The FHCR API is your primary tool for controlling your smart home. This is a RESTful interface over HTTP. The purpose of this web service interface is to give every single device in your system and every controllable parameter a URL in your local network. This means that controlling the system is achieved by simply sending a new value to this local URL.

You can simply discover the current status of any variable by getting a response, check it and make a change just by sending its the new value. That’s the basic idea of a RESTful interface. All responses and new values are sent and returned in JSON (JavaScript Object Notation) with UTF8 encoding so it’s easy to generate or parse.

How Fibaro works

Fibaro is a wireless system, based on Z-Wave technology. Fibaro provides many advantages when compared to similar systems. In general, radio systems create a direct connection between the receiver and transmitter. However, the radio signal is weakened by various obstacles located in its path (apartment walls, furniture, etc.) and in extreme cases it fails to transfer required data. The advantage of Fibaro System is that its devices, apart from being transmitters and signal receivers, also duplicate signal. When a direct connection path between the transmitter and the receiver cannot be established, the connection may be achieved through other intermediate devices.

Fibaro is a bi-directional wireless system. This means that the signal is not only sent to the receivers but also the receivers send the confirmation of its reception. This operation confirms their status, which checks whether they are active or not. Safety of the Fibaro System transmission is comparable to the safety of transmission in data bus wired systems.

Fibaro operates in the free bandwidth for data transmission. The frequency depends on radio regulations in individual countries. Each Fibaro network has its own unique network identification number (home ID), which is why it is possible to co-operate two or more independent systems in a single building without any interference. In addition, each device gets its own ID – Node ID. Each, newly added device gets two ID numbers – HOME ID and Node ID. Home ID is the same for all devices within the network, while Node ID is unique for a given node. If another controller (secondary master) is added to the network, it gets the same HOME ID as the main controller.

Although Z-Wave is quite a new technology, it has already become recognized and officially a binding standard, similarly to Wi-Fi. Many manufacturers in various industries offer solutions based on Z-Wave technology, guaranteeing their compatibility. This means that the system is open and it may be extended in the future. Find more information at www.fibaro.com (http://www.fibaro.com/).

Fibaro generates a dynamic network structure. After Fibaro System is switched on, the location of its individual components is automatically updated in real-time through status confirmation signals received from devices operating in a “mesh” network.

Getting started

Make sure you have your Fibaro Home Center working properly. In case of problems please go here, we will help you as soon as possible.

The fastest way to learn how to build apps which control our system is to use the simple test web app built into the main controller. This lets you directly input commands and send them.

First you need is to discover Home Center IP address. You can use Fibaro Finder or just type in IP address into your web browser.

The simplest thing you can do with a FHCR API resource URL is GET it. (When using REST API via HTTP, you “read” something by using the HTTP GET method).

Once you have the IP visit the following address in your web browser.

http://<Home Center IP address>/docs

This website contains a list of available functions grouped into the categories. You can simply click one of theme to expand a list of associated functions and methods with short descriptions.

Now you can select one operation by clicking it. It will show its Response Class, model and model schema. Detailed description is available at this website under "API" section. "Try it out" button lets you test selected operation on your connected Fibaro Home Center controller.

To retrieve a specific resource you can also append its identifier to the end of the URL. The example below shows retrieving a specific device using the HTTP GET method.

http://192.168.77.80/api/devices/1771

Document organization

API functions are divided into five main categories. Each of them contains function URL, description, available methods, response with table of variables and example.

o General API

o Settings API

o Panels API

o Plugins API

o Other

Glossary terms

API – Application Programming Interface

REST – Representational State Transfer

URL – Uniform Resource Locator

JSON – JavaScript Object Notation

FHCR API – Fibaro Home Center Rest API

Timestamp – a way to track time as a running total of seconds. This count starts at the Unix Epoch on January 1st, 1970 at UTC.

Message structure and response

The FHCR API consists of a set of commands that can be called over a REST web service. The API commands fall into one of 4 categories, depending on the HTTP method used:

Method: GET

Used for: Reading specific data from the Home Center controller.

Returns: JSON containing the requested resource.

Method: PUT

Used for: Modifying existing data on the Home Center controller.

Returns: A list containing one item per modified parameter.

Method: POST

Used for: Adding new data to the Home Center controller.

Returns: A list containing one item per created resource.

Method: DELETE

Used for: Deleting data from the Home Center controller.

Returns: A list containing one item per deleted resource.

Commands using PUT and POST methods will normally require a message body to be attached to the request. The message body must be formatted using JSON. More details and examples for formatting the message body can be found in the documentation for each command.

Response codes

API may return the following HTTP response codes

HTTP Status Code	Description
200	OK
400	Bad request – missing parameter
401	Unauthorized – authentication required
403	Forbidden – valid request, no server response
404	Not found – no content
405	Method not allowed – no content
500	Internal server error – unexpected condition
501	Not implemented – no content
502	Bad gateway – invalid response from the server
503	Service unavailable – server overloaded or temporarily down for maintenance
504	Gateway timeout – no timely response from the server

Data types

Type	Description
Number	A whole number (not a fractional number) that can be positive, negative, or zero
String	A sequence of characters
Boolean	A boolean value which can take only the true or false values
Array	A variable that holds multiple values of the same type

Note:

As you can see, by default a lot of data types are displayed as strings. It's necessary to add a specified custom header to get the correct RESTful requests data types as described in following example.

