Thinknx Integration Kit

A downloadable version of this document is available here.

The purpose of this document is to illustrate how to integrate third party systems with Thinknx servers and its functionalities. Integration can be on different levels depending on the structure of the plant, required system reliability and internet connection availability. Here follows the possible integration scenarios:

- Local based integration – 1 to 1
- Cloud based integration – 1 to 1 or 1 to many
- On premise integration – 1 to many

On the lowest level, the communication between the systems is based on a set of commands based on Representational State Transfer (RESTful API).

After the final commissioning, depending on the kind of plants and integrated devices, all the functionalities created for the end user can be also used by third party services through Thinknx Integration Kit API. It is possible to distinguish several main devices that can be controlled that are the same of the UI elements available from Thinknx Configurator

- Lights, switches and bulbs
- Blinds, curtains and lamellas
- Thermostats
- HVAC units and fans
- Analog values
- Scenes
- Generic Buttons

In addition to these UI objects it is also possible to send command directly to the system and interact with KNX bus without any UI layer. Devices are available only after the creation of the project and its deployment on the server. Each device is identified by a unique identifier (GUID) that can either been read from Configurator or from the dedicated server web page (Server→Integration Kit). When you sending content to or making a request to the API, the response will be returned in JSON. JSON is an open standard data format that is lightweight and human-readable, and looks like Objects do in JavaScript; hence the name.

All API access is over HTTP or HTTPS as better specified on the following chapters. HTTP requests will be redirected to HTTPS when possible. HTTP returns code are used to notify a successful request or errors. Possible error status codes are:

- 403 - the authentication information is incorrect.
- 400 - there is an error in the construction of the request. The body of the response will contain more detail of the problem.
- 404 - the requested device could not be found. This may also occur if the user does not have access to the requested device.
- 429 - the request exceeded the rate limit.
- 500 - something went wrong on the server. This error should not occur.

For sake of simplicity, all the calls are performed using GET requests. Also POST requests can be used instead of GET if needed. In this last case use “Content-Type: application/x-www-form-urlencoded” in header to grant maximum compatibility.

Local based integration

The communication between third party service and the server is direct and don’t require any internet connection. The third party app can directly authenticate with Thinknx server. By appending *&auth=basic* or *&auth=digest* to the API request as seen in the example below, the Authentication can be Simple Auth or Digest Auth. The server will respond to its IP at port 5051. All the API calls have the following form:

http://username:password@server_IP:5051/api/V1/controller?param1=value1&param2=value2&auth=digest

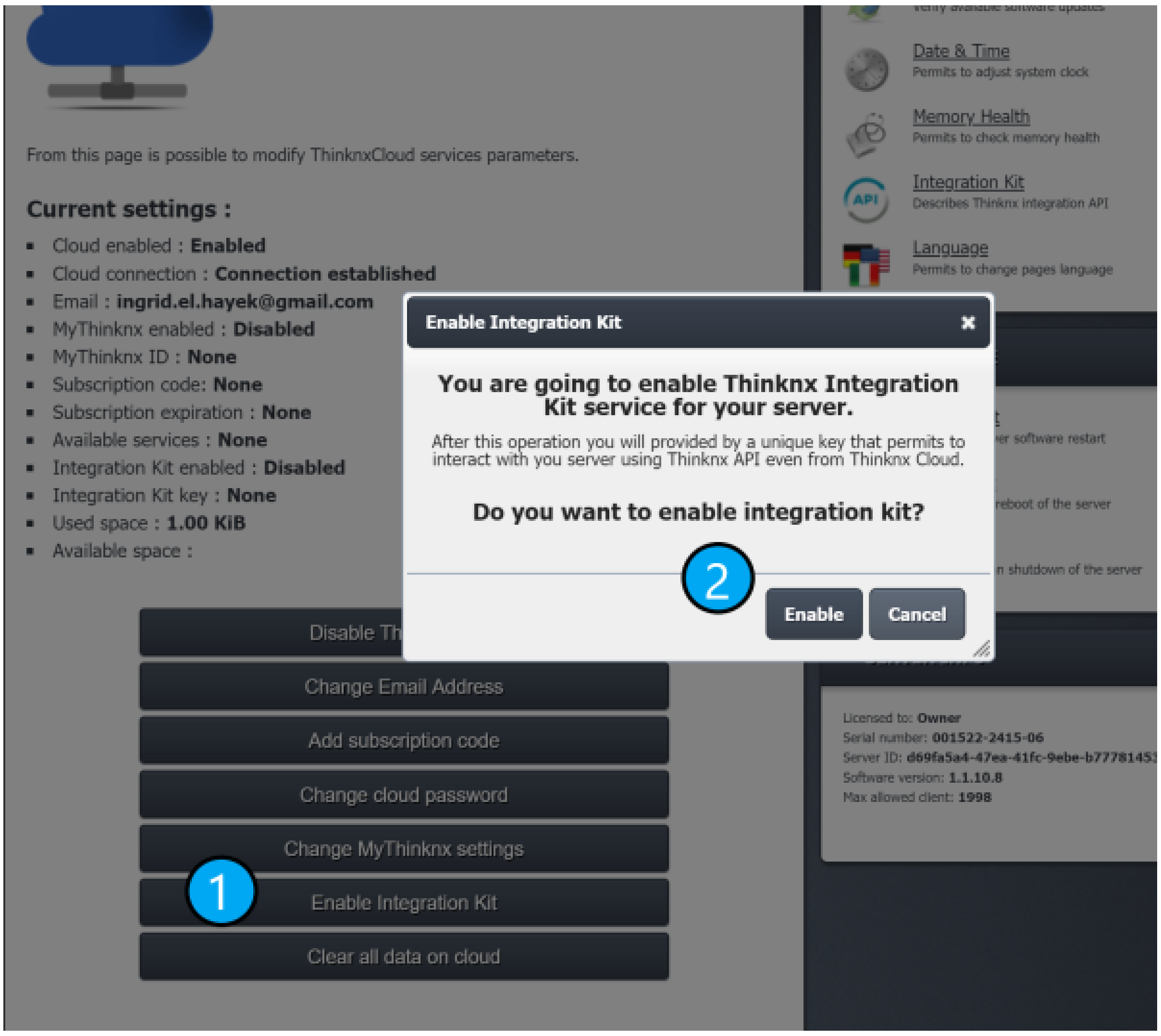
In case the proposed url scheme doesn't work for your integration, you can use the following syntax:

http://server_IP:5051/api/V1/controller?param1=value1&param2=value2&auth=basic&user=service&pass=password

Please note that in this case server will respond only to plain http requests (no https).

Cloud based integration

Thinknx servers are connected through internet to Thinknx Cloud service. Thinknx Cloud service permits to access to the servers independently from the network structure and to remotely control them. Each Thinknx server is identified by its own serial number and by a so called “Integration Kit” key. This key is an alphanumerical code that will be used to route the calls from the Cloud server to the correspondent Thinknx server. The key need to be enabled and generated from the Thinknx server web pages under “Server” tab, “Thinknx Cloud” page. Click on “Enable Integration Kit”.


From this page is possible to modify ThinknxCloud services parameters.

Current settings :

- Cloud enabled : **Enabled**
- Cloud connection : **Connection established**
- Email : **ingrid.el.hayek@gmail.com**
- MyThinknx enabled : **Disabled**
- MyThinknx ID : **None**
- Subscription code: **None**
- Subscription expiration : **None**
- Available services : **None**
- Integration Kit enabled : **Disabled**
- Integration Kit key : **None**
- Used space : **1.00 KiB**
- Available space :

Enable Integration Kit

You are going to enable Thinknx Integration Kit service for your server.

After this operation you will provided by a unique key that permits to interact with you server using Thinknx API even from Thinknx Cloud.

Do you want to enable integration kit?

Licensed to: **Owner**
Serial number: **001522-2415-06**
Server ID: **d69fa5a4-47ea-41fc-9ebe-b77781453**
Software version: **1.1.10.8**
Max allowed client: **1998**

Enable Integration Kit Key

Key permits to directly connect to the server without additional authentication; for this reason it should be kept secret and these precautions should be taken into account while writing code or using cloud based integration:

1. Don't expose key on forms or directly into web pages
2. Don't use key on software that run client side (i.e. javascript pages running on browser)
3. Make all the calls to cloud using https

All the API calls used over cloud have the following form:

https://data.thinknx.eu/KEY/api/V1/controller?param1=value1&param2=value2

HTTP calls will be redirect to HTTPS.

On premise integration

This type of integration is relevant when there is the need to maintains services operational even without internet connection and when it is also important to have contemporary multiple server access. In this case, all the Thinknx Cloud service software can be transferred into a local machine installed on premise. Thinknx servers will connect to this local machine and also third party service will connect to this machine using the methods described into the Cloud Based Integration.

