



DECEMBER 2020

ABB-free@home® – local REST API

Webinar – Competence Center Europe – Smart Buildings

Thorsten Reibel, Jürgen Schilder, Stefan Grosse, Martin Wichary & Olaf Stutzenberger

Agenda

Introduction

API – Application Programming Interface

ABB Developer Portal

Installation

Swagger: API Documentation & Design Tool

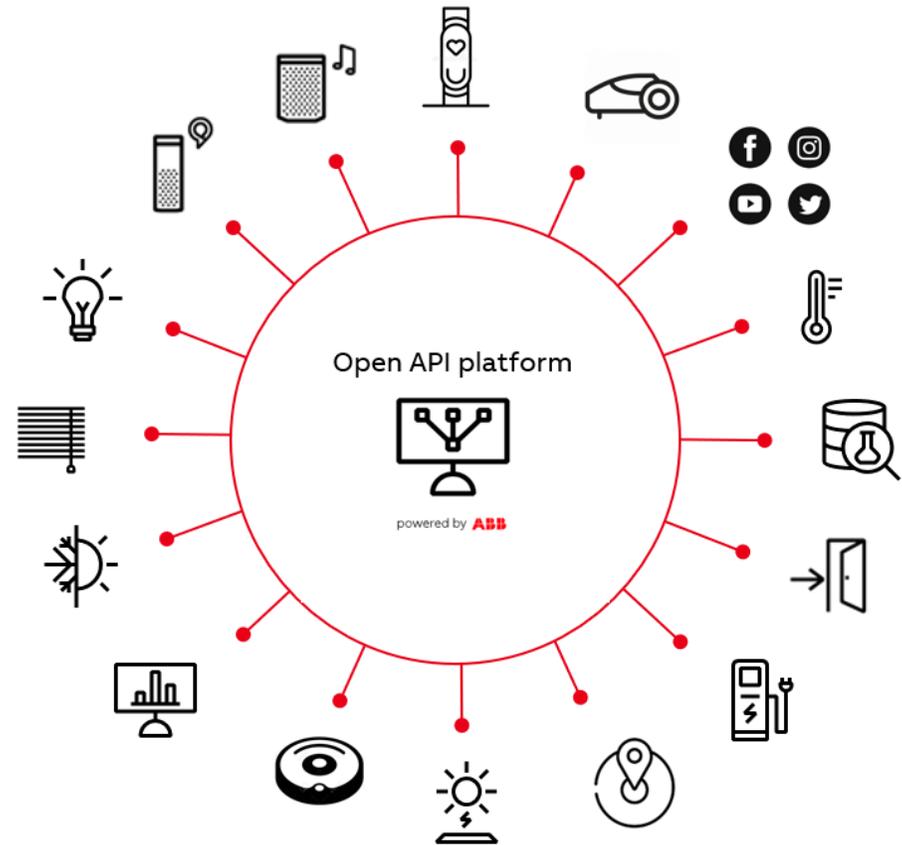


ABB-free@home[®] – local REST API

Introduction

ABB-free@home® – local REST API

Introduction

Motivation

- ABB-free@home® transforms the house or apartment into an intelligent home. Whether lighting, shading, room temperature, door communication or entertainment – comfort, safety and efficiency can be controlled intuitively
- The Smarter Home Developer Program has been created, so that this promise can be extended to other devices and services in the household
- This will give our customers the possibility to create a holistic smart home solution
- The “free@home local REST API“ will be the next interface extending our open systems approach

Smarter Home Developer Program



New perspectives on networking for external developers. With the "Smarter Home Developer Program" ABB opens its systems for developers and partners.

ABB-free@home® – local REST API

Introduction

Goals

- Customers are able to interconnect additional systems and solutions with free@home and to use them in one ecosystem
 - free@home can be the central system and can be extended with additional devices or services from 3rd party or even ABB offerings
 - free@home can be connected to other central systems inside a building (like BMS)
- The local REST API only works on premise and inside the same building
 - No data leaving the building
 - Limited to be used inside the same building
- Giving our customers the feeling of one overall “smart home” regardless of source and matter

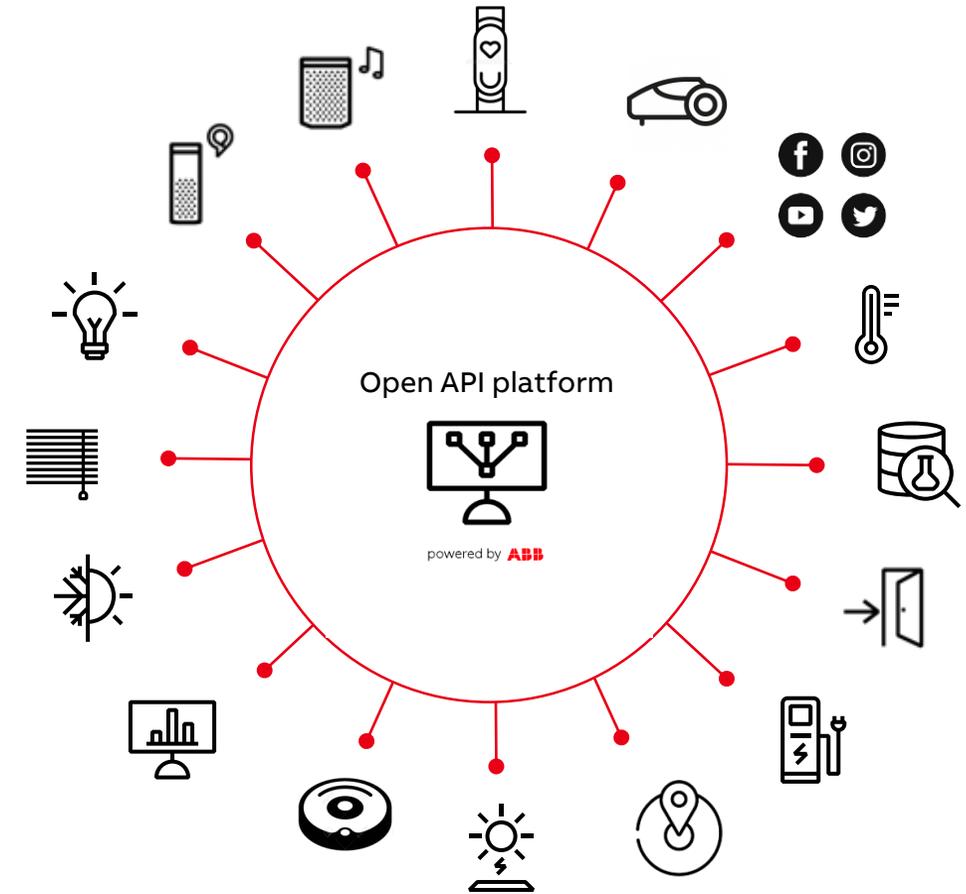


ABB-free@home® – local REST API

Introduction – USPs

Seamless integration into one system



Seamless integration into a professional installed smart home system

Add 3rd party systems and functions into free@home to sensors, panel displays, scenes and applications like timers or actions

Protecting customers' data



Easy to expand with other solutions due to open and standardized interface

Suitable for all markets globally where free@home is available

Targeting different building segments



Suitable for single family house, where tech-savvy end-user will integrate 3rd party solutions into free@home

Suitable for pre-fitted apartments, that come with free@home and other systems

(There must be a validation if the free@home REST API doesn't fit better to the needs in some cases)

Easy to expand



Easy to expand with other solutions due to open and standardized interface

Suitable for all markets globally where free@home is available

ABB-free@home® – local REST API

Introduction

Example use cases for end customers

Due to the local REST API tech savvy user or developer can develop own applications and can bring different solutions and services together with free@home

The integration of free@home into open Smart Home Hubs offers the possibility to enable many different use cases also to other end customers

Examples:

- Bring vacuum cleaner into free@home
- Control your smart robot mower
- Write and persist data inside a dashboard or google sheet

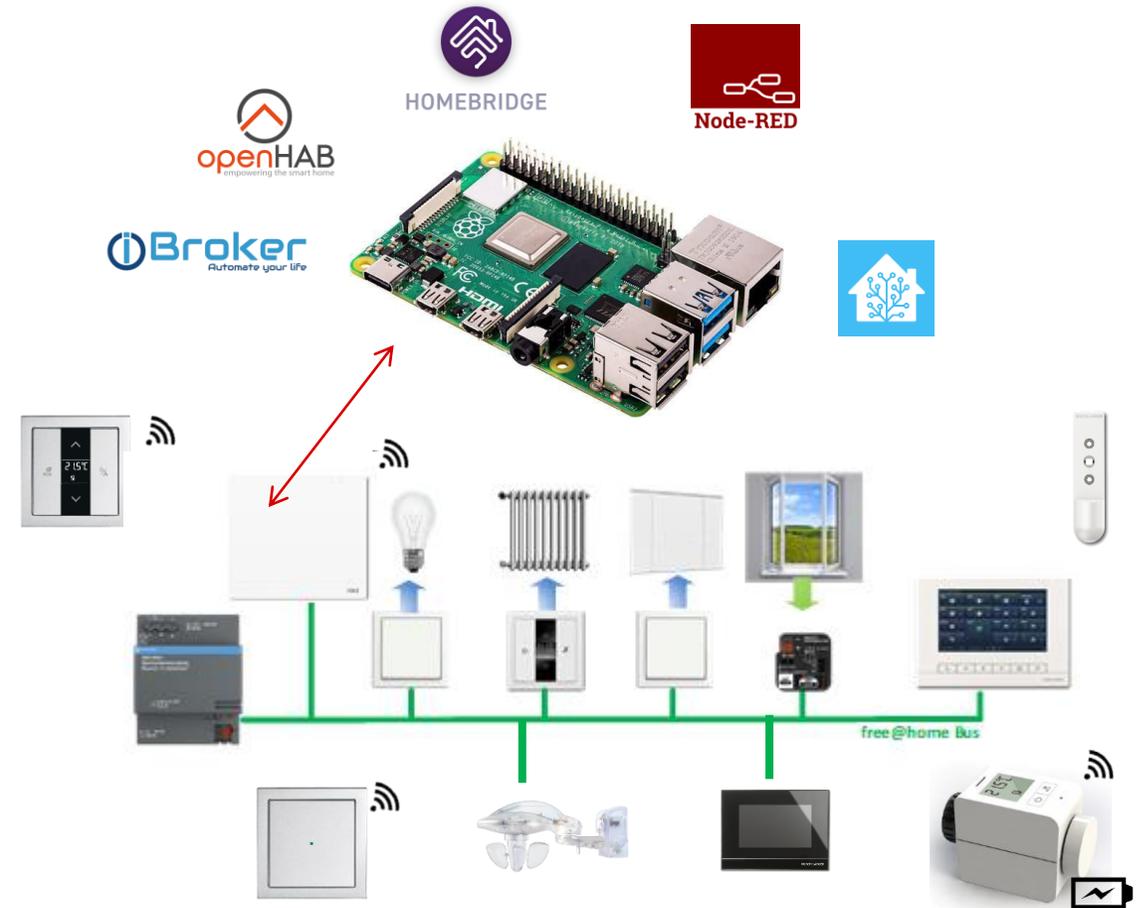


ABB-free@home® – local REST API

Introduction

Example use cases for professional partners

The integration of free@home into BMS offers the possibility to enable many different use cases

In this B2B cases it depends if the BMS manufacturer decides to integrate towards the local or cloud API of free@home because of different pros and contras

Examples:

- Hotels with multiple free@home systems installed
- Multi tenant apartment houses
- free@home and Energy Solutions like EV Charger and Solar

