Thinknx Integration Kit

Una versione scaricabile di questo documento è disponibile qui.

Lo scopo di questo documento è illustrare come integrare sistemi di terze parti con i server Thinknx e le loro funzionalità. L'integrazione può avvenire su diversi livelli, a seconda della struttura dell'impianto, dell'affidabilità richiesta e della disponibilità della connessione Internet. Di seguito sono riportati i possibili scenari di integrazione:

- Integrazione locale – 1 a 1
- Integrazione basata su cloud – 1 a 1 o 1 a molti
- Integrazione on-premise – 1 a molti

Al livello più basso, la comunicazione tra i sistemi si basa su un set di comandi basati su **API RESTful (Representational State Transfer)**.

Dopo la configurazione finale, a seconda del tipo di impianto e dei dispositivi integrati, tutte le funzionalità create per l'utente finale possono essere utilizzate anche da servizi di terze parti tramite le **API del Thinknx Integration Kit**. È possibile distinguere diversi dispositivi principali che possono essere controllati, che corrispondono agli stessi elementi dell'interfaccia utente disponibili nel Thinknx Configurator:

- Luci, interruttori e lampadine
- Tapparelle, tende e lamelle
- Termostati
- Unità HVAC e ventilatori
- Valori analogici
- Scene
- Pulsanti generici

Oltre a questi oggetti dell'interfaccia utente, è anche possibile inviare comandi direttamente al sistema e interagire con il bus KNX senza alcun livello UI. I dispositivi sono disponibili solo dopo la creazione del progetto e il suo deployment sul server. Ogni dispositivo è identificato da un **GUID unico**, che può essere letto dal Configurator o dalla pagina web dedicata del server (Server → Integration Kit). Quando si inviano contenuti o si effettuano richieste alle API, la risposta verrà restituita in **JSON**. JSON è un formato standard aperto per i dati, leggero e leggibile dall'uomo, simile agli **oggetti JavaScript**.

Tutti gli accessi alle API avvengono tramite **HTTP o HTTPS**, come specificato nei capitoli seguenti. Le richieste HTTP verranno reindirizzate a HTTPS quando possibile. I codici di stato HTTP vengono utilizzati per notificare il successo della richiesta o eventuali errori. I possibili codici di errore sono:

- **403** - Informazioni di autenticazione errate.
- **400** - Errore nella costruzione della richiesta. Il corpo della risposta conterrà maggiori dettagli sul problema.
- **404** - Il dispositivo richiesto non è stato trovato. Questo può accadere anche se l'utente non ha accesso al dispositivo richiesto.
- **429** - La richiesta ha superato il limite di frequenza.

- **500** - Errore del server. Questo errore non dovrebbe verificarsi.

Per semplicità, tutte le chiamate vengono eseguite utilizzando **richieste GET**. Le richieste **POST** possono essere utilizzate al posto delle GET se necessario. In questo caso, **è obbligatorio** includere l'intestazione: `Content-Type: application/x-www-form-urlencoded` per garantire la massima compatibilità.

Integrazione Locale

La comunicazione tra il servizio di terze parti e il server è diretta e **non** richiede alcuna connessione Internet. L'applicazione di terze parti può autenticarsi direttamente con il server Thinknx. Aggiungendo `&auth=basic` o `&auth=digest` alla richiesta API, come mostrato nell'esempio seguente, l'autenticazione può avvenire tramite **Simple Auth** o **Digest Auth**. Il server risponderà al proprio IP sulla porta **5051**. Tutte le chiamate API seguono questa forma:

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

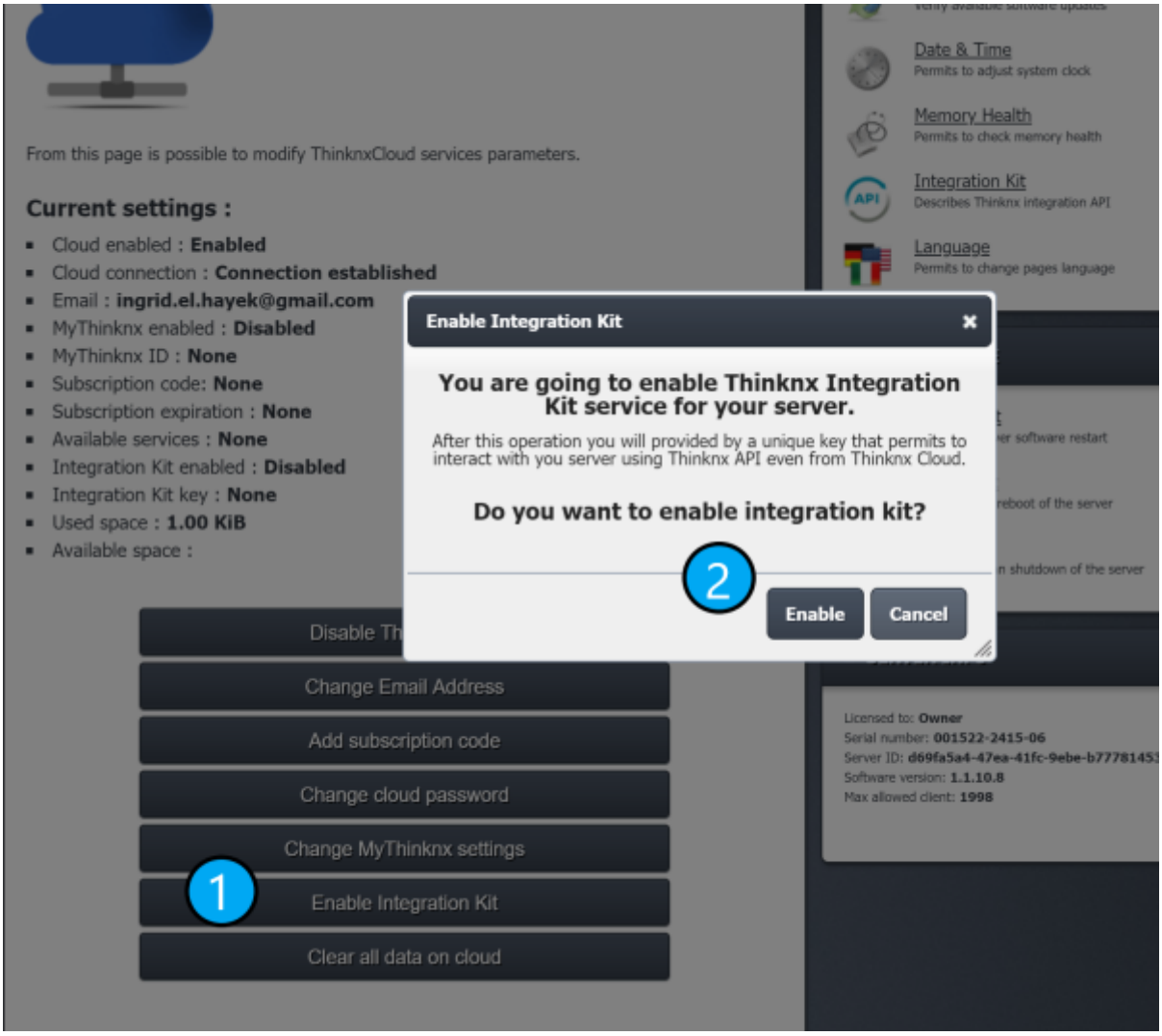
Se lo schema URL proposto non funziona per la tua integrazione, puoi usare questa sintassi alternativa:

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

⚠ **Nota**: in questo caso, il server risponderà solo alle richieste **HTTP non criptate (senza HTTPS)**.

Integrazione Basata su Cloud

I server Thinknx sono connessi a Internet tramite il servizio **Thinknx Cloud**. Il servizio **Thinknx Cloud** permette di accedere ai server indipendentemente dalla struttura di rete e di controllarli da remoto. Ogni server Thinknx è identificato dal proprio **numero di serie** e da una chiave chiamata **Integration Kit Key**. Questa chiave è un codice alfanumerico utilizzato per instradare le chiamate dal server Cloud al corrispondente server Thinknx. La chiave deve essere **abilitata e generata** dalle pagine web del server Thinknx nella scheda "Server", alla pagina **Thinknx Cloud**. Fai clic su **"Enable Integration Kit"** per generare la chiave.