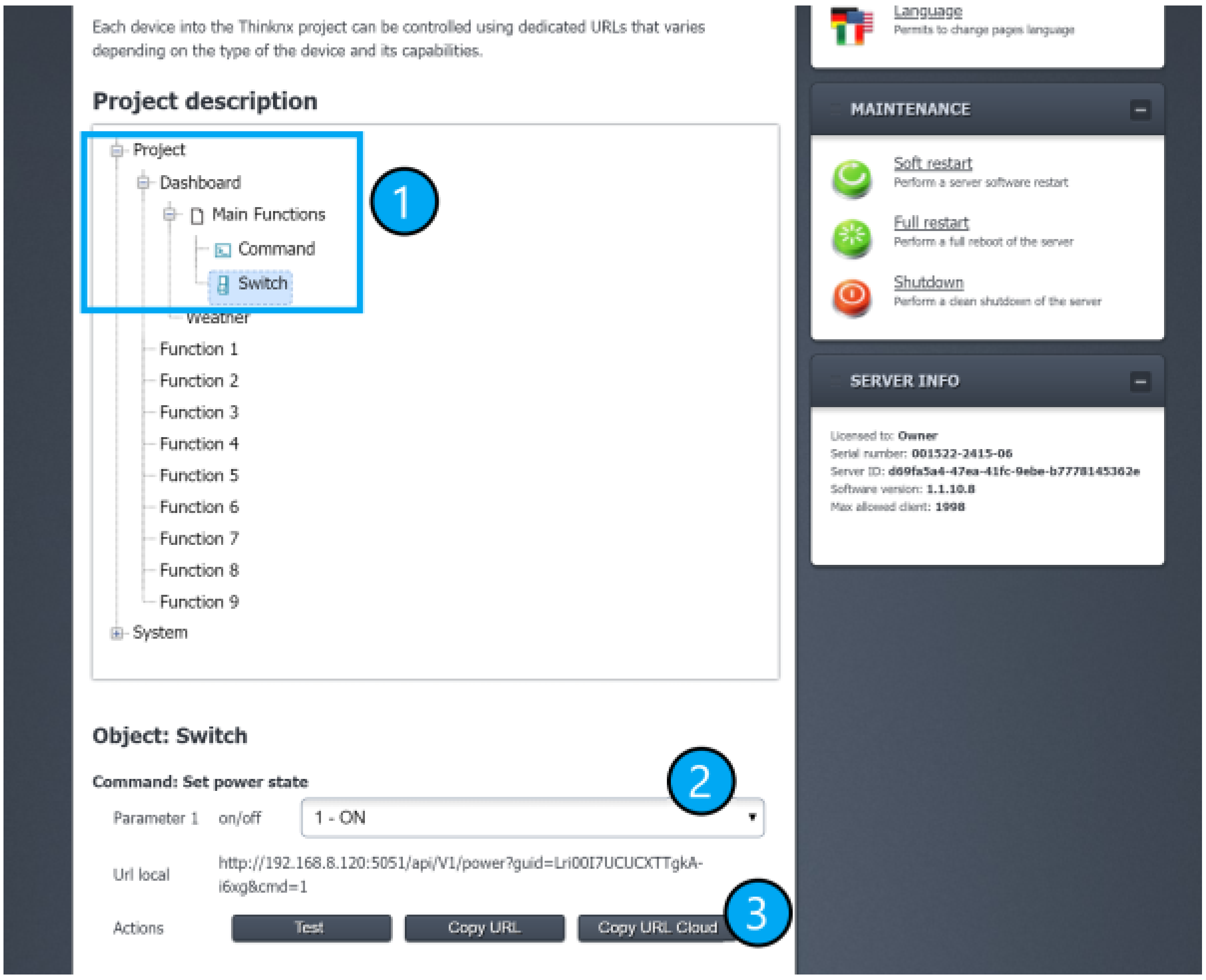
Devices capabilities

Each device type has its own capabilities that can be operated using the API. Not all capabilities are available to all the devices. Here the list of available capabilities:

- Lights, switches and bulbs
  - Power Controller - Turn a light on or off or get its state
  - PowerLevel Controller – Set or get the power level of an endpoint

2. Blinds, curtains and lamellas
   - Movement Controller – Move up, down or stop an endpoint
   - Position Controller – Set or get the position of an endpoint
   - Angle Controller – Set or get or step the angle of an endpoint
3. Analog values
   - Value Controller – Set or get a value to an endpoint on a continuous range
4. Thermostats
   - Actual temperature Controller – Get the actual temperature for an endpoint
   - Setpoint Controller – Set or get the desired temperature for an endpoint
   - Modality Controller – Set or get the functioning modality for an endpoint
   - Chronotable Controller – Enable or disable chrono scheduling for an endpoint
5. HVAC units and fans
   - Power Controller – Turn an endpoint on or off or get its state
   - Setpoint Controller – Set or get the desired temperature for an endpoint
   - Modality Controller – Set or get the functioning modality for an endpoint
   - FanSpeed Controller – Set or get the speed of the fan
6. Scenes
   - Launch Controller – Can recall a user recorded scene or stop its execution
7. Generic Button
   - Trigger Controller – Can trigger the actions associated with an endpoint

   From Thinknx server webpages inside “Server” tab, page “Integration Kit” it is possible to see the entire tree of the project running on the server. Going to the desired object to be controlled is possible to see the supported commands, to simulate them or to copy the URL to be able to send the command remotely.