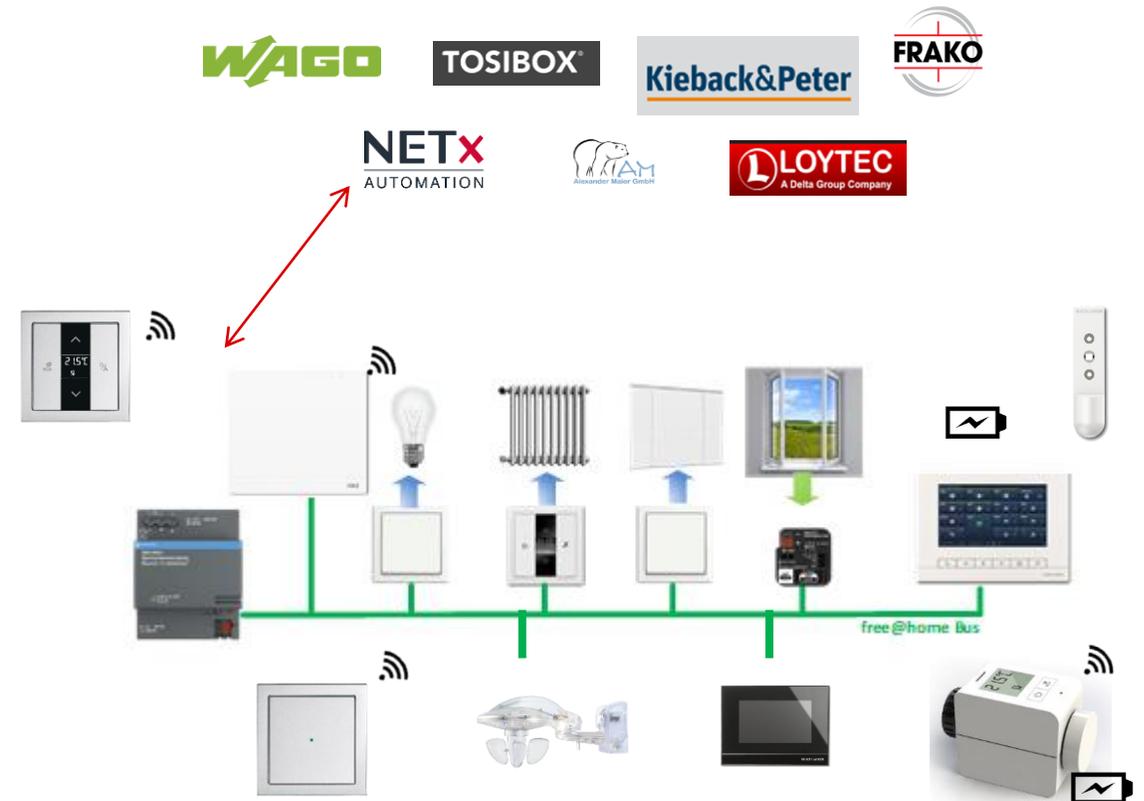


ABB-free@home[®] – local REST API

API – Application Programming Interface

ABB-free@home® – local REST API

API – Application Programming Interface

Questions

- APIs. What are they?
- What can I do with an API?
- Do I need to understand them?
- What are my customers constantly talking about?
- Do I really need to monitor APIs?
- How to get started?



ABB-free@home® – local REST API

API – Application Programming Interface

General Information

- **APIs** allow different systems to communicate with each other. It lets a mobile app talk to a social network, or a payments service work the same way across the web, TV, mobile and other platforms
- **REST** is a Representational State Transfer. It's a stateless, client-server, cacheable **communication protocol** that usually uses HTTP

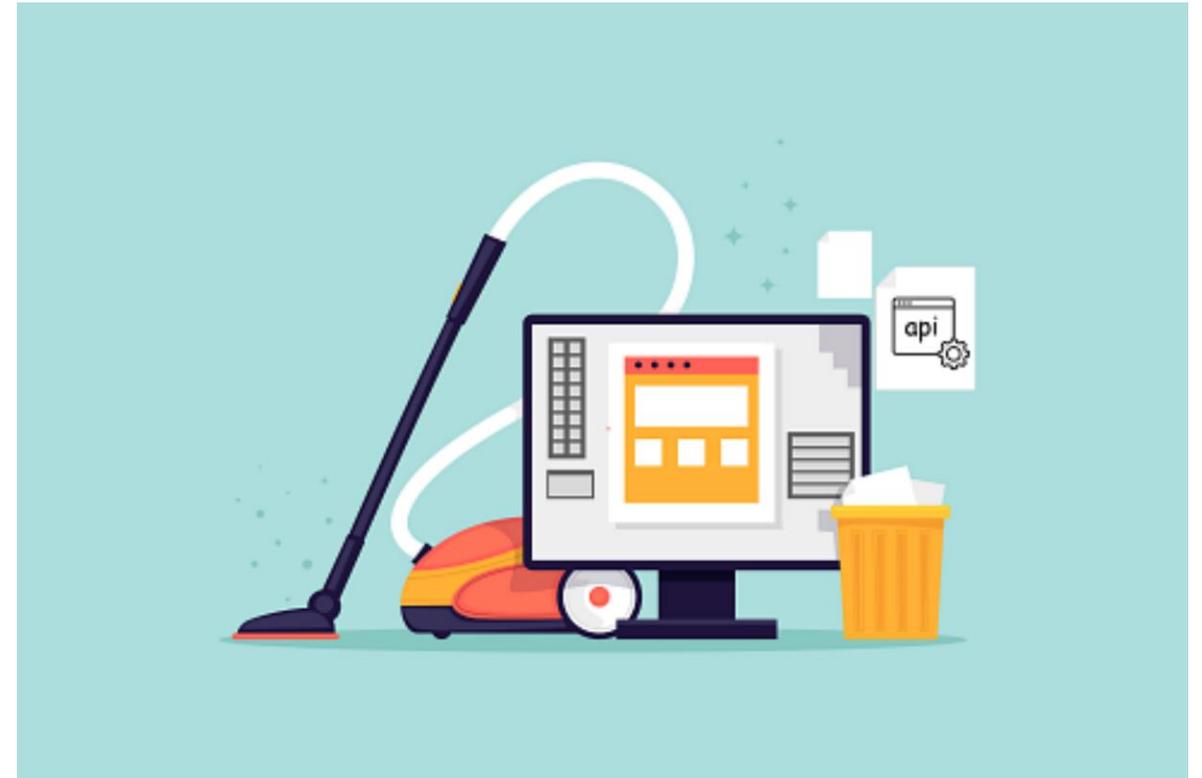


ABB-free@home® – local REST API

API – Application Programming Interface

Cloud vs. Local API

Cloud API / Web API

- Cloud-based API integrated in MyBuilding
- Controlled data-exchange with free@home to extend the Smart Home functionality
- Ideal for making Smart Home services available for free@home

Local API

- New interface for application programming (API) for ABB-free@home®

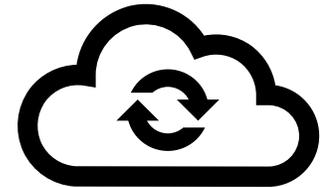
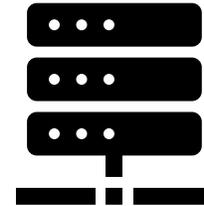


ABB-free@home® – local REST API

API – Application Programming Interface

Internal vs. Public vs. External API

Internal API / Private API

- A private API is intended for internal use only
- Example: free@home connection to MyBuildings

Public API / Partner API

- The API is shared with specific business partners
- This allows you to tap additional sources of income without compromising quality
- Example: free@home connection to HomeConnect, Miele, Sonos, HUE, etc.

External API / Open API

- This API is available to everyone
- This allows third parties to develop apps to interact with your API and opens up opportunities for innovation
- free@home open for end customers

THERE ARE THREE MAIN TYPES OF APIS



Open

Open APIs allow companies to **publicly expose information and functionalities** of one or various systems and applications to **third parties** that do not necessarily have a business relationship with them.

Advantages:

- Delegated R&D
- Increased reach, traffic
- New revenue stream



Partner

Partner APIs are used to **facilitate** communication and integration of software **between a company and its business partners**

Advantages:

- Value-added service
- Up sell
- Must have for business partners



Private

Private APIs are used **internally** to facilitate the **integration** of different applications and systems used by a company.

Advantages:

- Rationalized infrastructure
- Reduced costs
- Increase flexibility: “real-time” business
- Improved internal operations

ABB-free@home® – local REST API

API – Application Programming Interface

Situation today

- free@home is already working with API solutions
- The usage of API solutions is not new
- But: the available solutions were limited to the integrated solutions of ABB and their partners
- Now the end customer can generate new use cases and integrations on his own

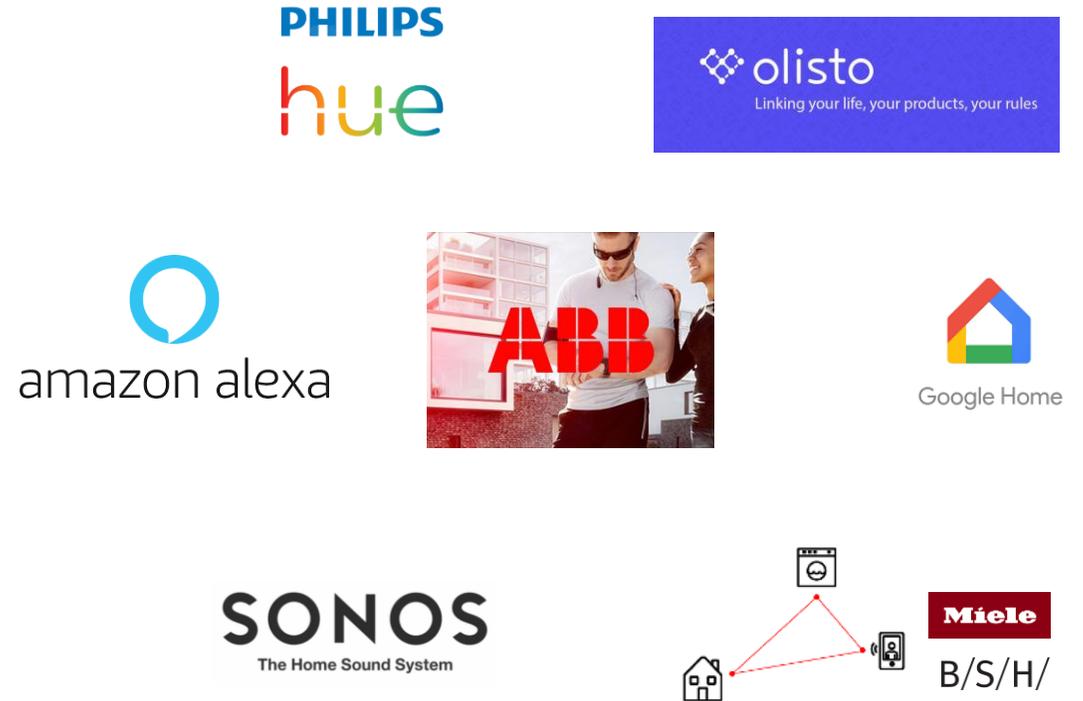


ABB-free@home[®] – local REST API

ABB Developer Portal

ABB-free@home® – local REST API

ABB Developer Portal

Introduction

- Whether you are looking to integrate our APIs into your application, website or service, you can find out how to do it at our ABB Developer Portal
- Our APIs enable you to create value added experiences with Smart Home and Smart Building solutions
 - Documentations
 - Tutorials
 - APIs
 - Products
 - Support
 - FAQ
- <https://developer.eu.mybuildings.abb.com/>

ABB Developer Portal for Smart Buildings

HOME DOCUMENTATION TUTORIALS APIS PRODUCTS ISSUES FAQ SIGN IN

Welcome to the ABB Developer Portal!