Abilita la chiave di Integration Kit

La chiave permette di connettersi direttamente al server **senza autenticazione aggiuntiva**; per questo motivo, deve essere mantenuta **segreta** e bisogna considerare le seguenti precauzioni quando si scrive codice o si utilizza l'integrazione basata su cloud:

1. **Non** esporre la chiave nei moduli o direttamente nelle pagine web.
2. **Non** utilizzare la chiave in software che girano lato client (ad es. JavaScript nel browser).
3. **Utilizzare solo HTTPS** per tutte le chiamate al cloud.

Tutte le chiamate API in modalità cloud seguono questa forma:

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

Le chiamate HTTP verranno automaticamente **reindirizzate a HTTPS**.

Integrazione On-Premise

Questo tipo di integrazione è utile quando è necessario mantenere i servizi operativi **anche senza connessione Internet** e quando è importante poter accedere contemporaneamente a **più server**.

In questo caso, tutto il software del servizio Thinknx Cloud può essere trasferito su una **macchina locale** installata on-premise. I server Thinknx si connetteranno a questa macchina locale, e anche il servizio di terze parti comunicherà con essa utilizzando i metodi descritti nell'integrazione basata su cloud.

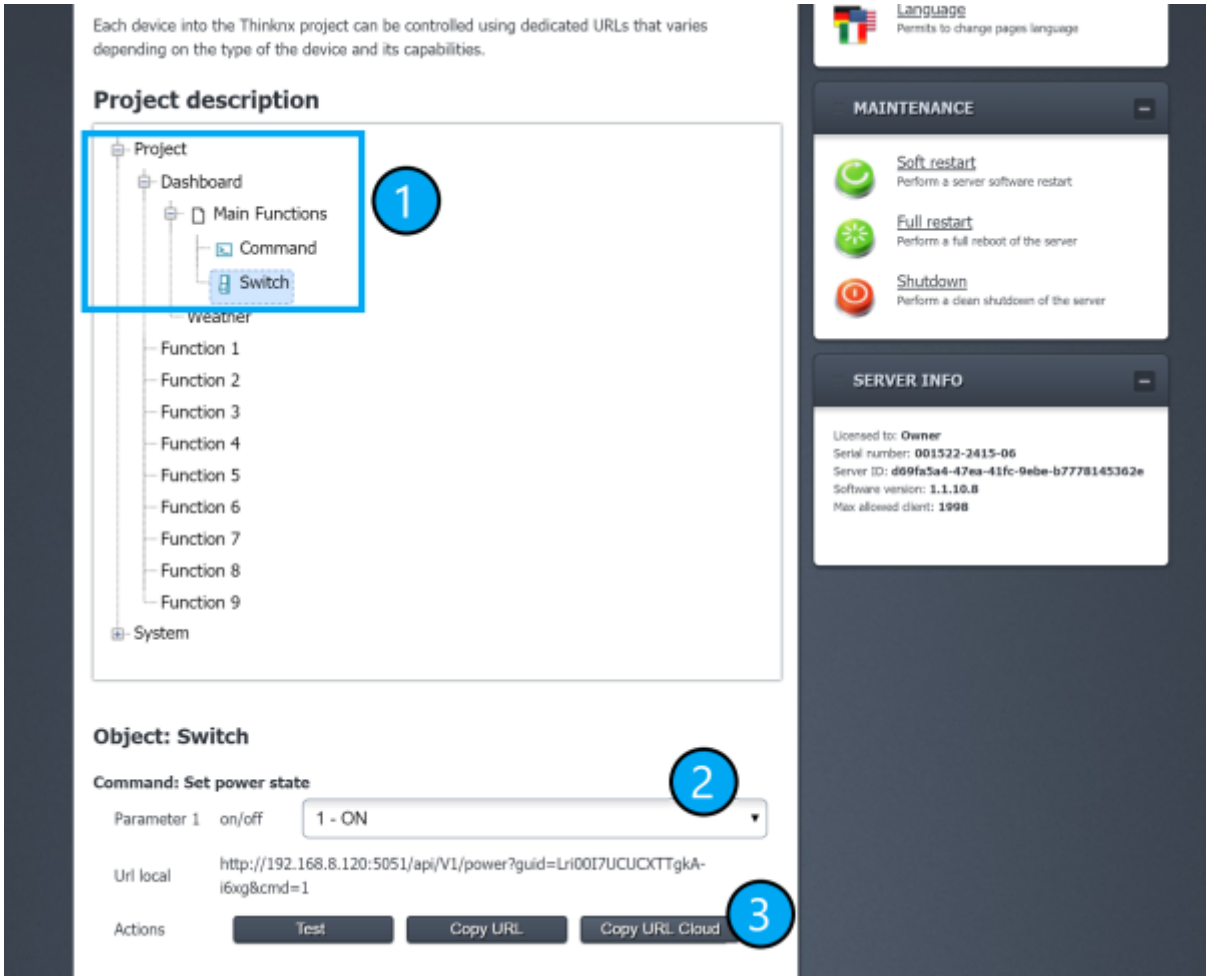
Capacità dei Dispositivi

Ogni tipo di dispositivo ha capacità specifiche che possono essere utilizzate tramite l'API. Non tutte le capacità sono disponibili per tutti i dispositivi. Di seguito, l'elenco delle capacità disponibili:

- **Luci, interruttori e lampadine**
  - **Power Controller** - Accende o spegne una luce e ne rileva lo stato
  - **PowerLevel Controller** – Imposta o legge il livello di potenza di un endpoint
2. **Tapparelle, tende e lamelle**
   - **Movement Controller** – Muove su/giù o ferma un endpoint
   - **Position Controller** – Imposta o legge la posizione di un endpoint
   - **Angle Controller** – Imposta, legge o modifica l'angolo di un endpoint