Example:

1. Typing an address like http://HC2IP/api/devices will give us an old and incomplete data structure.
2. If our goal is to get the correct data types (eg. bool, int, etc.), we need to install the REST Console, such as https://chrome.google.com/webstore/detail/rest-console/cokgbflfommojglbmbpenpphppikmonn (https://chrome.google.com/webstore/detail/rest-console/cokgbflfommojglbmbpenpphppikmonn) in our web browser.
3. Now you can type the address like http://HCIP/api/devices into the Request UR field as presented on the screenshot below.
   Custom Headers (/files/rest-api/restapi-1.png)
4. Then please type Fibaro Header - X-Fibaro-Version:2 into the Custom Headers field as showed below.
   Custom Headers (/files/rest-api/restapi-2.png)
5. You will see the correct data types as a response.

FGHC Rest API Functions

Settings

General settings

URL: /api/settings/info

Methods: GET, PUT

Description: Returns a list of parameters of Home Center controller, such as serial number, soft version or default language, etc.

Response: Gets an object containing all controller’s general settings.

Name	Type	Description
serialNumber	String	Home Center serial number
mac	String	Mac address of the controller
softVersion	String	Version of installed software
beta	Boolean	Beta software status
zwaveVersion	String	Version of Z-wave software
serverStatus	Number	Status of server in seconds
DefaultLanguage	String	Default interface language
sunsetHour	String	Time of sunset
sunriseHour	String	Time of sunrise
hotelMode	Boolean	Hotel mode status
updateStableAvailable	Boolean	Availability of stable update
temperatureUnit	String	Temperature unit
updateBetaAvailable	Boolean	Availability of beta version update
newestBetaVersion	String	Newest version of beta
batteryLowNotification	Boolean	Battery low notification status

smsManagement	Boolean	Sms notifications status

Example:

```
{
  "serialNumber": "HC2-000584",
  "mac": "38:60:77:4e:5c:11",
  "softVersion": "3.590",
  "beta": false,
  "zwaveVersion": "3.42",
  "serverStatus": 1404743890,
  "defaultLanguage": "en",
  "sunsetHour": "21:08",
  "sunriseHour": "04:51",
  "hotelMode": false,
  "updateStableAvailable": false,
  "temperatureUnit": "C",
  "updateBetaAvailable": true,
  "newestBetaVersion": "3.591",
  "batteryLowNotification": true,
  "smsManagement": false
}
```

Backups

URL: /api/settings/backups

Methods: GET

Description: Returns a list of saved controller's backups and their parameters like number of devices, rooms or scenes, etc.

Response: Gets an object containing stored backups.

Name	Type	Description
id	Number	Backup id
timestamp	Number	Timestamp in seconds
devices	Number	Number of devices
rooms	Number	Number of rooms
scenes	Number	Number of scenes
description	String	Backup description created by the user

Example:

```
{
  "id": 9,
  "timestamp": 1405506755,
  "devices": 143,
  "rooms": 5,
  "scenes": 10,
  "description": "alpha->3.903"
}
```

Location

URL: /api/settings/location

Methods: GET, PUT

Description: Returns a list of parameters related to date, time and location configured by user in Home Center interface.

Response: Gets objects containing date, time and location settings.

Name	Type	Description
houseNumber	Number	House number
timezone	String	Selected timezone
ntp	Boolean	Network time protocol status
ntpServer	String	Selected ntp server path
date: day, month, year	Number	Set date
time: hour, minute	Number	Set time
latitude	Number	Set latitude
longitude	Number	Set longitude
city	String	Selected city
temperatureUnit	String	Selected temperature unit
windUnit	String	Selected wind unit
timeFormat	Number	Time format (24/12)
dateFormat	String	Date format

Example:

```
{
  "houseNumber": 3,
  "timezone": "Europe/Warsaw",
  "ntp": true,
  "ntpServer": "",
  "date":
  {
    "day": 16,
    "month": 7,
    "year": 2014
  },
  "time":
  {
    "hour": 15,
    "minute": 11
  },
  "latitude": 52.425035319943,
  "longitude": 16.9306182861328,
  "city": "Poznan",
  "temperatureUnit": "C",
  "windUnit": "km/h",
  "timeFormat": 24,
  "dateFormat": "dd.mm.yy"
}
```

Network settings

URL: /api/settings/network

Methods: GET, PUT

Description: Returns a list of parameters related to network connection, such as DHCP status, remote access availability or IP number.

Response: Gets an object containing network settings.