Feel free to discover and learn about our APIs for Smarter Homes and Smarter Buildings. Just sign up and start developing and prototyping right away.

Develop the future of Smart Buildings with us

Whether you are looking to integrate our API into your application, website or service, you can find out how to do it here! Our APIs enable you to create value added experiences with Smart Home and Smart Building solutions. Eager to get started? Scroll down to find out how!

Door communication, Blinds, Heating, Light, Air-conditioning, Voice control, Media, Energy, Home Appliance

ABB-free@home® – local REST API

ABB Developer Portal

First steps

- To use the Cloud API an ABB MyBuildings account is needed
- Register on MyBuildings and send a request on the website to get access to the API

Become an ABB Developer Partner in 4 steps

 Sign Up	 Request	 Subscribe	 Try
Want to use our API? Create a myBuildings Portal account. If you already have one, move on to step two.	Ready to start development? Request for your individual credentials and developer role to access our APIs.	Subscribe to one of our APIs. After approval you are able to start trying the API right away.	Try out all our cloud-based APIs simply inside this Portal to get familiar with them.
Sign up	Request	Subscribe	Try out

ABB-free@home® – local REST API

ABB Developer Portal

Documentations

- Interested users will find many documentations, tutorials and examples under the tab “Documentation”
- Details step-by-step guides for all available APIs

Documentation

To start coding right away you will find below detailed documentation about our cloud infrastructure, how to set up first OAuth SSO (single sign-on) flow to let users connect to your application and use your service. Additionally you will find dedicated information about our cloud-ready systems like free@home and the KNX IoT Dashboard Server. Enjoy!

ABB Developer Portal and cloud authorization

- [How to get started on the ABB Developer Portal](#)
- [OAuth2 Authentication Flow](#)
- [OAuth2 sample](#)

ABB free@home cloud API

- [free@home cloud API](#)
 - [Prerequisites for using the free@home cloud API](#)
 - [Free@home cloud API concepts](#)
 - [Free@home cloud API samples](#)
 - [Free@home cloud API reference](#)

ABB free@home local API

- [free@home local API](#)
 - [Prerequisites for using the free@home local API](#)
 - [Free@home local API concepts](#)
 - [Free@home local API samples](#)
 - [Free@home local API reference](#)

ABB KNX IoT Dashboard Server

- [How to get started with the KNX i-bus® IoT Dashboard API](#)

ABB-free@home[®] – local REST API

Installation

ABB-free@home® – local REST API

Installation

First steps

- The free@home local API is capable accessing the free@home system access points of end users when they are connected to the local network. For this, a few prerequisites must be met by the end user
- This API is capable of controlling a SysAP in the local network
- The end user needs an ABB-free@home smart access point with version 2.6.0 or later
- The free@home next App for a mobile device
- The end user must enable the local API in the SysAP settings of the free@home next app, found under:
More -> Settings -> Local API

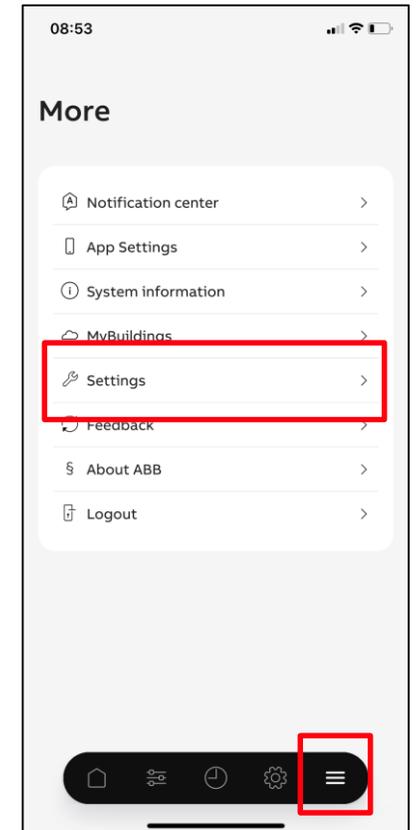
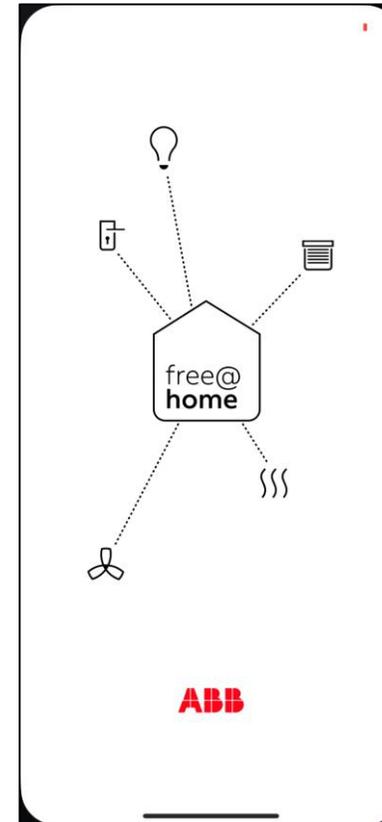


ABB-free@home® – local REST API

Installation

First steps

- The free@home local API is capable accessing the free@home system access points of end users when they are connected to the local network. For this, a few prerequisites must be met by the end user
- This API is capable of controlling a SysAP in the local network
- The end user needs an ABB-free@home smart access point with version 2.6.0 or later
- The free@home next App for a mobile device
- The end user must enable the local API in the SysAP settings of the free@home next app, found under:
More -> Settings -> Local API

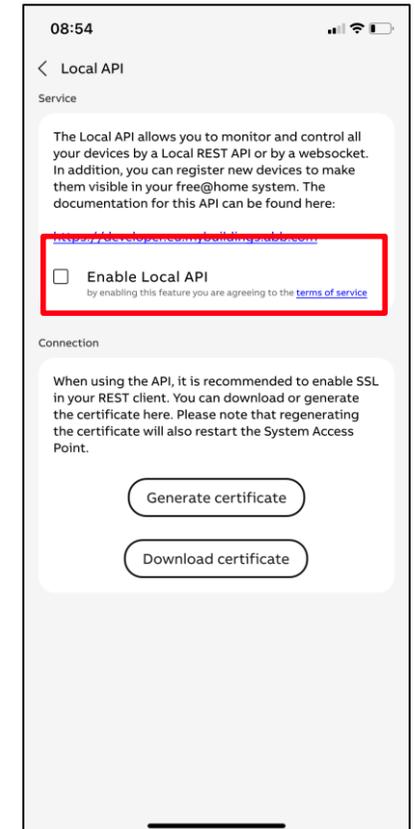
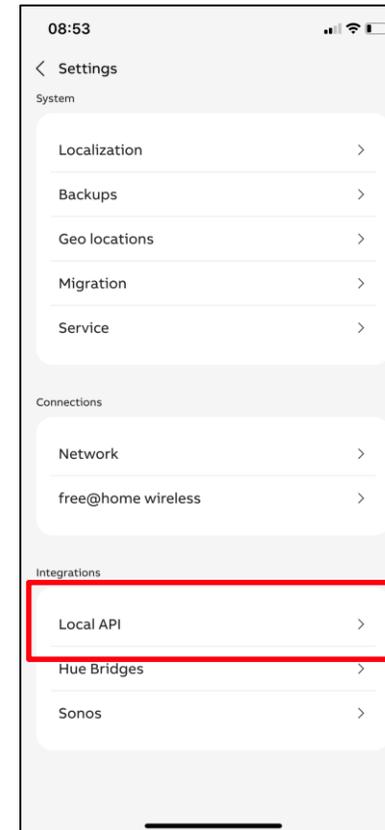


ABB-free@home® – local REST API

Installation

First steps

- The free@home local API is capable accessing the free@home system access points of end users when they are connected to the local network. For this, a few prerequisites must be met by the end user
- This API is capable of controlling a SysAP in the local network
- The end user needs an ABB-free@home smart access point with version 2.6.0 or later
- The free@home next App for a mobile device
- The end user must enable the local API in the SysAP settings of the free@home next app, found under:
More -> Settings -> Local API

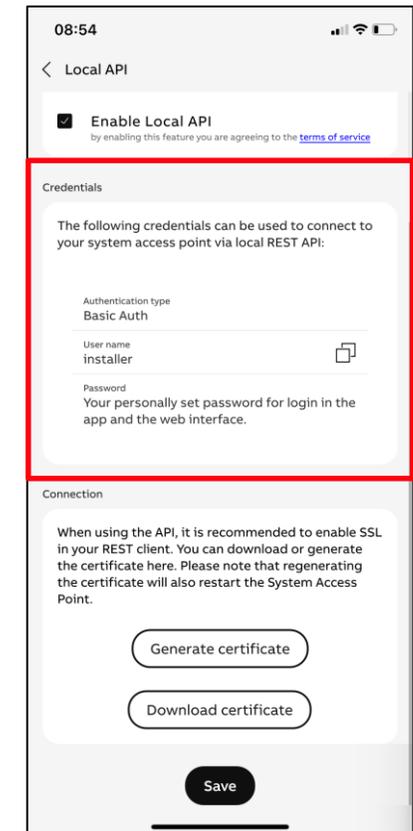
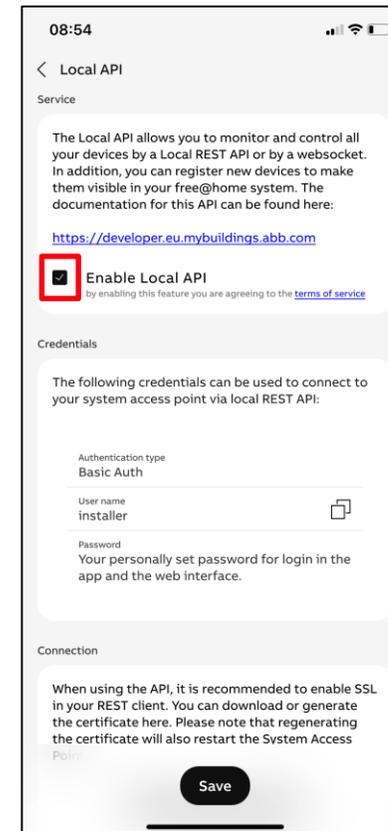


ABB-free@home® – local REST API

Installation

First steps

- SSL certificate to generate an encrypted communication between free@home and your Raspberry Pi
- The generated certificate must be integrated into Node-RED
- Website will be displayed as HTTPS (Hyper Text Transfer Protocol Secure)

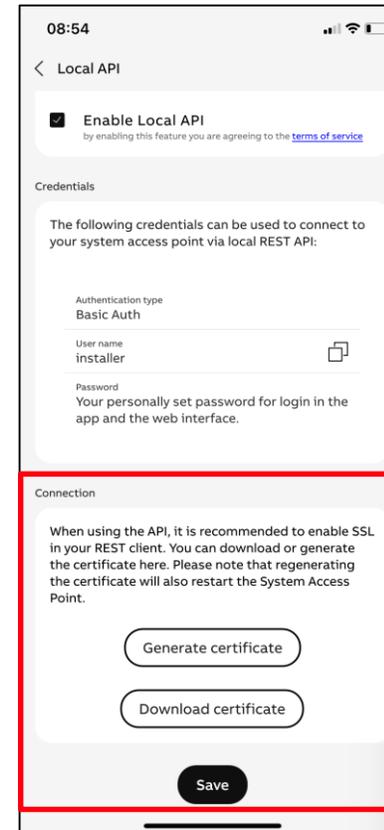


ABB-free@home[®] – local REST API

Testing the API

ABB-free@home® – local REST API

Testing the API

Web interface

- To test the API, the calls can be triggered via a web interface. This is available at <http://{IP-Address of the System Access Point}/swagger>

The screenshot displays the Swagger UI for the free@home API. At the top, it identifies the API as 'free@home API v1 OAS3'. Below this, there is a 'Servers' dropdown menu currently set to '/fhapi/v1'. The main content area is divided into two sections: 'api' (Main API) and 'default'. The 'api' section lists several endpoints:

- GET /api/rest/configuration: Get configuration
- GET /api/rest/devicelist: Get devicelist
- GET /api/rest/device/{sysap}/{device}: Get device
- GET /api/rest/datapoint/{sysap}/{device}.{channel}.{datapoint}: Get datapoint value
- PUT /api/rest/datapoint/{sysap}/{device}.{channel}.{datapoint}: Set datapoint value
- PUT /api/rest/virtualdevice/{sysap}/{serial}: Create virtual device

The 'default' section contains one endpoint:

- GET /api/ws: Websocket connection

Each endpoint entry includes a colored button indicating the HTTP method (blue for GET, orange for PUT) and a lock icon indicating that the endpoint is secured. An 'Authorize' button with a lock icon is located in the top right corner of the interface, and a larger, more prominent 'Authorize' button with a lock icon is shown above it, connected by an arrow.