Each device into the Thinknx project can be controlled using dedicated URLs that varies depending on the type of the device and its capabilities.

Project description

Object: Switch

Command: Set power state

Parameter 1 on/off 1 - ON

Url local http://192.168.8.120:5051/api/V1/power?guid=Lri00I7UCUCXTTgkA-i6xg&cmd=1

Actions

Language
Permits to change pages language

Soft restart
Perform a server software restart

Full restart
Perform a full reboot of the server

Shutdown
Perform a clean shutdown of the server

Licensed to: **Owner**
Serial number: **001522-2415-06**
Server ID: **d69fa5a4-47ea-41fc-9ebe-b7778145362e**
Software version: **1.1.10.8**
Max allowed client: **1998**

Project tree with API description

Asynchronous events (coming soon)

Thinknx Service can provide events to the third party service thanks to REST hooks. The third party system can define HTTP callbacks to be triggered by predefined events occurring on the devices. HTTP callbacks can be connected to the entire server, a single device or event a device controller.

API URLs

Here follows the list of the API URLs for each device type and its controllers.

Lights, switches and bulbs

Power Controller

Description	**Set the status On or Off for a defined switch object**
URL	***api/v1/power***

Description	**Set the status On or Off for a defined switch object**
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the light to operate to **cmd=*[0 or 1]*** where 0 turn off the device whilst 1 turn it on Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “power”,`
	`“guid”: [device_identifier],`
	` “comm_obj” : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`“namespace” : “power”,`
	`“guid”: [device_identifier],`
	`“status” : [0 or 1],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual power status (0=OFF / 1=ON)

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/power?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1&fb**

Power Controller (Status)

Description	**Get the status On or Off for a defined switch object**
URL	***api/v1/powerstatus***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the light to operate to
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “powerstatus”,`
	`“guid”: [device_identifier],`
	`“status” : [0 or 1],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual power status (0=OFF / 1=ON)

Description	**Get the status On or Off for a defined switch object**
Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/powerstatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

Power Level Controller

Description	**Set the power level for a defined switch dimmable object**
URL	***api/v1/powerlevel***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the light to operate to **cmd=*[0 to 100]*** where the number represents the power level (in percentage) to set the for the device Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “powerlevel”,`
	`“guid”: [device_identifier],`
	` “comm_obj” : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`“namespace” : “powerlevel”,`
	`“guid”: [device_identifier],`
	`“status” : [0 to 100],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual power level in percentage

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/powerlevel?guid=1IzPszHWAEuvzxtXBMkzWQ&&cmd=50&fb**

Power Level Controller (Status)

Description	**Get the status of power level for a defined dimmable switch object**
URL	***api/v1/powerlevelstatus***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the light to operate to
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “powerlevelstatus”,`
	`“guid”: [device_identifier],`
	`“status” : [0 to 100],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual power level in percentage

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/powerlevelstatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

Blinds, curtains and lamellas

Movement Controller / Up or Down

Description	**Move up, down a defined blind or curtain**
URL	***api/v1/updown***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **cmd=*[0 or 1]*** where the number represents the desired movement direction to set the for the device 0=Move UP / 1=Move DOWN
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “updown”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`

Error Response Code	400 or 404

Description	**Move up, down a defined blind or curtain**

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/updown?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1**

Movement Controller / Stop

Description	**Stop the movement of a defined blind or curtain**
URL	***api/v1/stop***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **cmd=1** needed to assert the stop command
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “stop”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/stop?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1**

Position Controller / Height

Description	**Set the height/position of a defined blind or curtain**
URL	***api/v1/heightlevel***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **val=*[0 to 100]*** where the number represents the desired height/position to set the for the device in percentage Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required

Description	**Set the height/position of a defined blind or curtain**
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “heightlevel”,`
	`“guid”: [device_identifier],`
	` “comm_obj” : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`“namespace” : “heightlevel”,`
	`“guid”: [device_identifier],`
	`“status” : [0 to 100],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual height/position in percentage

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/heightlevel?guid=1IzPszHWAEuvzxtXBMkzWQ&&val=50&fb**

Position Controller / Height (Status)

Description	**Get the height/position of a defined blind or curtain**
URL	***api/v1/heightlevelstatus***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “heightlevelstatus”,`
	`“guid”: [device_identifier],`
	`“status” : [0 to 100],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual height/position in percentage