Name	Type	Description
dhcp	Boolean	DHCP status
ip	String	Home Center IP address
mask	String	Subnet mask
gateway	String	Default gateway
dns	String	DNS server address
remoteAccess	Boolean	Remote access availability status

Example:

```
{
  "dhcp": true,
  "ip": "192.168.100.45",
  "mask": "255.255.254.0",
  "gateway": "192.168.100.1",
  "dns": "192.168.100.1",
  "remoteAccess": true
}
```

General

Devices

URL: /api/devices

Methods: GET, DELETE, POST, PUT

Description: Returns a list of devices, containing the main controller, all added devices, virtual devices and plugins as well, including all their parameters, properties and actions. Number of available data depends on the selected device.

Response: Gets an array of objects containing all devices and their parameters.

Name	Type	Description
id	Number	Device ID
name	String	Device name
roomID	Number	Room ID
type	String	Device type
baseType	String	Base type
enabled	Boolean	Device status
visible	Boolean	Visibility status
parentId	Number	Parent device ID
remoteGatewayId	Number	Remote gateway ID
viewXml	Boolean	Xml view status
configXml	Boolean	Xml configuration status
interfaces	Array	Available interfaces
created	Number	Time of creation
modifier	Number	Time of last modification
sortOrder	Number	Interface sort order

properties:

zwaveCompany	**String**	**Z-Wave chip producer**
zwaveInfo	String	Z-Wave protocol type and version
zwaveVersion	Number	Z-Wave version
wakeUpTime	Number	Wake up time in seconds
pollingTimeSec	Number	Polling time in seconds
batteryLevel	Number	Battery level in percent
alarmDelay	String	Time of alarm delay in seconds
alarmExclude	String	Alarm exclusion status
alarmTimeTimestamp	String	Alarm timestamp in seconds
armConditions	String	Conditions of arming an alarm
armConfig	String	Arming configuration
armDelay	String	Arming delay in seconds
armError	String	Error of arming
armTimeTimestamp	String	Arming timestamp in seconds
armed	String	Status of arming
batteryLowNotification	String	Low battery notification status
configured	String	Check if device is configured
dead	String	Check if device is dead
deviceControlType	String	Device control type
deviceIcon	String	Interface device icon
emailNotificationID	String	ID of email notification
emailNotificationType	String	Type of email notification
endPointId	String	ID of endpoint
fibaroAlarm	String	Status of usage in Fibaro Alarm
interval	String	Interval in seconds
lastBreached	String	Time of the last breach
liliOffCommand	String	Lili turn off command
liliOnCommand	String	Lili turn on command
log	String	Log status

logTemp	String	Temperature from log
manufacturer	String	Manufacturer of the device
markAsDead	String	Mark as dead if dead status
model	String	Model of the device
nodeId	String	Node ID
parametersTemplate	String	Template of paramaters
productInfo	String	Product version info
pushNotificationID	String	Push notification ID
pushNotificationType	String	Type of push notification
remoteGatewayId	String	ID of the remote gateway
saveLogs	String	Saving logs to event panel status
smsNotificationID	String	ID of sms notification
smsNotificationType	String	Type of sms notification
useTemplate	String	Template usage status
value	String	Current value

actions:

forceArm	**Number**	**Force arming of the device**
meetArmConditions	Number	Meet device arming conditions
reconfigure	Number	Perform device reconfiguration
setArmed	Number	Set device armed
setInterval	Number	Set interval

Example:

```
{
  "id": 1898,
  "name": "1897.0",
  "roomID": 0,
  "type": "com.fibaro.FGMS001",
  "baseType": "com.fibaro.motionSensor",
  "enabled": true,
  "visible": true,
  "parentId": 1897,
  "remoteGatewayId": 0,
  "viewXml": false,
  "configXml": false,
  "interfaces":
  [
    "battery",
    "zwave",
    "zwaveWakeup"
  ],
  "properties":
  {
    "zwaveCompany": "Fibargroup",
    "zwaveInfo": "3,3,67",
    "zwaveVersion": 2.6,
    "wakeUpTime": 4000,
    "pollingTimeSec": 0,
    "batteryLevel": 100,
    "alarmDelay": "0",
    "alarmExclude": "false",
    "alarmTimeTimestamp": "0",
    "armConditions":
    {
      "auto": false,
      "devices":
      [
        {
          "id": 1898,
          "propertyName": "value",
          "propertyValue": "0"
        }
      ],
      "time": 0
    },
    "armConfig": "0",
    "armDelay": "0",
    "armError": "{}",
    "armTimeTimestamp": "0",
    "armed": "false",
    "batteryLowNotification": "true",
    "configured": "true",
    "dead": "false",
    "deviceControlType": "0",
    "deviceIcon": "90",
    "emailNotificationID": "0",
    "emailNotificationType": "0",
    "endPointId": "0",
    "fibaroAlarm": "false",
    "interval": "0",
    "lastBreached": "1405522313",
    "liliOffCommand": "",
    "liliOnCommand": "",
    "log": "",
        "logTemp": "",
        "manufacturer": "",
        "markAsDead": "true",
        "model": "",
        "nodeId": "97",
        "parametersTemplate": "270",
        "productInfo": "1,15,8,0,16,1,2,6",
        "pushNotificationID": "0",
        "pushNotificationType": "0",
        "remoteGatewayId": "0",
        "saveLogs": "true",
        "smsNotificationID": "0",
        "smsNotificationType": "0",
        "useTemplate": "true",
        "value": "false"
      },
      "actions":
      {
        "forceArm": 0,
        "meetArmConditions": 0,
        "reconfigure": 0,
        "setArmed": 1,
        "setInterval": 1
      },
      "created": 1405516322,
      "modified": 1405516322,
      "sortOrder": 121
    }
```

Sections

URL: /api/sections

Methods: GET, DELETE, POST, PUT

Description: Returns a list of sections, their names, sort orders, etc.