ABB-free@home® – local REST API

Testing the API

Web interface

- Enter the username and the password and select “Authorize” to authorize

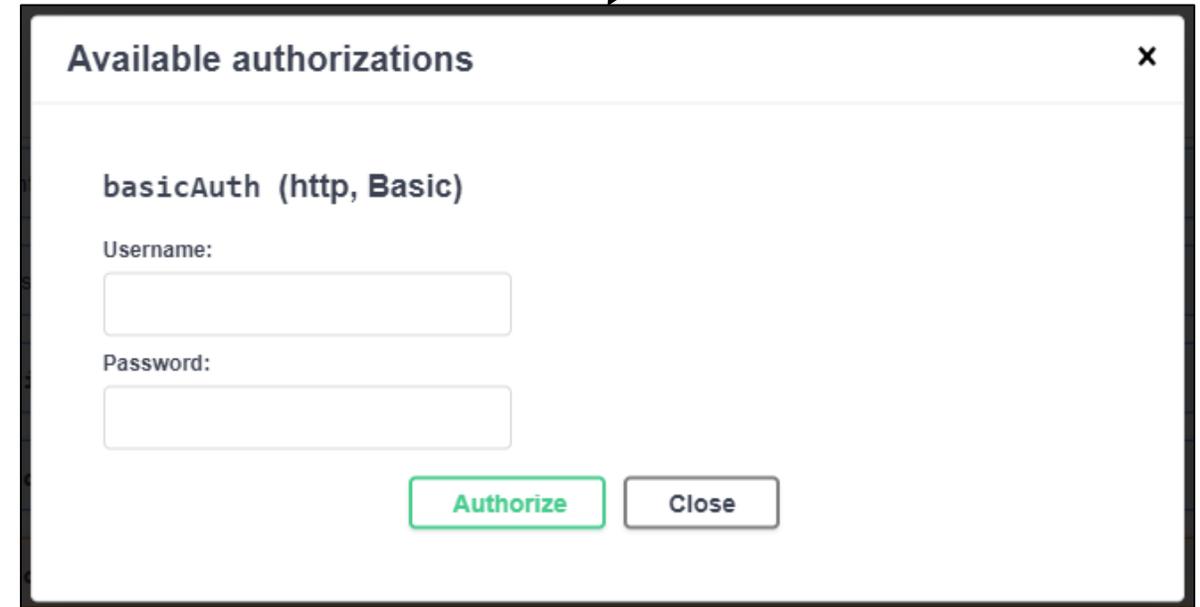
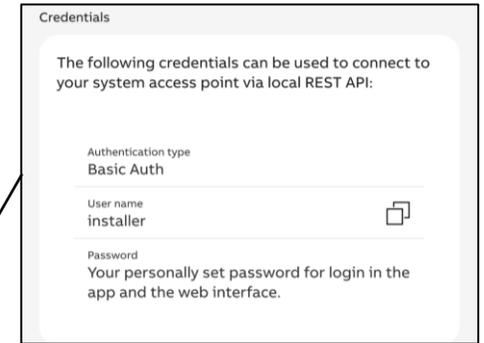


ABB-free@home® – local REST API

Testing the API

Web interface

- The status will be indicated as “Authorized” now
- You can use the available “get” and “put” commands in the web interface now

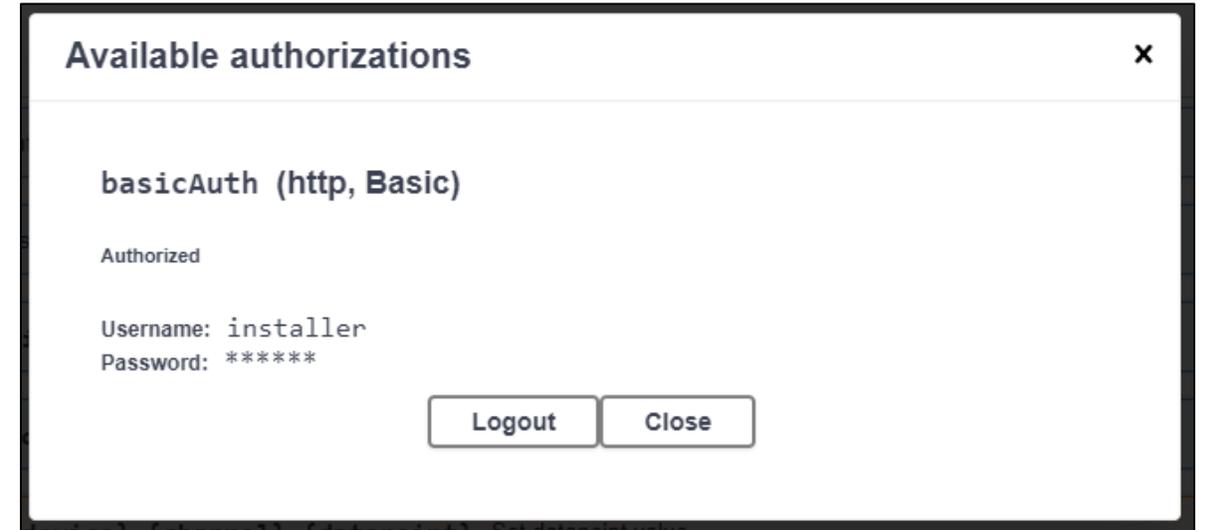
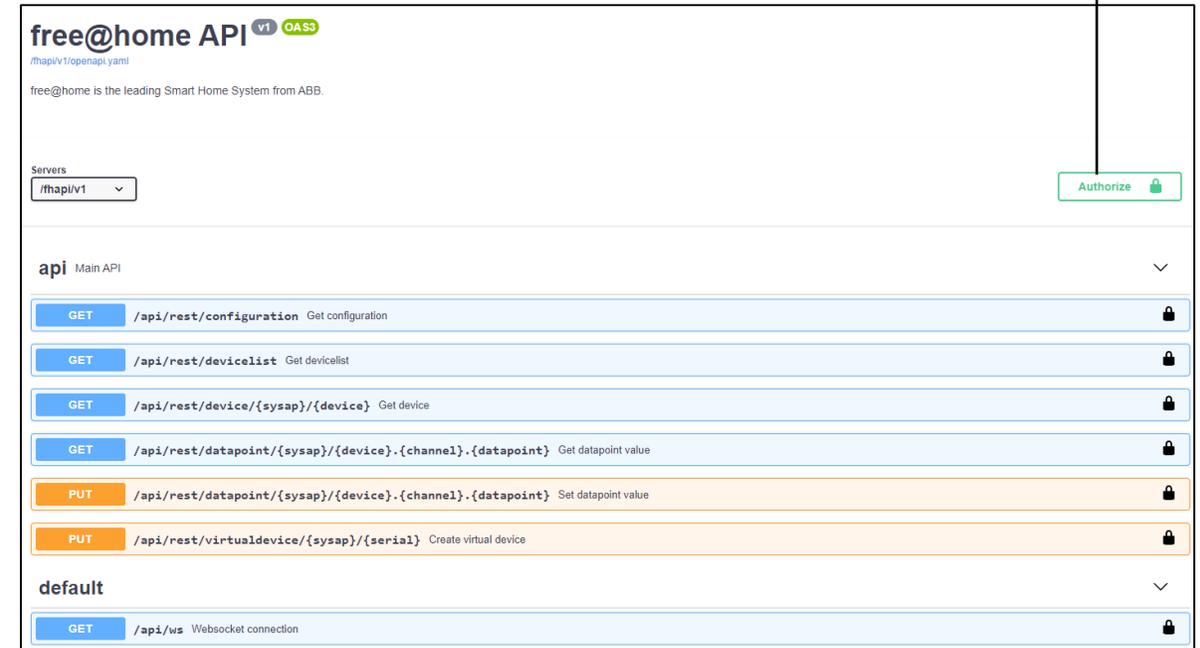


ABB-free@home® – local REST API

Testing the API

Web interface

- The status will be indicated as “Authorized” now
- You can use the available “get” and “put” commands in the web interface now
- To disconnect click again on “Authorize” and “Logout”



The screenshot shows the "free@home API" web interface. At the top, it says "free@home API v1 OAS3" and provides the URL "/fhapi/v1/openapi.yaml". Below this, it states "free@home is the leading Smart Home System from ABB." There is a "Servers" dropdown menu currently set to "/fhapi/v1". In the top right corner of the interface, there is a green "Authorize" button with a padlock icon. The main content area is divided into two sections: "api Main API" and "default". The "api Main API" section contains a list of REST API endpoints with their methods and descriptions, each with a lock icon indicating it is protected. The endpoints are: GET /api/rest/configuration (Get configuration), GET /api/rest/devicelist (Get devicelist), GET /api/rest/device/{sysap}/{device} (Get device), GET /api/rest/datapoint/{sysap}/{device}.{channel}.{datapoint} (Get datapoint value), PUT /api/rest/datapoint/{sysap}/{device}.{channel}.{datapoint} (Set datapoint value), and PUT /api/rest/virtualdevice/{sysap}/{serial} (Create virtual device). The "default" section contains one endpoint: GET /api/ws (Websocket connection).

ABB-free@home® – local REST API

Testing the API

Main API – Commands

- “Get configuration”: Get configuration for all user registered System Access Points, this includes the schema for all devices, channels and data points, the floorplan and current user information
- “Get devicelist”: Get list of devices for all System Access Points
- “Get device”: Get configuration information for given device
- “Get datapoint value”: Get the current value of a given datapoint
- “Set datapoint value”: Set a new value for a given datapoint
- “Create virtual device”: Create a virtual device inside free@home

api Main API	
GET	/api/rest/configuration Get configuration
GET	/api/rest/devicelist Get devicelist
GET	/api/rest/device/{sysap}/{device} Get device
GET	/api/rest/datapoint/{sysap}/{device}.{channel}.{datapoint} Get datapoint value
PUT	/api/rest/datapoint/{sysap}/{device}.{channel}.{datapoint} Set datapoint value
PUT	/api/rest/virtualdevice/{sysap}/{serial} Create virtual device

ABB-free@home® – local REST API

Testing the API

Main API – Get configuration

- All functions can be opened by clicking on the tab
- A complete description of the function with all linked parameters and possible responses will be displayed
- Function can be tested by clicking on “Try it out”

api Main API

GET /api/rest/configuration Get configuration

Get configuration for all user registered System Access Points, this includes the schema for all devices, channels and data points, the floorplan and current user information. The response body is a JSON object that uses the SysAP UUID of each SysAP of the requesting user account as key. The configuration of that SysAP is the corresponding value and is again a JSON object. You can find further description and an example of the returned data model in the [documentation](#) and an example of this request in the [samples](#).