3. **Valori analogici**
   - **Value Controller** – Imposta o legge un valore su un endpoint con un intervallo continuo
4. **Termostati**
   - **Actual Temperature Controller** – Legge la temperatura attuale di un endpoint
   - **Setpoint Controller** – Imposta o legge la temperatura desiderata di un endpoint
   - **Modality Controller** – Imposta o legge la modalità di funzionamento di un endpoint
   - **Chronotable Controller** – Abilita o disabilita la programmazione cronologica di un endpoint
5. **Unità HVAC e ventilatori**
   - **Power Controller** – Accende/spegne un endpoint e ne rileva lo stato
   - **Setpoint Controller** – Imposta o legge la temperatura desiderata di un endpoint
   - **Modality Controller** – Imposta o legge la modalità di funzionamento di un endpoint
   - **FanSpeed Controller** – Imposta o legge la velocità della ventola
   - **Scene**
     - **Launch Controller** – Richiama una scena registrata dall'utente o ne interrompe l'esecuzione
   2. **Pulsanti generici**
      - **Trigger Controller** – Attiva le azioni associate a un endpoint

   Nella scheda "Server", pagina **"Integration Kit"** del Thinknx Server, è possibile vedere l'intera struttura del progetto in esecuzione sul server. Navigando fino all'oggetto desiderato, è possibile:

   1. Vedere i comandi supportati
   2. Simulare i comandi
   3. Copiare l'URL per inviare il comando da remoto.



Struttura del progetto con descrizione API

Eventi Asincroni (prossimamente)

Il servizio Thinknx potrà fornire eventi ai servizi di terze parti grazie ai **REST hooks**. Il sistema di terze parti potrà definire **callback HTTP** che verranno attivate da eventi predefiniti che si verificano sui dispositivi. Le callback HTTP potranno essere associate:

1. All'intero server
2. A un singolo dispositivo
3. A un controller specifico del dispositivo.

URL delle API

Di seguito, l'elenco degli **URL API** per ogni tipo di dispositivo e i relativi controller.