Response: Gets an object containing all sections defined in the interface.

Name	Type	Description
id	Number	Section ID
name	String	Section name
sortOrder	Number	Interface sort order

```
    {
      "id": 2,
      "name": "Floor",
      "sortOrder": 2
    }
```

Rooms

URL: /api/rooms

Methods: GET, DELETE, POST, PUT

Description: Returns a list of rooms, their names, icons, sort orders, etc.

Response: Gets objects containing all rooms defined in the interface.

Name	**Type**	**Description**
id	Number	Room ID
name	String	Section name
sectionID	Number	Section ID
icon	String	Room icon
defaultThermostat	Number	Main thermostat
sortOrder	Number	Interface sort order

defaultSensors:

temperature	**Number**	**Main temperature sensor**
humidity	Number	Main humidity sensor
light	Number	Main light sensor

Example:

```
{
  "id": 1,
  "name": "bathroom",
  "sectionID": 1,
  "icon": "room_bathroom",
  "defaultSensors":
  {
    "temperature": 1701,
    "humidity": 1777,
    "light": 0
  },
  "defaultThermostat": 0,
  "sortOrder": 1
}
```

Scenes

URL: /api/scenes

Methods: GET, DELETE, POST, PUT

Description: Returns a list of all saved scenes and their parameters, such as name, id and sort order.

Response: Gets an object containing scenes defined in the interface.

Name	Type	Description
id	Number	Section ID
name	String	Section name
type	String	Type of scene
properties	String	Properties of scene
roomID	Number	Room ID
iconID	Number	Icon ID
enabled	Boolean	Scene status
autostart	Boolean	Autostart status
protectedByPIN	Boolean	PIN protection status
killable	Boolean	Ability to be interrupted
runningInstances	Number	Number of running instances
isLua	Boolean	Status of being LUA scene
liliStartCommand	String	Lili start command
liliStopCommand	String	Lili stop command
sortOrder	Number	Interface sort order

Example:

```
{
  "id": 20,
  "name": "New Scene",
  "type": "",
  "properties": "",
  "roomID": 0,
  "iconID": 5,
  "enabled": true,
  "autostart": false,
  "protectedByPIN": false,
  "killable": true,
  "runningInstances": 0,
  "isLua": false,
  "liliStartCommand": "",
  "liliStopCommand": "",
  "sortOrder": 119
}
```

Users

URL: /api/users

Methods: GET, DELETE, POST, PUT

Description: Returns a list of users, their names, types, rights, etc.

Response: Gets an object containing all users added to the interface.

Name	**Type**	**Description**
id	Number	Section ID
name	String	Section name
type	String	Type of user
email	String	User's email address
hasGPS	Boolean	GPS user status
deviceRights	Array	Rights to edit devices
sceneRights	Array	Rights to edit scenes
hotelRoom	Number	Hotel mode
sendNotification	Boolean	Notifications sending status
tracking	Number	Tracking status
usePin	Boolean	PIN usage status
useOptionalArmPin	Boolean	Optional arming PIN status
initialWizard	Boolean	Sort order in the interface

Example:

```
{
  "id": 1919,
  "name": "test",
  "type": "user",
  "email": "test@test.pl",
  "hasGPS": false,
  "deviceRights":
  [
  ],
  "sceneRights":
  [
  ],
  "hotelRoom": 0,
  "sendNotifications": false,
  "tracking": 0,
  "usePin": false,
  "useOptionalArmPin": false,
  "initialWizard": true
}
```

Global variables

URL: /api/globalVariables

Methods: GET, DELETE, POST, PUT

Description: Returns a list of global variables, their values and parameters.

Response: Gets an object containing all global variables.

Name	Type	Description
name	String	Name of variable
value	String	Variable value
readOnly	Boolean	Read only status
isEnum	Boolean	Enum type status

Example:

```
{
  "name": "var1",
  "value": "1",
  "readOnly": false,
  "isEnum": false
}
```

RGB programs

URL: /api/RGBPrograms

Methods: GET, DELETE, POST, PUT

Description: Returns a list of RGB lights programs.

Response: Gets an object containing RGB lights programs.

Name	Type	Description
id	Number	Program ID
name	String	Program name

Example:

```
{
  "id": 1,
  "name": "Fireplace"
}
```

Tracking schedules

URL: /api/trackingSchedules

Methods: GET, DELETE, POST, PUT

Description: Returns a list of weekly tracking schedules divided into four parts of the day.

Response: Gets an object containing weekly tracking schedules.