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/heightlevelstatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

Angle Controller / Blind step

Description	**Step clockwise or counter-clockwise lamellas of a defined blind**
URL	***api/v1/blindstep***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **cmd=*[0 or 1]*** where the number represents the desired step direction to set the for the device 0 = Step clockwise / 1 = Step counter-clockwise
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “blindstep”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/blindstep?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1**

Angle Controller / Blind angle

Description	**Set the angle for the lamellas of a defined blind**
URL	***api/v1/blindangle***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **cmd=*[0 to 100]*** where the number represents the desired angle (in percentage) to set the for the device Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required
Success Response Code	200 (OK)

Description	**Set the angle for the lamellas of a defined blind**

Success Response Content	`{`
	`“namespace” : “blindangle”,`
	`“guid”: [device_identifier],`
	` “comm_obj” : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`“namespace” : “blindangle”,`
	`“guid”: [device_identifier],`
	`“status” : [0 to 100],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual angle in percentage

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/blindangle?guid=1IzPszHWAEuvzxtXBMkzWQ&&cmd=50&fb**

Angle Controller / Blind angle (Status)

Description	**Get the angle for the lamellas of a defined blind**
URL	***api/v1/blindanglestatus***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “blindanglestatus”,`
	`“guid”: [device_identifier],`
	`“status” : [0 to 100],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual angle in percentage

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/blindanglestatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

Analog values

Analog controller / Value

Description	**Set the desired value for a defined "analog value" device**
URL	***api/v1/setvalue***
URL Params	**guid=*device_identifier*** where "device_identifier" is the string that identifies the device to operate to **cmd=*[new_value]*** where "new_ value" represent the desired value to set. Number can be in decimal format with '.' as decimal separator Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`"namespace" : "setvalue",`
	`"guid": [device_identifier],`
	`"comm_obj" : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`"namespace" : "setvalue",`
	`"guid": [device_identifier],`
	`"status" : [actual_value],`
	`"comm_obj" : [obj_num]`
	`}`
	"status" variable value represent actual value

Error Response Code	400 or 404

Error Response Content	`{ "error" : "device not found"}`
	or
	`{ "error" : "params errors" }`
	or
	`{ "error" : "impossible to execute"}`

Example	**/api/V1/setvalue?guid=1IzPszHWAEuvzxtXBMkzWQ&&cmd=85.325&fb**

Analog controller / Value (Status)

Description	**Get the actual value for a defined "analog value" device**
URL	***api/v1/getvalue***
URL Params	**guid=*device_identifier*** where "device_identifier" is the string that identifies the device to operate to
Data Params	not required

Description	**Get the actual value for a defined "analog value" device**
Success Response Code	200 (OK)

Success Response Content	`{`
	`"namespace" : "getvalue",`
	`"guid": [device_identifier],`
	`"status" : [actual_value],`
	`"comm_obj" : [obj_num]`
	`}`
	"status" variable value represent actual value

Error Response Code	400 or 404

Error Response Content	`{ "error" : "device not found"}`
	or
	`{ "error" : "params errors" }`
	or
	`{ "error" : "impossible to execute"}`

Example	**/api/V1/getvalue?guid=1IzPszHWAEuvzxtXBMkzWQ**

Thermostats

Thermostat Controller / Actual temperature

Description	**Get the actual measured temperature for a defined thermostat**
URL	***api/v1/actualtemp***
URL Params	**guid=*device_identifier*** where "device_identifier" is the string that identifies the device to operate to
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`"namespace" : "actualtemp",`
	`"guid": [device_identifier],`
	`"status" : [actual_temperature],`
	`"comm_obj" : [obj_num]`
	`}`
	"status" variable value represent actual temperature

Error Response Code	400 or 404

Error Response Content	`{ "error" : "device not found"}`
	or
	`{ "error" : "params errors" }`
	or
	`{ "error" : "impossible to execute"}`

Example	**/api/V1/actualtemp?guid=1IzPszHWAEuvzxtXBMkzWQ**

Thermostat Controller / Setpoint

Description	**Set the desired setpoint temperature for a thermostat**
URL	***api/v1/setpoint***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **cmd=*[new_setpoint]*** where “new_setpoint” represent the desired temperature. Number can be in decimal format with ‘.’ as decimal separator Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	{
	`“namespace” : “setpoint”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	}
	In case of feedback request (fb parameter present)
	{
	`“namespace” : “setpoint”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_setpoint],`
	`“comm_obj” : [obj_num]`
	}
	“status” variable value represent actual setpoint

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/setpoint?guid=1IzPszHWAEuvzxtXBMkzWQ&&cmd=20.5&fb**