Luci, interruttori e lampadine

Power Controller

Descrizione	**Accende o spegne un interruttore definito**
URL	***api/v1/power***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo della luce da controllare **cmd=*[0 o 1]*** dove 0 spegne il dispositivo mentre 1 lo accende Parametro opzionale: **fb** Se presente, il server attenderà il feedback dal dispositivo prima di rispondere.
Data Params	non richiesti
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “power”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`
	In caso di richiesta di feedback (`fb` presente):
	`{`
	`“namespace” : “power”,`
	`“guid”: [device_identifier],`
	`“status” : [0 o 1],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” rappresenta lo stato attuale della luce (0=OFF / 1=ON)

Error Response Code	400 o 404

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

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

Power Controller (Stato)

Descrizione	**Legge lo stato ON/OFF di un interruttore definito**
URL	***api/v1/powerstatus***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo della luce da controllare
Data Params	non richiesti
Success Response Code	200 (OK)

Descrizione	**Legge lo stato ON/OFF di un interruttore definito**

Success Response Content	{
	“namespace” : “powerstatus”,
	“guid”: [device_identifier],
	“status” : [0 o 1],
	“comm_obj” : [obj_num]
	}
	“status” rappresenta lo stato attuale della luce (0=OFF / 1=ON)

Error Response Code	400 o 404
Error Response Content	{ “error” : “device not found”} oppure { “error” : “params errors” } oppure { “error” : “impossible to execute”}
Example	**/api/V1/powerstatus?guid=1IzPszHWAEuvzxtXBMkzWQ**

Power Level Controller

Descrizione	**Imposta il livello di potenza di un dispositivo dimmerabile**
URL	***api/v1/powerlevel***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo della luce da controllare **cmd=*[0-100]*** dove il numero rappresenta il livello di potenza in percentuale Parametro opzionale: **fb** Se presente, il server attenderà il feedback dal dispositivo prima di rispondere.
Data Params	non richiesti
Success Response Code	200 (OK)

Success Response Content	{
	“namespace” : “powerlevel”,
	“guid”: [device_identifier],
	“comm_obj” : [obj_num]
	}
	In caso di richiesta di feedback (`fb` presente):
	{
	“namespace” : “powerlevel”,
	“guid”: [device_identifier],
	“status” : [0-100],
	“comm_obj” : [obj_num]
	}
	“status” rappresenta il livello di potenza attuale in percentuale

Error Response Code	400 o 404

Descrizione	**Imposta il livello di potenza di un dispositivo dimmerabile**

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

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

Tapparelle, tende e lamelle

Movement Controller / Su o Giù

Descrizione	**Muove su o giù una tapparella o tenda definita**
URL	***api/v1/updown***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo della tapparella o tenda da controllare **cmd=*[0 o 1]*** dove 0 muove SU il dispositivo e 1 lo muove GIÙ
Data Params	non richiesti
Success Response Code	200 (OK)

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

Error Response Code	400 o 404

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

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

Movement Controller / Stop

Descrizione	**Ferma il movimento di una tapparella o tenda definita**
URL	***api/v1/stop***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo della tapparella o tenda da controllare **cmd=1** comando necessario per confermare lo stop
Data Params	non richiesti

Descrizione	**Ferma il movimento di una tapparella o tenda definita**
Success Response Code	200 (OK)

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

Error Response Code	400 o 404
Error Response Content	{ “error” : “device not found”} oppure { “error” : “params errors” } oppure { “error” : “impossible to execute”}
Example	**/api/V1/stop?guid=1IzPszHWAEuvzxtXBMkzWQ&cmd=1**

Position Controller / Altezza

Descrizione	**Imposta l'altezza di una tapparella o tenda definita**
URL	***api/v1/heightlevel***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo della tapparella o tenda da controllare **val=*[0-100]*** dove il numero rappresenta l’altezza in percentuale Parametro opzionale: **fb** Se presente, il server attenderà il feedback dal dispositivo prima di rispondere.
Data Params	non richiesti
Success Response Code	200 (OK)

Success Response Content	{
	“namespace” : “heightlevel”,
	“guid”: [device_identifier],
	“comm_obj” : [obj_num]
	}
	In caso di richiesta di feedback (`fb` presente):
	{
	“namespace” : “heightlevel”,
	“guid”: [device_identifier],
	“status” : [0-100],
	“comm_obj” : [obj_num]
	}
	“status” rappresenta l'altezza attuale in percentuale