Name	Type	Description
id	Number	Current weather condition code
MondayMorningHour	String	Monday morning start hour
MondayMorningHourTo	String	Monday morning end hour
MondayMorningTime	String	Monday morning time

Example:

```
{
  "id": 2,
  "MondayMorningHour": "6",
  "MondayMorningHourTo": "12",
  "MondayMorningTime": "0",
  "MondayDayHour": "12",
  "MondayDayHourTo": "18",
  "MondayDayTime": "0",
  "MondayEveningHour": "18",
  "MondayEveningHourTo": "24",
  "MondayEveningTime": "0",
  "MondayNightHour": "24",
  "MondayNightHourTo": "6",
  "MondayNightTime": "0"
}
```

Linked devices

URL: /api/linkedDevices

Methods: GET, DELETE, POST, PUT

Description: Returns a list of linked devices and their parameters.

Response: Gets an array of objects containing all linked devices.

Name	Type	Description
id	Number	Link ID number
name	String	Linked device name
roomID	Number	Room ID number
type	String	Linked device type
deviceID	Number	Device ID number
created	Number	Time of creation
modified	Number	Time of last modification
sortOrder	Number	Interface sort order

devices	**Array**	**Array of linked devices**
id	Number	Linked device ID number
innerType	String	Device inner type

Example:

```
{
  "id": 3,
  "name": "New Linked Devices",
  "roomID": 1,
  "type": "heating",
  "deviceID": 1716,
  "devices":
  [
    {
      "id": 1854,
      "innerType": ""
    }
  ],
  "created": 1405605040,
  "modified": 1405605040,
  "sortOrder": 111
}
```

Virtual devices

URL: /api/virtualDevices

Methods: GET, DELETE, POST, PUT

Description: Returns a list of virtual devices, their source codes, properties and actions.

Response: Gets an array of objects containing all virtual devices.

Name	Type	Description
id	Number	Virtual device ID number
name	String	Virtual device name
roomID	Number	Room ID number
type	String	Type of device
created	Number	Time of creation
modified	Number	Time of last modification
sortOrder	Number	Interface sort order

properties:

deviceIcon	**Number**	**Virtual device icon**
ip	String	IP address

port	Number	Port number
currentIcon	String	Virtual device current icon
mainLoop	String	Main loop code
saveLogs	String	Number of log

rows	**Array**	**Array of rows**
type	String	Type of element

elements	**Array**	**Array of elements**
id	Number	Element ID number
lua	Boolean	Lua usage status
waitForResponse	Boolean	Waiting for response status
caption	String	Showed caption
name	String	Name of element
empty	Boolean	Empty status
msg	String	Message text
buttonIcon	Number	Element icon
favourite	Boolean	Status of being favourite
main	Boolean	Status of being main element

actions:

pressButton	**Number**	**Press button**
setSlider	Number	Set slider
setProperty	Number	Set property

Example:

```
{
  "id": 167,
  "name": "Scene activation",
  "roomID": 0,
  "type": "virtual_device",
  "properties":
  {
    "deviceIcon": 0,
    "ip": "",
    "port": 0,
    "currentIcon": "0",
    "mainLoop": "",
    "saveLogs": "1",
    "rows":
    [
      {
        "type": "button",
        "elements":
        [
          {
            "id": 1,
            "lua": true,
            "waitForResponse": false,
            "caption": "Scene activation",
            "name": "Button11",
            "empty": false,
            "msg": "HC2 = Net.FHttp("192.168.100.45") HC2:setBasicAuthentication("admir
            "buttonIcon": 0,
            "favourite": false,
            "main": true
          }
        ]
      },
      {
        "type": "button",
        "elements":
        [
          {
            "id": 2,
            "lua": true,
            "waitForResponse": false,
            "caption": "Deactivate scene",
            "name": "Button21",
            "empty": false,
            "msg": "HC2 = Net.FHttp("192.168.100.45") HC2:setBasicAuthentication("admir
            "buttonIcon": 0,
            "favourite": false,
            "main": false
          }
        ]
      }
    ]
  },
  "actions":
  {
    "pressButton": 1,
    "setSlider": 2,
    "setProperty": 2
  },
  "created": 1405599778,
  "modified": 1405599778,
  "sortOrder": 117
}
```

iOS devices

URL: /api/iosDevices

Methods: GET

Description: Returns a list of added iOS devices and their parameters.

Response: Gets an object containing added iOS devices.

Name	Type	Description
id	Number	iOS device ID
name	String	iOS device name
udid	String	Unique device identifier
push	Boolean	Push notifications status

Example:

```
{
  "id": 1739,
  "name": "iPod (Tom Jones)",
  "udid": "2185A2FE-BBFF-4B83-8BDC-56508C0BF39B",
  "push": true
}
```

VoIP devices

URL: /api/voip

Methods: GET, PUT

Description: Returns a list of VoIP clients associated with Home Center end their parameters.

Response: Gets an array of objects containing configured VoIP clients.

Name	Type	Description
voipDevices	Array	Array of voip devices
id	Number	VoIP client ID
sipDisplayName	String	SIP client name
sipUserID	String	SIP user ID

deviceIcon	String	Type of used icon

Example:

```
{
  "voipDevices":
  [
    {
      "id": 2,
      "sipDisplayName": "admin",
      "sipUserID": "555",
      "deviceIcon": "91"
    }
  ]
}
```

Icons

URL: /api/icons

Methods: GET

Description: Returns a list of icons available in the system.