Thermostat Controller / Setpoint (Status)

Description	**Get the actual setpoint temperature for a thermostat**
URL	***api/v1/setpointstatus***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to
Data Params	not required
Success Response Code	200 (OK)

Description	**Get the actual setpoint temperature for a thermostat**

Success Response Content	`{`
	`“namespace” : “setpointstatus”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_setpoint],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual setpoint

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/setpointstatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

Thermostat Controller / Modality

Description	**Set the desired modality for a thermostat**
URL	***api/v1/tempmode***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **cmd=*[new_mode]*** where “new_mode” represent the desired operating modality using the following numbers: 0 = standby 1 = comfort 2 = night/economy 3 = frost/heat protection Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “tempmode”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`“namespace” : “tempmode”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_mode],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable values meaning are same as the ones of the request

Description	**Set the desired modality for a thermostat**
Error Response Code	400 or 404

Error Response Content	`{ "error" : "device not found"}`
	or
	`{ "error" : "params errors" }`
	or
	`{ "error" : "impossible to execute"}`

Example	**/api/V1/tempmode?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1&fb**

Thermostat Controller / Modality (Status)

Description	**Get the actual modality for a thermostat**
URL	***api/v1/tempmodestatus***
URL Params	**guid=*device_identifier*** where "device_identifier" is the string that identifies the device to operate to
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`"namespace" : "tempmodestatus",`
	`"guid": [device_identifier],`
	`"status" : [actual_mode],`
	`"comm_obj" : [obj_num]`
	`}`
	"status" variable values will be as follow:
	0 = standby
	1 = comfort
	2 = night/economy
	3 = frost/heat protection

Error Response Code	400 or 404

Error Response Content	`{ "error" : "device not found"}`
	or
	`{ "error" : "params errors" }`
	or
	`{ "error" : "impossible to execute"}`

Example	**/api/V1/tempmodestatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

Thermostat Controller / Chronoscheduling

Description	**Enable/Disable the chronoscheduling for a defined thermostat**
URL	***api/v1/chronoen***

Description	**Enable/Disable the chronoscheduling for a defined thermostat**
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **cmd=*[0 or 1]*** where 0 disable the chronoscheduling whilst 1 enable it Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “chronoen”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`“namespace” : “chronoen”,`
	`“guid”: [device_identifier],`
	`“status” : [0 or 1],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual chronoscheduling enable status (0=Disabled / 1=Enabled)

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/chronoen?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1&fb**

Thermostat Controller / Chronoscheduling (Status)

Description	**Get the status of chronoscheduling for a defined thermostat**
URL	***api/v1/chronoenstatus***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to
Data Params	not required
Success Response Code	200 (OK)

Description	**Get the status of chronoscheduling for a defined thermostat**

Success Response Content	`{`
	`“namespace” : “chronoenstatus”,`
	`“guid”: [device_identifier],`
	`“status” : [0 or 1],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual chronoscheduling enable status (0=Disabled / 1=Enabled)

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/chronoenstatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

HVAC units and fans

HVAC Controller / Powering

Description	**Set the desired power state for a defined HVAC device**
URL	***api/v1/hvacpower***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **cmd=*[0 or 1]*** where 0 turn off the device whilst 1 turn it on Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “hvacpower”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`“namespace” : “hvacpower”,`
	`“guid”: [device_identifier],`
	`“status” : [0 or 1],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual power status for the device (0=OFF / 1=ON)

Description	**Set the desired power state for a defined HVAC device**
Error Response Code	400 or 404

Error Response Content	`{ "error" : "device not found"}`
	or
	`{ "error" : "params errors" }`
	or
	`{ "error" : "impossible to execute"}`

Example	**/api/V1/hvacpower?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1&fb**

HVAC Controller / Powering (Status)

Description	**Get the actual power state for a defined HVAC device**
URL	***api/v1/hvacpowerstatus***
URL Params	**guid=*device_identifier*** where "device_identifier" is the string that identifies the device to operate to
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`"namespace" : "hvacpowerstatus",`
	`"guid": [device_identifier],`
	`"status" : [0 or 1],`
	`"comm_obj" : [obj_num]`
	`}`
	"status" variable value represent actual power status for the device (0=OFF / 1=ON)

Error Response Code	400 or 404

Error Response Content	`{ "error" : "device not found"}`
	or
	`{ "error" : "params errors" }`
	or
	`{ "error" : "impossible to execute"}`

Example	**/api/V1/hvacpowerstatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

HVAC Controller / Setpoint

Description	**Set the desired setpoint temperature for a defined HVAC device**
URL	***api/v1/hvacsetpoint***
URL Params	**guid=*device_identifier*** where "device_identifier" is the string that identifies the device to operate to **cmd=*[new_setpoint]*** where "new_setpoint" represent the desired temperature. Number can be in decimal format with '.' as decimal separator Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.