Error Response Code	400 o 404

Descrizione	**Imposta l'altezza di una tapparella o tenda definita**

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

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

Position Controller / Altezza (Stato)

Descrizione	**Legge l'altezza attuale di una tapparella o tenda definita**
URL	***api/v1/heightlevelstatus***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo della tapparella o tenda da controllare
Data Params	non richiesti
Success Response Code	200 (OK)

Success Response Content	{
	`“namespace” : “heightlevelstatus”,`
	`“guid”: [device_identifier],`
	`“status” : [0-100],`
	`“comm_obj” : [obj_num]`
	}
	“status” rappresenta l'altezza attuale in percentuale

Error Response Code	400 o 404

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

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

Angle Controller / Angolo Lamelle

Descrizione	**Imposta l'angolo delle lamelle di una tapparella**
URL	***api/v1/blindangle***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo della tapparella da controllare **cmd=*[0-100]*** dove il numero rappresenta l'angolo in percentuale Parametro opzionale: **fb** Se presente, il server attenderà il feedback dal dispositivo prima di rispondere.
Data Params	non richiesti

Descrizione	**Imposta l'angolo delle lamelle di una tapparella**
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “blindangle”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`
	In caso di richiesta di feedback (`fb` presente):
	`{`
	`“namespace” : “blindangle”,`
	`“guid”: [device_identifier],`
	`“status” : [0-100],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” rappresenta l'angolo attuale in percentuale

Error Response Code	400 o 404

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

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

Termostati

Thermostat Controller / Temperatura Attuale

Descrizione	**Legge la temperatura misurata attualmente per un termostato definito**
URL	***api/v1/actualtemp***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo del termostato da controllare
Data Params	non richiesti
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “actualtemp”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_temperature],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” rappresenta la temperatura attuale misurata

Error Response Code	400 o 404

Descrizione	**Legge la temperatura misurata attualmente per un termostato definito**

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

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

Thermostat Controller / Setpoint

Descrizione	**Imposta la temperatura desiderata per un termostato**
URL	***api/v1/setpoint***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo del termostato da controllare **cmd=*[nuovo_setpoint]*** dove “nuovo_setpoint” rappresenta la temperatura desiderata (formato decimale con ‘.’ come separatore) Parametro opzionale: **fb** Se presente, il server attenderà il feedback dal dispositivo prima di rispondere.
Data Params	non richiesti
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “setpoint”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`
	In caso di richiesta di feedback (`fb` presente):
	`{`
	`“namespace” : “setpoint”,`
	`“guid”: [device_identifier],`
	`“status” : [actual_setpoint],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” rappresenta la temperatura desiderata attuale

Error Response Code	400 o 404

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

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

HVAC - Riscaldamento e Ventilazione

HVAC Controller / Accensione e Spegnimento

Descrizione	**Accende o spegne un'unità HVAC definita**
URL	***api/v1/hvacpower***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo dell'unità HVAC da controllare **cmd=*[0 o 1]*** dove 0 spegne l’unità e 1 la accende Parametro opzionale: **fb** Se presente, il server attenderà il feedback dal dispositivo prima di rispondere.
Data Params	non richiesti
Success Response Code	200 (OK)

Success Response Content	`{`
	`“namespace” : “hvacpower”,`
	`“guid”: [device_identifier],`
	`“comm_obj” : [obj_num]`
	`}`
	In caso di richiesta di feedback (`fb` presente):
	`{`
	`“namespace” : “hvacpower”,`
	`“guid”: [device_identifier],`
	`“status” : [0 o 1],`
	`“comm_obj” : [obj_num]`
	`}`
	“status” rappresenta lo stato attuale dell’unità HVAC (0=OFF / 1=ON)

Error Response Code	400 o 404

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

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

Scene

Scene Controller / Avvio e Stop di una Scena

Descrizione	**Avvia o interrompe una scena registrata**
URL	***api/v1/scenelaunch***