Response: Gets an object containing interface icons.

Name	Type	Description
id	Number	Icon ID number
deviceType	String	Type of assigned icon
iconSetName	String	Name of icon set

Example:

```
{
  "id": 7,
  "deviceType": "com.fibaro.binarySwitch",
  "iconSetName": "alarm"
}
```

Panels

SMS notifications

URL: /api/panels/sms

Methods: GET, PUT

Description: Returns number of available sms and list of associated phone numbers.

Response: Gets an array of objects containing predefined phone numbers getting sms notifications.

Name	Type	Description
smsCount	Number	Number of available sms

phones	**Array**	**Array of added phones**
id	Number	Phone ID
number	String	Phone number
alarm	Boolean	Alarm association

```
{
 "smsCount": 0,
 "phones":
 [
  {
   "id": 3574,
   "number": "4855555525255",
   "alarm": false
  }
 ]
}
```

Location

URL: /api/panels/location

Methods: GET, DELETE, POST, PUT

Description: Returns a list of predefined locations and their parameters.

Response: Gets an object containing predefined locations.

Name	Type	Description
id	Number	Location id
name	String	Location name
latitude	Number	Location latitude
longitude	Number	Location Longitude
created	Number	Time of creation
modified	Number	Time of last modification

Example:

History panel

URL: /api/panels/history

Methods: GET

Description: Gets an array of objects containing actions stored in the event panel for a specified time period

Response:

Name	Type	Description
id	Number	Event ID
deviceID	Number	Device ID
timestamp	Number	Timestamp value
value	String	Parameter value

Example:

```
[
 {
  "id": 2,
  "deviceID": 269,
  "timestamp": 1409204323,
  "value": "26.0"
 },
 {
  "id": 1,
  "deviceID": 268,
  "timestamp": 1409204323,
  "value": "23.0"
 }
]
```

Notifications panel

URL: /api/panels/notifications

Methods: GET, DELETE, POST, PUT

Description: Returns a list of notifications and their names.

Response: Gets an object containing all stored notifications.

Name	Type	Description
id	Number	Notification ID

name	String	Notification name

Example:

```
{
  "id": 1,
  "name": "not1"
}
```

Heating panel

URL: /api/panels/heating

Methods: GET, DELETE, POST, PUT

Description: Returns a list of heating zones and their settings, such as temperature sets.

Response: Gets objects containing heating panel settings.

Name	**Type**	**Description**
id	Number	Heating zone ID
name	String	Heating zone name

properties:

handTemperature	**Number**	**Manual mode temperature**
handTimestamp	Number	Manual mode timestamp
vacationTemperature	Number	Holiday mode temperature

Example:

```
{
  "id": 1,
  "name": "zone1",
  "properties":
  {
    "handTemperature": 22,
    "handTimestamp": 1405172970,
    "vacationTemperature": 18
  }
}
```

AC panel

URL: /api/panels/cooling

Methods: GET, DELETE, POST, PUT

Description: Returns a list of cooling zones and their settings, such as temperature sets.

Response: Gets objects containing AC panel settings.

Name	Type	Description
id	Number	Heating zone ID
name	String	Heating zone name

properties:

handTemperature	**Number**	**Manual mode temperature**
handTimestamp	Number	Manual mode timestamp
vacationTemperature	Number	Holiday mode temperature

Example:

```
{
  "id": 2,
  "name": "zone1",
  "properties":
  {
    "handTemperature": 20,
    "handTimestamp": 1405172970,
    "vacationTemperature": 24
  }
}
```

Humidity panel

URL: /api/panels/humidity

Methods: GET, DELETE, POST, PUT

Description: Returns a list of humidity zones and their settings, such as humidity levels.

Response: Gets objects containing humidity panel settings.

Name	Type	Description
id	Number	Heating zone ID
name	String	Heating zone name

properties:

handHumidity	Number	Manual mode humidity level
handTimestamp	Number	Manual mode timestamp
vacationHumidity	Number	Holiday mode humidity level

Example:

```
{
  "id": 1,
  "name": "zone1",
  "properties":
  {
    "handHumidity": 30,
    "handTimestamp": 1405172970,
    "vacationHumidity": 50
  }
}
```

Alarm panel

URL: /api/panels/alarm

Methods: GET, DELETE, POST, PUT

Description: Returns a list of alarms and associated devices.

Response: Gets an object containing Alarm panel settings.

Name	Type	Description
id	Number	Alarm ID
name	String	Alarm name

properties:

armDeviceID	Number	Arm Device ID
armStateDeviceID	Number	Arm State Device ID
alarmStateDeviceID	Number	Alarm State Device ID

Drenchers panel

URL: /api/panels/drenchers

Methods: GET, DELETE, POST, PUT

Description: Returns a list of added drenchers and their parameters.

Response: Gets an array of objects containing Drenchers panel settings.