Description	**Set the desired setpoint temperature for a defined HVAC device**
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “hvacsetpoint”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`“namespace” : “hvacsetpoint”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_setpoint],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual setpoint

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/hvacsetpoint?guid=1IzPszHWAEuvzxtXBMkzWQ&&cmd=20.5&fb**

HVAC Controller / Setpoint (Status)

Description	**Get the actual setpoint temperature for a defined HVAC device**
URL	***api/v1/hvacsetpointstatus***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “hvacsetpointstatus”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_setpoint],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable value represent actual setpoint

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Description	**Get the actual setpoint temperature for a defined HVAC device**
Example	**/api/V1/hvacsetpointstatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

HVAC Controller / Modality

Description	**Set the desired modality for a defined HVAC device**
URL	***api/v1/hvacmode***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **cmd=*[new_mode]*** where “new_mode” represent the desired operating modality using the following numbers: 0 = Cooling 1 = Heating 2 = Dry 3 = Fan 4 = Auto Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “hvacmode”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`“namespace” : “hvacmode”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_mode],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable values meaning are same as the ones of the request

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/hvacmode?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1&fb**

HVAC Controller / Modality (Status)

Description	**Get the actual modality for a defined HVAC device**
URL	***api/v1/hvacmodestatus***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “hvacmodestatus”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_mode],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable values will be as follow:
	0 = Cooling
	1 = Heating
	2 = Dry
	3 = Fan
	4 = Auto

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/hvacmodestatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

HVAC Controller / Fan speed

Description	**Set the desired fan speed for a defined HVAC device**
URL	***api/v1/hvacfan***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to **cmd=*[new_speed]*** where “new_speed” represent the desired fan speed using the following numbers: 0 = Auto 1 = Minimum speed 2 = Average speed 3 = Maximum speed Optional parameter for set: **fb** if the parameter is present the server will suspend the reply and will wait for the feedback from the operated device.
Data Params	not required
Success Response Code	200 (OK)

Description	**Set the desired fan speed for a defined HVAC device**

Success Response Content	`{`
	`“namespace” : “hvacfan”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`
	In case of feedback request (fb parameter present)
	`{`
	`“namespace” : “hvacfan”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_speed],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable values meaning are same as the ones of the request

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/hvacfan?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=2&fb**

HVAC Controller / Fan speed (Status)

Description	**Get the actual fan speed for a defined HVAC device**
URL	***api/v1/hvacfanstatus***
URL Params	**guid=*device_identifier*** where “device_identifier” is the string that identifies the device to operate to
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “hvacfanstatus”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_speed],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” variable values will be as follow:
	0 = Auto
	1 = Minimum speed
	2 = Average speed
	3 = Maximum speed

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Description	**Get the actual fan speed for a defined HVAC device**
Example	**/api/V1/hvacfanstatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

Scenes

Scene controller / Scene launch

Description	**Launch or stop the execution of a defined scene**
URL	***api/v1/scenelaunch***
URL Params	**guid=*device_identifier*** where "device_identifier" is the string that identifies the scene to operate to **cmd=*[0 or 1]*** where 0 stops the execution of a running scene whilst 1 launches it **Attention:** A running scene is a scene with operations that are not executed instantly (e.g. pauses between single operations). Scenes don't have a state (on or off) but are intended just as a macro i.e. a list of operations to perform.
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`"namespace" : "scenelaunch",`
	`"guid": [device_identifier],`
	`"comm_obj" : [obj_num]`
	`}`

Error Response Code	400 or 404

Error Response Content	`{ "error" : "device not found"}`
	or
	`{ "error" : "params errors" }`
	or
	`{ "error" : "impossible to execute"}`

Example	**/api/V1/scenelaunch?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1**

Generic Button

Generic Button controller / Operation trigger

Description	**Trigger the operation associated to a defined generic button**
URL	***api/v1/trigger***
URL Params	**guid=*device_identifier*** where "device_identifier" is the string that identifies the generic button to operate to
Data Params	not required
Success Response Code	200 (OK)

Description	**Trigger the operation associated to a defined generic button**

Success Response Content	`{`
	`“namespace” : “trigger”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/trigger?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1**

System commands

System controller / Send KNX telegram DPT1

Description	**Set a KNX telegram with datatype DPT1 (bit)**
URL	***api/v1/system/sendKNXbit***
URL Params	**group=*[KNX_group_address]*** where “KNX_group_addres” is KNX group to operate to **value=*[0 or 1]*** value to send to KNX bus at the specified group address
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “system/sendKNXbit”,`
	`“group”: [KNX_group_address],`
	`“value” : [value_sent]`
	`}`