Parameters

No parameters

Responses

Code	Description	Links
200	Configuration	No links

Media type: application/json

Controls Accept header.

ABB-free@home® – local REST API

Testing the API

Main API – Get configuration

- All functions can be opened by clicking on the tab
- A complete description of the function with all linked parameters and possible responses will be displayed
- Function can be tested by clicking on “Try it out”
- To start the application, click on “Execute”

api Main API

GET /api/rest/configuration Get configuration

Get configuration for all user registered System Access Points, this includes the schema for all devices, channels and data points, the floorplan and current user information. The response body is a JSON object that uses the SysAP UUID of each SysAP of the requesting user account as key. The configuration of that SysAP is the corresponding value and is again a JSON object. You can find further description and an example of the returned data model in the [documentation](#) and an example of this request in the [samples](#).

Parameters

No parameters

Cancel

Execute

ABB-free@home® – local REST API

Testing the API

Main API – Get configuration

- All functions can be opened by clicking on the tab
- A complete description of the function with all linked parameters and possible responses will be displayed
- Function can be tested by clicking on “Try it out”
- To start the application, click on “Execute”
- The response will be displayed below
- The “Get configuration” function will generate the complete configuration of your free@home system

Server response	
Code	Details
200	<p>Response body</p> <pre>{ "00000000-0000-0000-0000-000000000000": { "connectionState": true, "sysapName": "Martins SysAP", "devices": { "ABB700000000": { "displayName": "test", "unresponsive": false, "channels": { "ch0000": { "displayName": "SysAP", "functionID": "25", "inputs": { "idp0000": { "pairingID": 20, "value": "0" } }, "outputs": { "odp0000": { "pairingID": 18, "value": "0" } } } } } } } }</pre>

ABB-free@home® – local REST API

Testing the API

Main API – Get devicelist

- Get list of devices for all System Access Points
- This endpoint is similar to the /api/rest/configuration endpoint, but only provides access to the list of Device IDs that are known by each SysAP, not their corresponding configuration

```
Server response
Code      Details
200      Response body
{
  "00000000-0000-0000-0000-000000000000": [
    "ABB700000000",
    "FFFF48020002",
    "FFFF4A000001",
    "ABB700D963C3",
    "FFFF4A000002",
    "ABB700CBB094",
    "ABB700CBB0B0",
    "FFFF4A010001",
    "BEED3F4B0001",
    "FFFF4A010002",
    "ABB700CBB07D",
    "A000CFA66C46",
    "FFFF48000001",
    "FFFF48000002",
    "FFFF48000003",
    "FFFF48000004",
    "A000EA738144",
    "FFFF48000005",
    "ABB700CBB085",
    "FFFF48020001"
  ]
}
```

ABB-free@home® – local REST API

Testing the API

Main API – Get device

- Get configuration information for given device
- This endpoint is similar to the `/api/rest/configuration` endpoint, but except for returning the full configuration of all SysAPs, it returns the configuration of a single device in a single SysAP only and therefore is much faster and requires less bandwidth
- The response body is a JSON object that maps the (single) SysAP UUID to an object that contains "devices" object (only) which in turn holds the specified device object (only)

GET `/api/rest/device/{sysap}/{device}` Get device

Get configuration information for given device.

This endpoint is similar to the `/api/rest/configuration` endpoint, but except for returning the full configuration of all SysAPs, it returns the configuration of a single device in a single SysAP only and therefore is much faster and requires less bandwidth.

The response body is a JSON object that maps the (single) SysAP UUID to an object that contains "devices" object (only) which in turn holds the specified device object (only).

See also the [documentation](#) for the meaning of the contents of the device object.

Parameters Cancel

Name	Description
sysap * required string (path) pattern: <code>^[0-9]{8}-[0-9]{4}-[0-9]{4}-[0-9]{12}\$</code>	SysAP UUID
<input type="text" value="00000000-0000-0000-0000-000000000000"/>	
device * required string (path) pattern: <code>^[0-9A-Fa-f]{12}\$</code>	Device Serial
<input type="text" value="BEED3F4B0001"/>	

Execute

ABB-free@home® – local REST API

Testing the API

Main API – Get device

- Get configuration information for given device
- This endpoint is similar to the `/api/rest/configuration` endpoint, but except for returning the full configuration of all SysAPs, it returns the configuration of a single device in a single SysAP only and therefore is much faster and requires less bandwidth
- The response body is a JSON object that maps the (single) SysAP UUID to an object that contains "devices" object (only) which in turn holds the specified device object (only)

```
Server response
Code    Details
200
Response body
{
  "00000000-0000-0000-0000-000000000000": {
    "devices": {
      "BEED3F4B0001": {
        "floor": "01",
        "room": "02",
        "displayName": "test",
        "unresponsive": false,
        "channels": {
          "ch0000": {
            "displayName": "Stehlampe",
            "functionID": "2f",
            "inputs": {
              "idp0000": {
                "pairingID": 1,
                "value": "0"
              },
              "idp0001": {
                "pairingID": 16,
                "value": "0"
              },
              "ido0002": {
                "pairingID": 17,
                "value": "0"
              },
              "idp0003": {
                "pairingID": 2,
                "value": "0"
              }
            }
          }
        }
      }
    }
  }
}
```

Documentation:

https://developer.eu.mybuildings.abb.com/fah_local/concepts/#data-model

SSID Displayed as 000...000, when the local API is used

2f FID_RGB_ACTUATOR Dim actuator

1 in Switch On/Off 0 - Off
1 - On

17 in Absolute control of the set value percent from 0 to 100

ABB-free@home® – local REST API

Testing the API

Comparison API vs. KNX – Where is the device located?

Server response

Code	Details
200	<p>Response body</p> <pre>{ "00000000-0000-0000-0000-000000000000": { "devices": { "BEEF2F4B0001": { "floor": "01", "room": "02", "displayName": "test", "unresponsive": false, "channels": { "ch0000": { "displayName": "Stehlampe", "functionID": "2f", "inputs": { "idp0000": { "pairingID": 1, "value": "0" }, "idp0001": { "pairingID": 16, "value": "0" }, "idp0002": { "pairingID": 17, "value": "0" }, "idp0003": { "pairingID": 2, "value": "0" } } } } } } } }</pre>

- Building
 - ABB RoomTouch
 - IP Touch
 - Other devices
 - SBS/U6.0 HVAC-Slave-Gerät, 6fach BE
 - DG/S1.16.1 DALI-Gateway,16G,1f,MDRC
 - SAH/S8.6.7.1 Switch/Shutter Act, 8-f, 6...
 - 6131/51-500 Presence detector corrid...
 - 6108/18 Room temperature controller...
 - Flush-mounted room temperature co...
 - 1.1.3 UD/S2.300.2 Universal Dim Act,2-fol...
 - 1.1.4 6127/02 c. elem., solo® conf. 4gang,...
 - 1.1.5 6127/02 c. elem., solo® conf. 4gang,...
 - 1.1.6 SBC/U6.0 HVAC/CO2 device, 6gang BE

ABB-free@home® – local REST API

Testing the API

Comparison API vs. KNX – How is the device called and what is it?

```
Server response
Code    Details
200
Response body
{
  "00000000-0000-0000-0000-000000000000": {
    "devices": {
      "BEE3F4B0001": {
        "floor": "01",
        "room": "02",
        "displayName": "test",
        "unresponsive": false,
        "channels": {
          "ch0000": {
            "displayName": "Stehlampe",
            "functionID": "2f",
            "inputs": {
              "idp0000": {
                "pairingID": 1,
                "value": "0"
              },
              "idp0001": {
                "pairingID": 16,
                "value": "0"
              },
              "idp0002": {
                "pairingID": 17,
                "value": "0"
              },
              "idp0003": {
                "pairingID": 2,
                "value": "0"
              }
            }
          }
        }
      }
    }
  }
}
```

Properties		
Settings	Comments	Information
Catalog		Application
Manufacturer	Busch-Jaeger Elektro	
Product	6127/02 c. elem., solo® conf. 4gang, fl. mtd.	
Application	Control element, comfort 4gang, TP/1	
Device Type	\$6127	
Program Version	5.5	
Certification	Registered	
Fingerprint	1AEC	

ABB-free@home® – local REST API

Testing the API

Comparison API vs. KNX – Which functions/channels are available?

Server response

Code	Details
200	<p>Response body</p> <pre>{ "00000000-0000-0000-0000-000000000000": { "devices": { "BEED3F4B0001": { "floor": "01", "room": "02", "displayName": "test", "unresponsive": false, "channels": { "ch0000": { "displayName": "Stehlampe", "functionID": "2F", "inputs": { "idp0000": { "pairingID": 1, "value": "0" }, "idp0001": { "pairingID": 16, "value": "0" }, "idp0002": { "pairingID": 17, "value": "0" }, "idp0003": { "pairingID": 2, "value": "0" } } } } } } } }</pre>

Number	Name	Object Function	Description	Group Address	Length	C	R	W	T	U	Data Type	Priority
1	S1.1: Switching	Input / output			1 bit	C	-	W	T	U	switch	Low
1	S1.1: Switching	Input / output			1 bit	C	-	W	T	U	switch	Low
2	S1.1: Relative dimming	Output			4 bit	C	-	-	T	-	dimming c...	Low
21	S2.1: Switching	Input / output			1 bit	C	-	W	T	U	switch	Low
41	S3.1: Switching	Input / output			1 bit	C	-	W	T	U	switch	Low
61	S4.1: Switching	Input / output			1 bit	C	-	W	T	U	switch	Low
61	S4.1: Value switching	Input / output			1 byte	C	-	W	T	U	percentag...	Low

Data Type

1* 1-bit
1.001 switch

ABB-free@home® – local REST API

Testing the API

Main API – Get datapoint

- Get the current value of a given datapoint

```
Server response
Code    Details
200     Response body
{
  "00000000-0000-0000-0000-000000000000": {
    "values": [
      "0"
    ]
  }
}
```

GET /api/rest/datapoint/{sysap}/{device}.{channel}.{datapoint} Get datapoint value

Get the current value of a given datapoint. See the [documentation on datapoints](#) for further information and the [samples](#) for an example.

Parameters Cancel

Name	Description
sysap * required string (path) pattern: ^0{8}-0{4}-0{4}-0{4}-0{12}\$	SysAP UUID
device * required string (path) pattern: ^[0-9A-Fa-f]{12}\$	Datapoint Serial. as obtained from e.g. a /api/rest/configuration call.
channel * required string (path) pattern: ^ch[0-9A-Fa-f]{4}\$	Channel of a device. Selects a channel in a device, as obtained from e.g. a /api/rest/configuration call.
datapoint * required string (path) pattern: ^[io]dp[0-9A-Fa-f]{4}\$	Datapoint Serial. Selects a datapoint in a channel of a device.