Name	Type	Description
adjustWater	Number	Adjusted water percentage
rainDelay	Number	Rain delay in hours
cycles	Number	Number of cycles per day

drenchers	Array	Array of drenchers
id	Number	Sprinkler ID
name	String	Sprinkler name
mode	String	Active mode
dead	Boolean	Status of being dead
manualTime	Number	Manual time of enable
days	Array	?
cycles	Array	?
nextDrenching	Number	Time of next drenching
state	Boolean	Current status

Example:

```
{
  "adjustWater": 0,
  "rainDelay": 0,
  "cycles": 1,
  "drenchers":
  [
    {
      "id": 1613,
      "name": "1612.0",
      "mode": "off",
      "dead": "false",
      "manualTime": 0,
      "days":
      [
      ],
      "cycles":
      [
      ],
      "nextDrenching": 0,
      "state": "true"
    }
  ]
}
```

Favorite colors

URL: /api/panels/favoriteColors

Methods: GET, DELETE, POST, PUT

Description: Returns a list of favorite colors presets, representing their RGBW values.

Response: Gets an object containing user’s favorite colors.

Name	Type	Description
id	Number	Preset ID
r	Number	Red color value
g	Number	Green color value
b	Number	Blue color value
w	Number	White color value

Example:

```
{
  "id": 1,
  "r": 125,
  "g": 44,
  "b": 125,
  "w": 0
}
```

Fibaro Alarm panel

URL: /api/panels/fibaroAlarm

Methods: GET, PUT

Description: Returns Fibaro Alarm settings list, its properties, conditions, etc.

Response: Gets an array of objects containing Fibaro Alarm panel settings.

Name	**Type**	**Description**
lastAlarmTime	Number	Time of last alarm

triggerActions	**Array**	**Array of triggered actions**
id	Number	Action ID
name	String	Action name
description	String	Action description
time	Number	Time of action
isPredefined	Boolean	Status of being predefined
isActive	Boolean	Status of being active

properties:

conditions	**Array**	**Array of conditions**
type	String	Type of device used in alarm

properties	**Array**	**Arrau of alarm properties**
name	String	Name of property
value	String	Device value

Example:

```
{
  "lastAlarmTime": 0,
  "triggerActions":
  [
    {
      "id": 2,
      "name": "Lights On",
      "description": "Switch on selected lights or all lights in the house.",
      "time": 0,
      "isPredefined": true,
      "isActive": false,
      "properties":
      {
        "conditions":
        [
          {
            "type": "com.fibaro.binarySwitch",
            "properties":
            [
              {
                "name": "isLight",
                "value": "1"
              }
            ]
          },
          {
            "type": "com.fibaro.multilevelSwitch",
            "properties":
            [
              {
                "name": "isLight",
                "value": "1"
              }
            ]
          }
        ]
      }
    }
  ]
}
```

Energy panel

URL: /api/panels/energy

Methods: GET

Description: Returns Energy panel data

Response: ?

Temperature panel

URL: /api/panels/temperature

Methods: GET

Description: Returns Temperature panel data

Response: ?

Events panel

URL: /api/panels/event

IP/api/panels/event?from=xxx&to=yyy

IP – Home Center IP address

xxx – start date timestamp

yyy – end date timestamp

Methods: GET

Description: Returns events history from defined time, device states, state changes, their old and new values, etc.

Response: Gets an object containing events panel settings.

Name	Type	Description
id	Number	Event ID
type	String	Event type
timestamp	Number	Event timestamp
deviceID	Number	Device ID
deviceType	String	Device type
propertyName	String	Device property name
oldValue	Number	Old device value
newValue	Number	New device value

Example:

```
{
  "id": 8126,
  "type": "DEVICE_EVENT",
  "timestamp": 1404723546,
  "deviceID": 1701,
  "deviceType": "com.fibaro.temperatureSensor",
  "propertyName": "value",
  "oldValue": 28.6,
  "newValue": 26.7
}
```

Plugins

Plugins types

URL: /api/plugins/types

Methods: GET

Description: Returns a list of plugins divided into categories and their parameters.

Response: Gets an array of objects containing all available plugins.

Name	**Type**	**Description**
types	Array	Array of plugin’s types
category	Number	Number of plugin category

plugins	**Array**	**Array of plugins**
type	String	Type of plugin
name	String	Name of plugin
description	String	Plugin description
user	String	Plugin creator
compatibility	Array	Plugin compatibility
predefined	Boolean	Predefinition status
version	String	Plugin version
url	String	Plugin URL
installed	Boolean	Status of being installed

Example:

```
{
 "types":
 [
  {
   "category": 0,
   "plugins":
   [
    {
     "type": "com.fibaro.dscAlarm",
     "name": "DSC Alarm",
     "description": "Add and configure Satel control panel, check states of inputs, outp
     "user": "Fibar Group Sp. z o.o.",
     "compatibility":
     [
      "iPad",
      "iPhone",
      "AndroidPhone",
      "config"
     ],
     "predefined": true,
     "version": "1.0",
     "url": "panels/external-alarm.html?type=com.fibaro.dscAlarm",
     "installed": true
    }
   ],
   "installed": 6
  }
 ]
}
```

Plugins installed

URL: /api/plugins/installed

Methods: GET

Description: Returns a list of installed plugins, their names and predefinition status.