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/system/sendKNXbit?group=1/0/1&value=0**

System controller / Send KNX telegram DPT3 / DPT2

Description	**Set a KNX telegram with datatype DPT3 / DPT2 (4 bits)**
URL	***api/v1/system/sendKNX4bits***
URL Params	**group=*[KNX_group_address]*** where “KNX_group_addres” is KNX group to operate to **value=*[0 to 31]*** value to send to KNX bus at the specified group address
Data Params	not required

Description	**Set a KNX telegram with datatype DPT3 / DPT2 (4 bits)**
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “system/sendKNX4bits”,`
	`“group”: [KNX_group_address],`
	`“value” : [value_sent]`
	`}`

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/system/sendKNX4bits?group=1/0/1&value=30**

System controller / Send KNX telegram DPT5

Description	**Set a KNX telegram with datatype DPT5 (1 byte unsigned)**
URL	***api/v1/system/sendKNXbyte***
URL Params	**group=*[KNX_group_address]*** where “KNX_group_addres” is KNX group to operate to **value=*[0 to 255]*** value to send to KNX bus at the specified group address
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “system/sendKNXbyte”,`
	`“group”: [KNX_group_address],`
	`“value” : [value_sent]`
	`}`

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/system/sendKNXbyte?group=1/0/1&value=30**

System controller / Send KNX telegram DPT6

Description	**Set a KNX telegram with datatype DPT6 (1 byte signed)**
URL	***api/v1/system/sendKNXbyteun***
URL Params	**group=*[KNX_group_address]*** where “KNX_group_addres” is KNX group to operate to **value=*[-128 to 127]*** value to send to KNX bus at the specified group address
Data Params	not required
Success Response Code	200 (OK)

Description	**Set a KNX telegram with datatype DPT6 (1 byte signed)**

Success Response Content	`{`
	`“namespace” : “system/sendKNXbyteun”,`
	`“group”: [KNX_group_address],`
	`“value” : [value_sent]`
	`}`

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/system/sendKNXbyteun?group=1/0/1&value=30**

System controller / Send KNX telegram DPT9

Description	**Set a KNX telegram with datatype DPT9 (2 bytes floating point)**
URL	***api/v1/system/sendKNXfloat2bytes***
URL Params	**group=*[KNX_group_address]*** where “KNX_group_addres” is KNX group to operate to **value=*[new_value]*** value to send to KNX bus at the specified group address
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “system/sendKNXfloat2bytes”,`
	`“group”: [KNX_group_address],`
	`“value” : [value_sent]`
	`}`

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/system/sendKNXfloat2bytes?group=1/0/1&value=85.235**

System controller / Send KNX telegram DPT14

Description	**Set a KNX telegram with datatype DPT14 (4 bytes floating point)**
URL	***api/v1/system/sendKNXfloat4bytes***
URL Params	**group=*[KNX_group_address]*** where “KNX_group_addres” is KNX group to operate to **value=*[new_value]*** value to send to KNX bus at the specified group address
Data Params	not required

Description	**Set a KNX telegram with datatype DPT14 (4 bytes floating point)**
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “system/sendKNXfloat4bytes”,`
	`“group”: [KNX_group_address],`
	`“value” : [value_sent]`
	`}`

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Example	**/api/V1/system/sendKNXfloat4bytes?group=1/0/1&value=85.235**

System controller / Get server status

Description	**Get the most important information from server**
URL	***api/v1/system/serverStatus***
URL Params	not required
Data Params	not required
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “system/serverStatus”,`
	`“version”: [software_version],`
	`“process_uptime”: [process_uptime],`
	`“server_uptime” : [server_uptime]`
	`“config_time”: [config_time],`
	`“connected_clients”: [connected_clients],`
	`“pbx_users”: [connected_pbx_users]`
	`}`
	Where:
	software_version = version of the server firmware in the format “xx.xx.xx.xx”
	process_uptime = time from last server software restart in the format “X days X hours X min”
	server_uptime = time from last server full reboot/repower in the format “X days X hours X min”
	config_time = timestamp of the running configuration in the format “hh:mm:ss YYYY-MM-DD”
	connected_clients = number of active clients connection as number
	connected_pbx_users = identifiers of active connected PBX users separated by commas

Error Response Code	400 or 404

Error Response Content	`{ “error” : “device not found”}`
	or
	`{ “error” : “params errors” }`
	or
	`{ “error” : “impossible to execute”}`

Description	**Get the most important information from server**
Example	**/api/V1/system/serverStatus**