Execute Clear

ABB-free@home® – local REST API

Testing the API

Comparison API vs. KNX – How is the device called and what is it?

```
Curl  
curl -X GET "http://192.168.0.129/fhapi/v1/api/rest/datapoint/00000000-0000-0000-0000-000000000000/6000D2CB27B2.ch0000.odp0000" -H "accept: application/json"
```

Server response

Code 200

Details

Response body

```
{  
  "00000000-0000-0000-0000-000000000000": {  
    "values": [  
      "0"  
    ]  
  }  
}
```

Group Address 10/0/7

Data point type 9.001 temperature (°C)

Delay time[sec] 0

Last received value 0C 1A | 21 °C

Value 0 °C

Send cyclically

Write

Read

#	Time	Serv	Fla	Pri	Source Add	Source Name	Destination	Destination Name	Rout	Type	DPT	Info
20.11.2020 13:59:05,137		to b...	L...	1.1254	-		10/0/7	Setpoint display	6	GroupValueRead		
20.11.2020 13:59:05,158		fro...	L...	1.16	SBC/U6.0 HVAC/CO2 device, 6ga...		10/0/7	Setpoint display	6	GroupValueRespo...	9.001 temperature	0C 1A 21 °C

ABB-free@home® – local REST API

Testing the API

Main API – Set datapoint value

- Set a new value for a given datapoint

Server response	
Code	Details
200	<p>Response body</p> <pre>{ "00000000-0000-0000-0000-000000000000": { "result": "OK" } }</pre>

PUT /api/rest/datapoint/{sysap}/{device}.{channel}.{datapoint} Set datapoint value

Set a new value for a given datapoint. See the [documentation on datapoints](#) for further information and the [samples](#) for an example.

Parameters Cancel

Name	Description
sysap * required string (path) pattern: ^0{8}-0{4}-0{4}-0{4}-0{12}\$	SysAP UUID <input type="text" value="00000000-0000-0000-0000-000000000000"/>
device * required string (path) pattern: ^[0-9A-Fa-f]{12}\$	Datapoint Serial. as obtained from e.g. a /api/rest/configuration call. <input type="text" value="BEED3F4B0001"/>
channel * required string (path) pattern: ^ch[0-9A-Fa-f]{4}\$	Channel of a device. Selects a channel in a device, as obtained from e.g. a /api/rest/configuration call. <input type="text" value="ch0000"/>
datapoint * required string (path) pattern: ^[io]dp[0-9A-Fa-f]{4}\$	Datapoint Serial. Selects a datapoint in a channel of a device. <input type="text" value="idp0002"/>

Request body text/plain

New value

ABB-free@home® – local REST API

Testing the API

Main API – Virtual devices

- Create a virtual device inside free@home
- You can choose a serial number freely

- BinarySensor
- SwitchingActuator
- CeilingFanActuator
- RTC
- DimActuator
- WindowSensor
- ShutterActuator
- WeatherStation
- Weather-TemperatureSensor
- Weather-WindSensor
- Weather-BrightnessSensor
- Weather-RainSensor
- CODetector
- FireDetector

The screenshot shows a REST client interface with the following sections:

- Parameters:** A table with columns 'Name' and 'Description'.

Name	Description
sysap * required string (path) pattern: ^0{8}-0{4}-0{4}-0{4}-0{12}\$	SysAP UUID <input type="text" value="00000000-0000-0000-0000-000000000000"/>
serial * required string (path) pattern: ^[a-zA-Z0-9\-_]{1,64}\$	serialnumber for virtual device (choose freely) <input type="text" value="abcd12345"/>
- Request body** * required: A dropdown menu set to 'application/json'.
- Optional description in JSON:** A text area containing the following JSON:

```
{
  "type": "SwitchingActuator",
  "properties": {
    "ttl": "180",
    "displayname": "Virtual switch"
  }
}
```
- Execute:** A large blue button at the bottom.
- Cancel:** A small red button in the top right corner.

ABB-free@home® – local REST API

Testing the API

Main API – Virtual devices – TTL

- TTL = Time to Life
- Timelife of the virtual device in seconds
- To mark a virtual device as unresponsive (or ready for removal), reregister the device with a TTL value of zero
- To avoid re-registering of the virtual device at regular intervals, the API user can specify a TTL value of -1

```
{  
  "type": "RTC",  
  "properties": {  
    "ttl": "180",  
    "displayname": "Virtual switch"  
  }  
}
```

ABB-free@home® – local REST API

Testing the API

Use cases with virtual devices

- You can register a garage door opener in your home automation system as virtual device to control and monitor it from there
- Use virtual switch actuators for internal actions (logic functions)
- Use virtual switch actuators to control 3rd party systems like Gardena or Velux via free@home
- Integrate weather data from Netatmo
- Integrate a nest thermostat into free@home
- Etc.

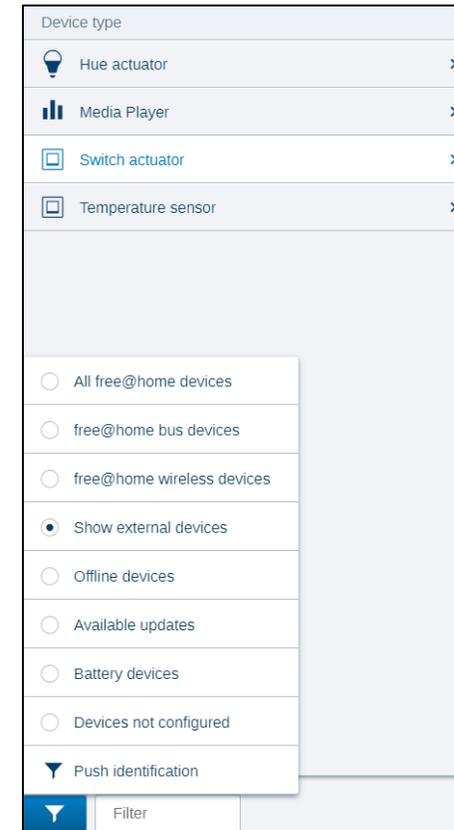


ABB-free@home® – local REST API

Testing the API

Deleting virtual devices via free@home

- Virtual devices can be deleted via the free@home menu
- Device configuration → select the virtual device, which should be deleted → Maintenance → Reset and “Delete external device”

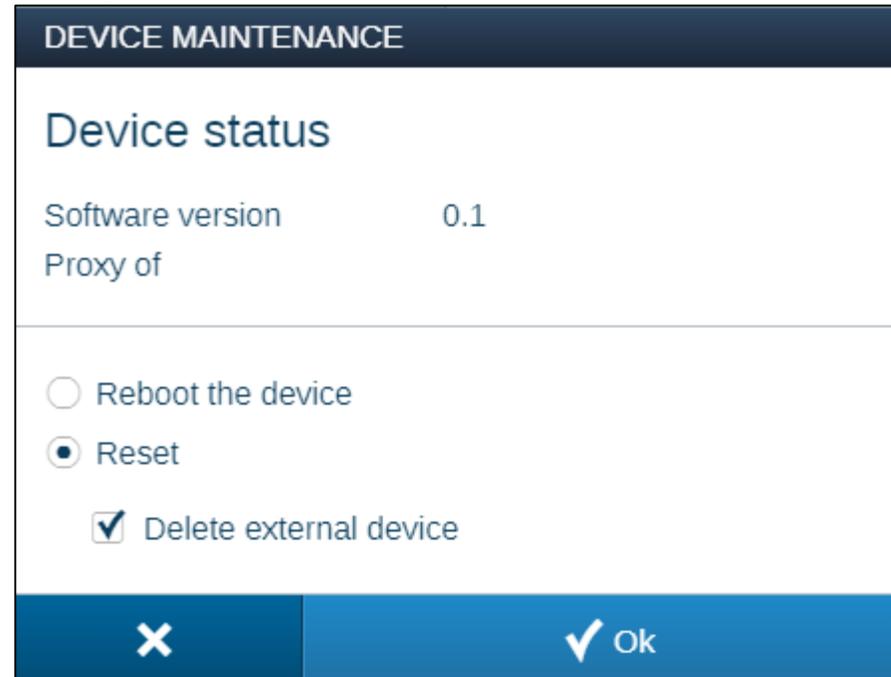
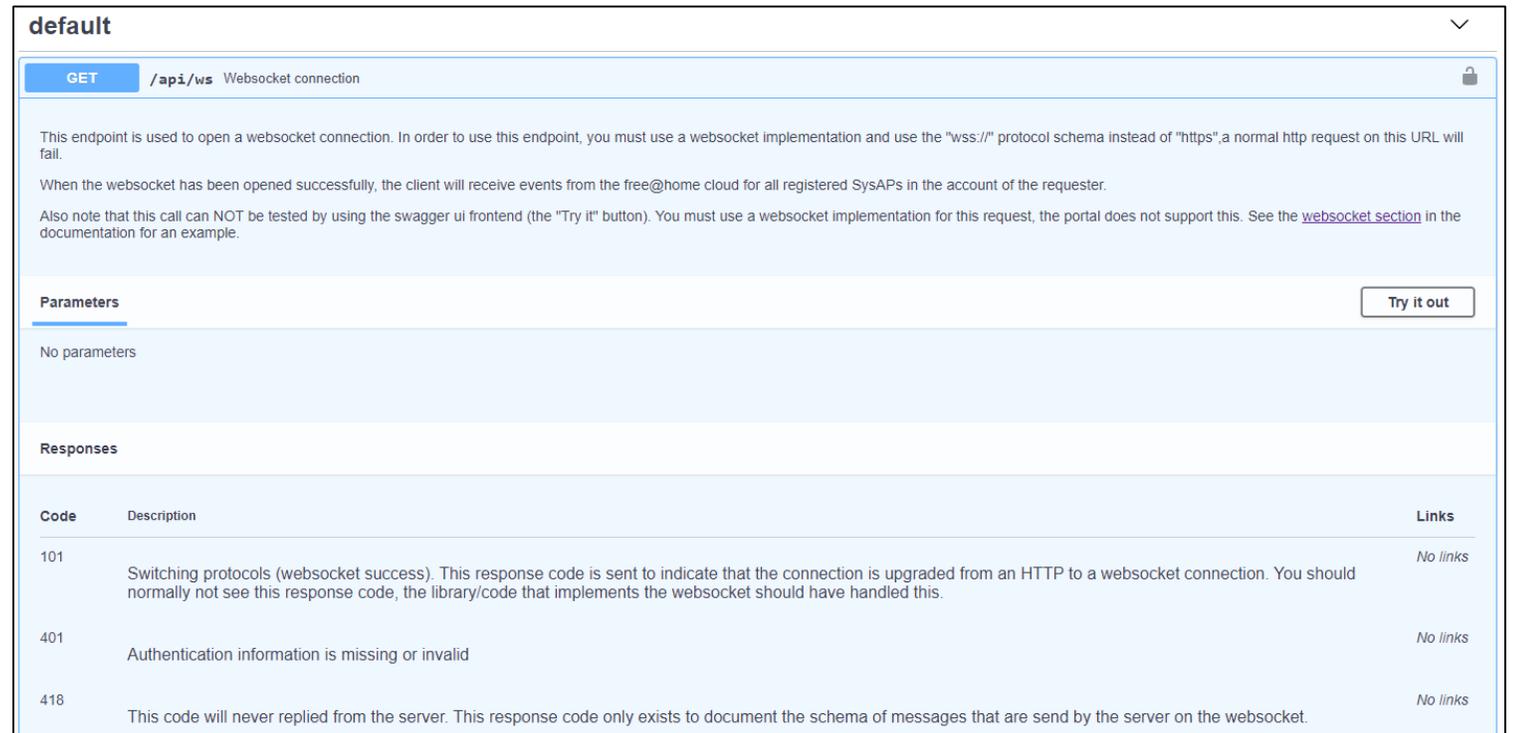


ABB-free@home® – local REST API

Testing the API

WS - Websockets

- WebSocket is a computer communication protocol, providing full-duplex communication channels over a single TCP connection
- Used to push, send and receive data to and from browsers (example: chat applications like Telegram, Whatsapp, etc.)
- Here we use websockets to send and receive data to and from the System Access Point



The screenshot shows the Swagger UI for the endpoint `/api/ws` with the method `GET`. The title is "default" and the endpoint is labeled "Websocket connection".