Response: Gets an object containing all installed plugins.

Name	Type	Description
name	String	Name of plugin
predefined	Boolean	Plugin predefinition status

Example:

```
{
  "name": "com.fibaro.satelAlarm",
  "predefined": true
}
```

Other

Login status

URL: /api/loginStatus

Methods: GET, PUT

Description: Returns a list of parameters related to user’s login, such as status, username or type of currently logged in user.

Response: Gets an object containing current login status.

Name	Type	Description
status	Boolean	Login status
userID	Number	ID of logged in user
username	String	Username
type	String	Type of logged in user

Example:

```
{
  "status": true,
  "userID": 2,
  "username": "admin",
  "type": "superuser"
}
```

Password reminder

URL: /api/passwordForgotten

Methods: GET

Description: Returns a password to your account sending it by e-mail.

Response: Gets user’s password in case of forgetting.

Correct response:

```
{
  "status": "OK"
}
```

Example:

IP/api/passwordForgotten?login=xxx

IP – your Home Center IP address

xxx – your login

Refresh states

URL: /api/refreshStates

Methods: GET

Description: Returns refreshment details and performed changes.

Response: Gets an object containing detailed last status refreshments.

Name	Type	Description
status	String	Current status
last	Number	Last refresh
date	String	Status date
timestamp	Number	Timestamp
logs	Array	Detailed logs
changes	Array	Changes details

Example:

```
{
  "status": "IDLE",
  "last": 38,
  "date": "12:58 | 21.07.2014",
  "timestamp": 1405940300,
  "logs":
  [
  ],
  "changes":
  [
  ]
}
```

Network discovery

URL: /api/networkDiscovery/arp

Methods: PUT

Description: Find IP and MAC physical addresses for specified network

Response:

Debug scene

URL: /api/scene/ID/debugMessages

ID – scene ID

Methods: GET

Description: Returns messages displayed by scene of given ID.

Response: Gets an array of objects containing messages displayed during scene debug.

Name	Type	Description
timestamp	Number	Scene timestamp
type	String	Type of message
txt	String	Debugged text

Example:

```
[
 {
  "timestamp": 1406036436,
  "type": "DEBUG",
  "txt": "Helloworld!"
 }
]
```

Call Action

URL: /api/devices/deviceID/action/actionName

deviceID – ID of an existing device

actionName – Name of an action

Methods: POST

Description: Trigger an action of the specified device

Response:

Weather status

URL: /api/weather

Methods: GET

Description: Returns a list of current and previous weather parameters downloaded from weather.yahoo.com for your location.

Response: Gets an object containing weather data.

Name	Type	Description
ConditionCode	String	Current weather condition code
Humidity	String	Current humidity level
PreviousConditionCode	String	Previous weather condition code
PreviousHumidity	String	Previous humidity level
PreviousTemperature	String	Previous temperature
PreviousWeatherConditionConverted	String	Previous weather condition
PreviousWind	String	Previous wind speed
Temperature	String	Current temperature
WeatherCondition	String	Current weather condition
WeatherConditionConverted	String	Current weather condition
Wind	String	Current wind speed
saveLogs	String	Number of log
TemperatureUnit	String	Selected temperature unit

Example:

```
{
  "ConditionCode": "34",
  "Humidity": "45",
  "PreviousConditionCode": "30",
  "PreviousHumidity": "48",
  "PreviousTemperature": "27",
  "PreviousWeatherConditionConverted": "cloudy",
  "PreviousWind": "24.14",
  "Temperature": "28",
  "WeatherCondition": "rain",
  "WeatherConditionConverted": "clear",
  "Wind": "27.36",
  "saveLogs": "1",
  "TemperatureUnit": "C"
}
```

Diagnostics

URL: /api/diagnostics

Methods: GET

Description: Returns a list of system parameters, such as memory usage, cpu load, etc.

Response: Gets an array of objects containing system diagnostic data.

Name	**Type**	**Description**
memory	Number	Percentage of free RAM memory

storage	**Array**	**Storage array**
name	String	Disk name
used	Number	Percentage of used space

cpuLoad	**Array**	**Array of CPU load**
user	String	Percentage of CPU utilization that occurred while executing at the user level
nice	String	Percentage of CPU utilization that occurred while executing at the user level with nice priority

system	String	Percentage of CPU utilization that occurred while executing at the system level
idle	String	Percentage of time that the CPU was idle and the system did not have an outstanding disk I/O request.

Example:

```
{
  "memory": 62,
  "storage":
  [
    {
      "name": "system",
      "used": 40
    },
    {
      "name": "recovery",
      "used": 22
    }
  ],
  "cpuLoad":
  [
    {
      "cpu0":
      {
        "user": "291016",
        "nice": "116",
        "system": "237122",
        "idle": "43946780"
      }
    },
    {
      "cpu1":
      {
        "user": "316296",
        "nice": "324",
        "system": "286170",
        "idle": "45617346"
      }
    }
  ]
}
```