Descrizione	**Avvia o interrompe una scena registrata**
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo della scena da avviare o interrompere **cmd=*[0 o 1]*** dove 0 interrompe l'esecuzione di una scena in corso, mentre 1 la avvia **Attenzione:** Una scena in esecuzione è una scena con operazioni che non vengono eseguite istantaneamente (ad es. pause tra le operazioni). Le scene non hanno uno stato ON/OFF, ma rappresentano una macro (una serie di operazioni da eseguire).
Data Params	non richiesti
Success Response Code	200 (OK)

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

Error Response Code	400 o 404

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

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

Generic Button

Generic Button Controller / Attivazione Operazione

Descrizione	**Attiva l’operazione associata a un pulsante generico**
URL	***api/v1/trigger***
URL Params	**guid=*device_identifier*** dove “device_identifier” è l'identificativo del pulsante generico da controllare
Data Params	non richiesti
Success Response Code	200 (OK)

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

Descrizione	**Attiva l'operazione associata a un pulsante generico**
Error Response Code	400 o 404

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

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

System Commands

System Controller / Invio Telegramma KNX DPT1

Descrizione	**Invia un telegramma KNX con datatype DPT1 (bit)**
URL	***api/v1/system/sendKNXbit***
URL Params	**group=*[KNX_group_address]*** dove "KNX_group_address" è l'indirizzo di gruppo KNX su cui inviare il valore **value=*[0 o 1]*** valore da inviare sul bus KNX all'indirizzo di gruppo specificato
Data Params	non richiesti
Success Response Code	200 (OK)

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

Error Response Code	400 o 404

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

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

System Controller / Invio Telegramma KNX DPT5

Descrizione	**Invia un telegramma KNX con datatype DPT5 (1 byte senza segno)**
URL	***api/v1/system/sendKNXbyte***
URL Params	**group=*[KNX_group_address]*** dove "KNX_group_address" è l'indirizzo di gruppo KNX su cui inviare il valore **value=*[0-255]*** valore da inviare sul bus KNX all'indirizzo di gruppo specificato

Descrizione	**Invia un telegramma KNX con datatype DPT5 (1 byte senza segno)**
Data Params	non richiesti
Success Response Code	200 (OK)

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

Error Response Code	400 o 404
Error Response Content	{ “error” : “device not found”} oppure { “error” : “params errors” } oppure { “error” : “impossible to execute”}
Example	**/api/V1/system/sendKNXbyte?group=1/0/1&value=30**

System Controller / Invio Telegramma KNX DPT9

Descrizione	**Invia un telegramma KNX con datatype DPT9 (2 byte in virgola mobile)**
URL	***api/v1/system/sendKNXfloat2bytes***
URL Params	**group=*[KNX_group_address]*** dove “KNX_group_address” è l’indirizzo di gruppo KNX su cui inviare il valore **value=*[new_value]*** valore da inviare sul bus KNX all'indirizzo di gruppo specificato
Data Params	non richiesti
Success Response Code	200 (OK)

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

Error Response Code	400 o 404
Error Response Content	{ “error” : “device not found”} oppure { “error” : “params errors” } oppure { “error” : “impossible to execute”}
Example	**/api/V1/system/sendKNXfloat2bytes?group=1/0/1&value=85.235**

System Controller / Stato del Server

Descrizione	**Ottiene le informazioni principali dal server**
URL	***api/v1/system/serverStatus***
URL Params	non richiesti
Data Params	non richiesti
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]`
	`}`
	Dove:
	software_version = versione del firmware del server (es. “xx.xx.xx.xx”)
	process_uptime = tempo trascorso dall’ultimo riavvio del software del server (“X giorni X ore X min”)
	server_uptime = tempo trascorso dall’ultimo riavvio completo (“X giorni X ore X min”)
	config_time = timestamp della configurazione attualmente in esecuzione (“hh:mm:ss YYYY-MM-DD”)
	connected_clients = numero di client attualmente connessi
	connected_pbx_users = identificatori degli utenti PBX attualmente connessi, separati da virgole

Error Response Code	400 o 404

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

Example	**/api/V1/system/serverStatus**