GET /api/ws Websocket connection

This endpoint is used to open a websocket connection. In order to use this endpoint, you must use a websocket implementation and use the "wss://" protocol schema instead of "https", a normal http request on this URL will fail.

When the websocket has been opened successfully, the client will receive events from the free@home cloud for all registered SysAPs in the account of the requester.

Also note that this call can NOT be tested by using the swagger ui frontend (the "Try it" button). You must use a websocket implementation for this request, the portal does not support this. See the [websocket section](#) in the documentation for an example.

Parameters

No parameters

Responses

Code	Description	Links
101	Switching protocols (websocket success). This response code is sent to indicate that the connection is upgraded from an HTTP to a websocket connection. You should normally not see this response code, the library/code that implements the websocket should have handled this.	No links
401	Authentication information is missing or invalid	No links
418	This code will never replied from the server. This response code only exists to document the schema of messages that are send by the server on the websocket.	No links

ABB-free@home[®] – local REST API

Practical Examples

ABB-free@home® – local REST API

Installation

Preparation

- The following devices are needed to work with the free@home API:
 - System Access Point 2.0 SAP/S.3 (FW: 2.6.0. or higher)
 - Raspberry Pi + Power Supply + Housing
 - SD card with 4GB or more
 - Windows/MAC/Linux PC with SD card reader



ABB-free@home® – local REST API

Practical Examples

Integration free@home in Apple Home(Kit)

- free@home can be integrated into Apple Home Kit with the local REST API



ABB-free@home® – local REST API

Practical Examples

Integration free@home in Apple Home(Kit)

- Setup your Raspberry Pi and install the Homebridge application
- Open <http://homebridge.local> or `http://<ip address of your server>`

The screenshot displays the Homebridge web interface. At the top, there are navigation tabs for 'Homebridge', 'Status', 'Plugins', 'Konfiguration', and 'Geräte'. The main content area is divided into several sections:

- Homebridge Status:** Shows three green checkmarks indicating that the Homebridge service is up-to-date (v1.1.6), running, and plugins are also up-to-date.
- System Metrics:** A row of five cards showing: Processor (3% Last), Temperature (33°C), Memory (3.74 GB Total, 3.47 GB Free), and Uptime (39m Server, 25m Process).
- QR Code:** A large QR code with the pairing code '321-65-986' below it. Text below the QR code reads: 'Scanne diesen Code mit der Kamera auf deinem iOS-Gerät, um Homebridge zu Apple Home hinzuzufügen.'
- System Informationen:** A table listing system details:

Zeitzone	GMT+0100
OS	Raspbian GNU/Linux Buster (10)
Hostname	homebridge
IPv4 (eth0)	192.168.0.131
IPv6 (eth0)	2a02:908:1f43:d3e0:5ffd:f2ad:1b42:120d
Node.js Version	v14.15.1
Npm Version	v6.14.9
Benutzer	pi
Speicherpfad	/var/lib/homebridge
Konfigurationspfad	/var/lib/homebridge/config.json
Service-Modus	Yes
- Homebridge Protokoll:** A log window showing the startup sequence, including messages from the Supervisor, Node.js, and the Homebridge UI. It ends with the message: 'Homebridge is running on port 51220.'

At the bottom right of the interface, there is a small purple plus icon and a footer that reads 'homebridge-config-ui-x v4.36.0 - © 2020 aznu'.

ABB-free@home® – local REST API

Practical Examples

Integration free@home in Apple Home(Kit)

- Open the Home app on your mobile device



- Tap the Home tab, then tap 
- Add Accessories and scan the QR code displayed in the Homebridge user interface
- Follow the installation wizard to finish the integration

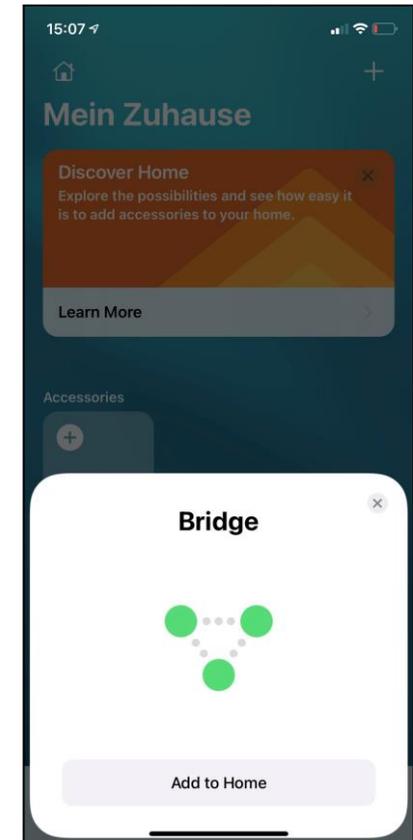
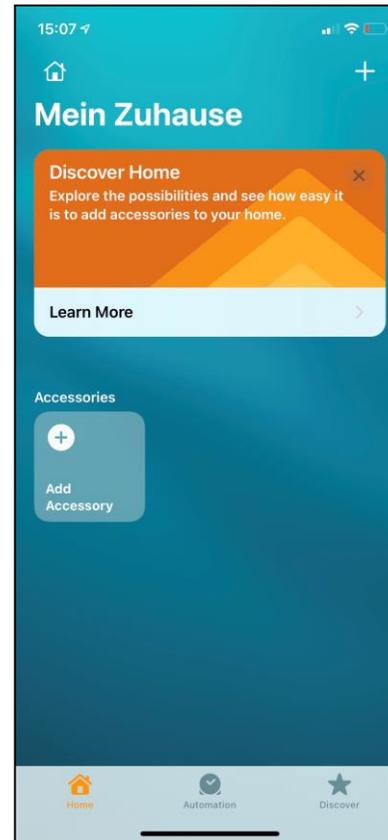


ABB-free@home® – local REST API

Practical Examples

Integration free@home in Apple Home(Kit)

- Open the Homebridge interface again and navigate to “Plugins”
- Search for “freeathome” and install the plugin
- Navigate to “configuration”
- Edit the config.json with your personal credentials

```
{  
  "platform": "free@home",  
  "sysIP": "<IP>",  
  "username": "<USERNAME>",  
  "password": "<PASSWORD>",  
  "mappings": {}  
}
```

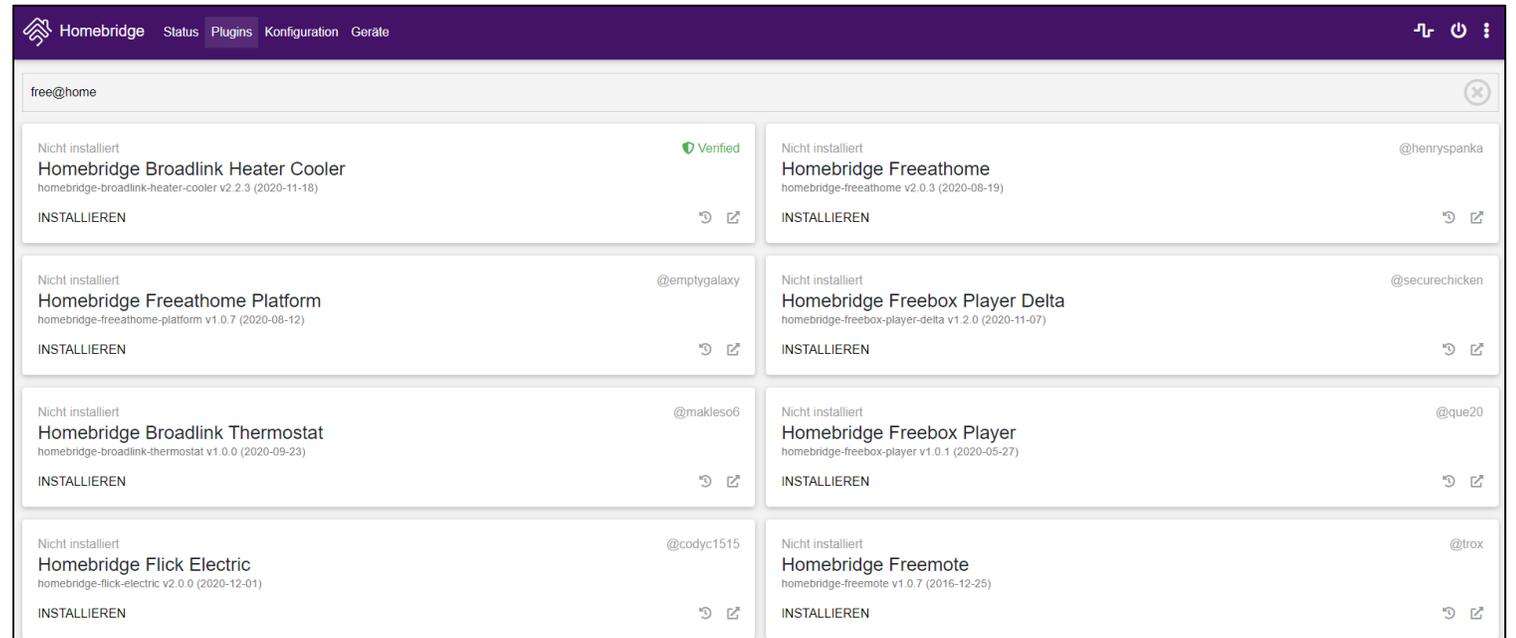


ABB-free@home® – local REST API

Practical Examples

Integration free@home in Apple Home(Kit)

- The free@home devices should appear in your Home app now (this may take some minutes after the restart)

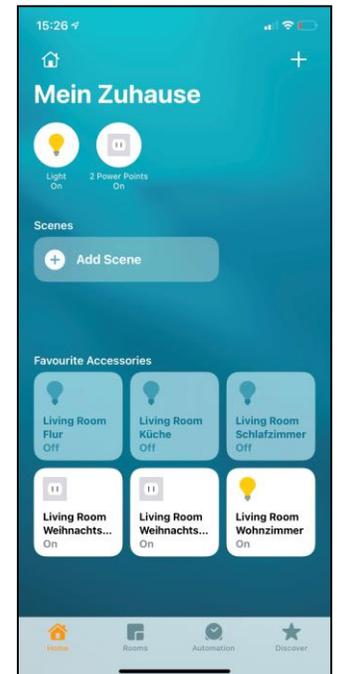
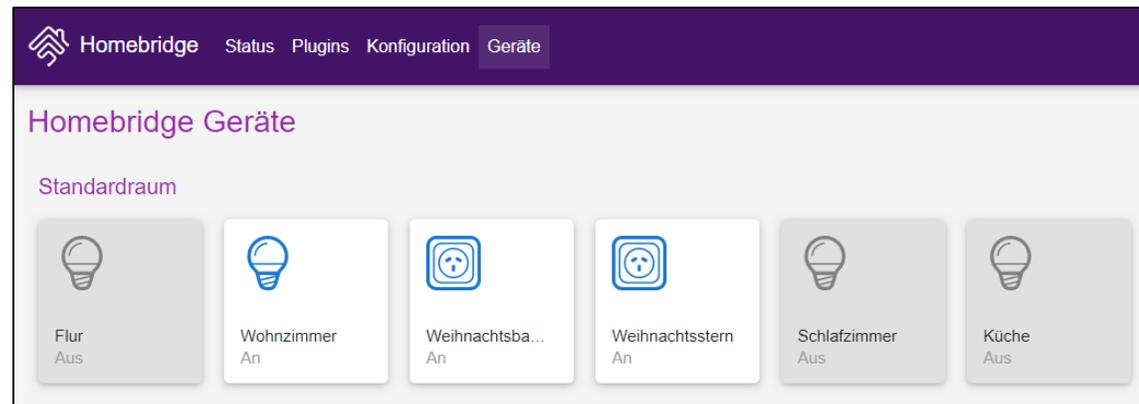


ABB-free@home® – local REST API

Practical Examples

Netatmo integration

- In this example an open hardware platform like a raspberry pi is taken and equipped with open-source software like „Node-RED“
- Inside free@home a virtual weather station is created which appear in free@home, but is not belonging to any physical auxiliary device
- Now the data from a 3rd party weather station is injected to the virtual weather station inside free@home
- On the righthand side you can see the data inside free@home next app. It appears as native weather station data
- Precondition: IoT Hub set up and running and a SysAP 2.0 with activated API

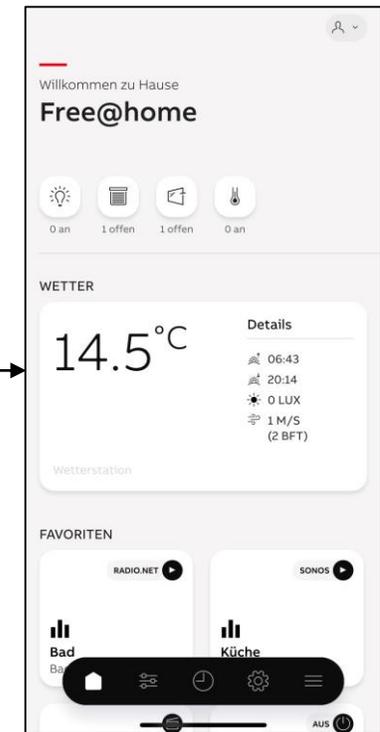


ABB-free@home® – local REST API

Practical Examples

Setup of the IoT Hub

- To integrate a 3rd party system to free@home an IoT Hub must be installed on your Raspberry Pi
- Possible IoT Hubs:
 - OpenHAB2: open IoT hub to network several systems and services with each other (free@home binding already included)
 - Node-RED: visual IoT logic editor for connecting several systems with the help of "flows"
 - Grafana: open web visualization to be able to easily build dashboards
 - influxDB: Database in the background to persist all values in the smart home



ABB-free@home® – local REST API

Practical Examples

Linking free@home and Netatmo

- 1. Integrate free@home in openHAB
- 2. Integrate Netatmo in openHAB
- 3. Create a virtual weather station in free@home (swagger interface)
- 4. Linking Netatmo and free@home via Node-RED

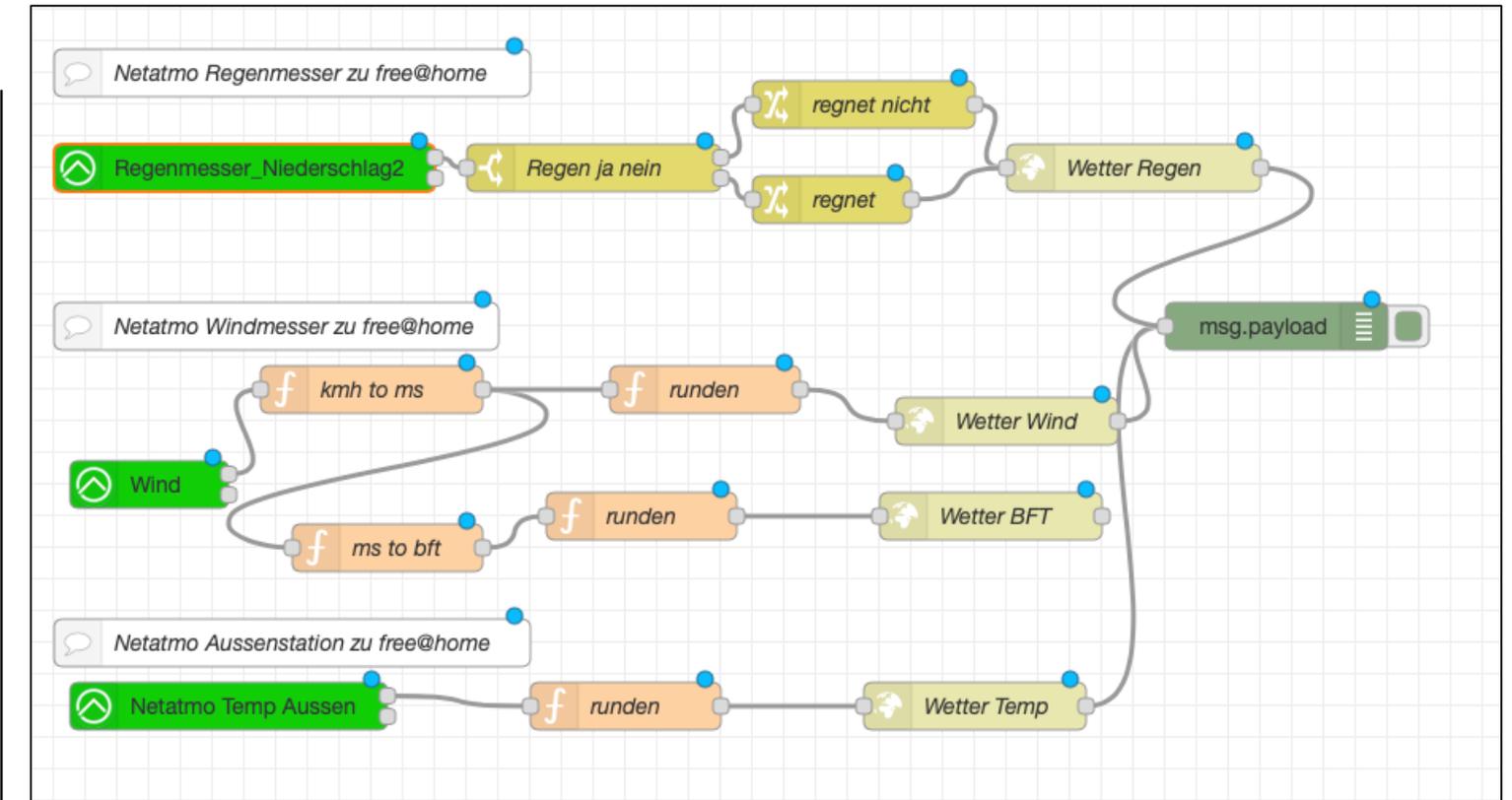
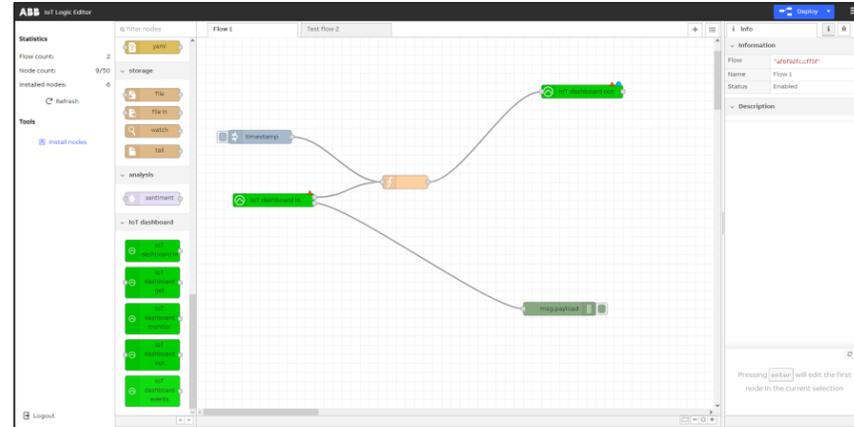


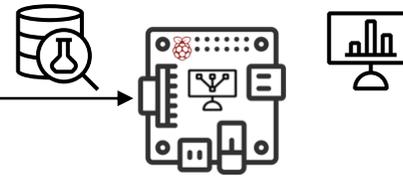
ABB-free@home® – local REST API

Practical Examples

Visualize data from free@home



free@home



Open HW like
Raspberry Pi

ABB-free@home® – local REST API

Practical Examples

Visualize data from free@home

- In this example an open HW platform like a raspberry pi is taken an equipped with open-source software like „Node-RED, InfluxDB and Grafana
- This gives the possibility to easily gather and persist all data coming from the free@home system and to visualize it

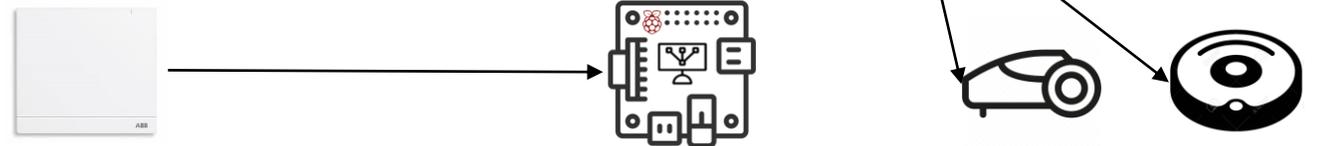
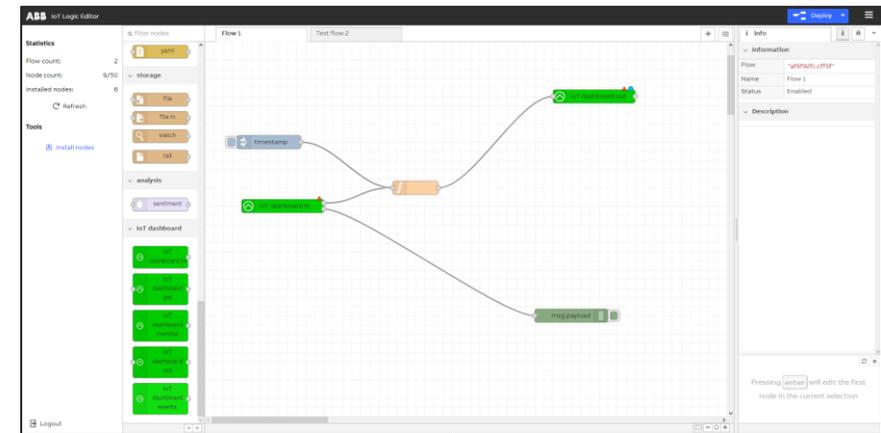


ABB-free@home® – local REST API

Practical Examples

Smart robot mower integration

- In this example a smart robot mower can be started with virtual devices in the free@home interface



Disclaimer

Technical data in this presentation are only approximate figures. The information in this presentation is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this presentation.

ABB shall in no case be liable under, or in connection with the presentation towards any person or entity, to which the presentation has been made available, in view of any damages or losses – irrespective of the legal grounds. In particular ABB shall in no event be liable for any indirect, consequential or special damages, such as - but not limited to – loss of profit, loss of revenue, loss of earnings, cost of capital or cost connected with an interruption of business.

© Copyright 2020 ABB. All rights reserved.

ABB