überall. schnell. laden.

Die flexible Wallbox.



Custom API

nicht nativ unterstützte Geräte integrieren

Version 1.5

1. Custom API

Übersicht des Webservers
Dieses Skript implementiert einen grundlegenden Webserver, der auf eingehende Verbindungen lauscht und auf GET-Requests mit JSON-Daten antwortet. Der NRGkick kommuniziert mit diesem Webserver, um Gerätewerte und Informationen abzurufen.

Anforderungen
Bitte stelle sicher, dass Python auf deinem System installiert ist. Das Skript verwendet die Bibliotheken socket und json, welche standardmäßig in Python enthalten sind.

Konfiguration
Befülle bitte die folgenden Parameter in Custom_API_Webserver.py mit deinen Gerätedaten:

Über die Custom API ist es möglich, aktuell nicht unterstützte Geräte bzw. Hersteller einzubinden. Dazu müssen die Gerätedaten abgefragt und mittels Webserver an den NRGkick kommuniziert werden. Das Python-Skript Custom_API_Webserver.py wird dabei von DiniTech zur Verfügung gestellt.

1. deviceValues: In dieser JSON-Struktur definierst du deine Gerätewerte. Dabei können Wechselrichter, Smart Meter, Batterien und Smart Loads angegeben werden. Du kannst bis zu fünf verschiedene Geräte (mit Indizes von 0-4) einbinden. Batterie und Smart Loads können jedoch nur eine hinzugefügt werden.

```json
{
  "device_values": {
    "inverters": {
      "0": {
        "inv_power_ac_w": 2225,          #Set current ac inverter power in W
        "pv_energy_day_wh": 8750,        #Set day energy generation of the inverter in Wh
        "pv_energy_total_wh": 346890,    #Set total energy generation of the inverter in Wh
        "battery_connected": True        #Set if configured battery is connected to this inverter
      }
    },
    "energy_meters": {
      "0": {
        "grid_power_w": -542             #Set current grid power in W (-… feed-in, +… consumption)
      }
    },
    "batteries": {
      "0": {
        "bat_power_w": -749,             #Set current battery power in W (-… charge, +… discharge)
        "soc": 78.2,                     #Set current battery soc in %
        "mode": 0                        #Set current battery mode (0-normal)
      }
    },
    "smart_loads": {
      "0": {
        "sl_power_w": 1241,              #Set current smart load power in W
        "temp": 46.8,                    #Set current smart load temperature in °C
        "mode": 1                        #Set current smart load mode
      }
    },
    "load_power_w": 1683                 #Set current house load in W
  }
```

1. Custom API

```json
{
  "device_info": {
    "inverters": {
      "0": {
        "name": "Testinverter",        #Set name of inverter
        "model": "Testmodel",          #Set model of inverter
        "serial": "0123456789",        #Set serial number of inverter
        "max_dc_w": 5500,              #Set maximum / connected dc power in W
        "max_ac_w": 5000               #Set maximum ac power in W
      }
    },
    "energy_meters": {
      "0": {
        "name": "Testmeter",           #Set name of smart meter
        "model": "Testmodel",          #Set model of smart meter
        "serial": "0123456789",        #Set serial number of smart meter
        "location": 1                  #Set meter location
      }
    },
    "batteries": {
      "0": {
        "name": "Testbatterie",        #Set name of battery
        "model": "Testmodel",          #Set model of battery
        "serial": "0123456789",        #Set serial number of battery
        "capacity_wh": 4700.           #Set capacity of batter in Wh
      }
    },
    "smart_loads": {
      "0": {
        "name": "Testsmartload",       #Set name of smart load
        "model": "Testmodel",          #Set model of smart load
        "serial": "0123456789"         #Set serial number of smart load
      }
    }
  }
}
```

2. deviceInfo: In dieser JSON-Struktur definierst du deine Geräteinformationen. Dabei können Wechselrichter, Smart Meter, Batterien und Smart Loads angegeben werden. Du kannst pro Kategorie bis zu fünf verschiedene Geräte (mit Indizes von 0-4) bereitstellen.
3. webserverHost: Lege die IP-Adresse oder den Hostnamen fest, auf dem der Webserver ausgeführt werden soll.
4. webserverPort: Gib den Port an, auf dem der Webserver auf eingehende Verbindungen lauschen soll. Wähle einen verfügbaren Port, z. B. 1000.

Eklärung des Skripts

1. Bibliotheken importieren: Das Skript importiert die erforderlichen Bibliotheken socket für die Netzwerkkommunikation und json für die JSON-Verarbeitung.
2. Gerätewerte und Geräteinformationen: Das Skript enthält Beispiele für die beiden Variablen deviceValues und deviceInfo. Diese JSON-Strukturen werden mit den von Ihnen definierten Daten vom Webserver an den NRGkick kommuniziert.
3. Socket-Konfiguration: Das Skript erstellt einen Socket, bindet ihn an den konfigurierten Host und Port und lässt ihn auf eingehende Verbindungen hören.
4. Handling von Verbindungen: Das Skript tritt in eine Schleife ein, um eingehende Clientverbindungen zu behandeln. Für jede Verbindung gilt:
    a. Die Request-Daten werden empfangen und von Bytes in einen String dekodiert.
    b. Die Request-Header werden aufgeteilt, um den Anfragetyp (z.B. GET) und den Pfad (z.B. /api/v1/values.json) zu extrahieren.
    c. Das Skript antwortet basierend auf dem Request-Typen und dem Pfad. Es behandelt GET Requests für /api/v1/values.json und /api/v1/info.json.
    d. Wenn der Request-Pfad nicht erkannt wird, antwortet das Skript mit einem 404-Fehler.
    e. Wenn der Request-Typ nicht GET ist, antwortet das Skript mit einem 500-Fehler.

1. Custom API

5. Response Handling: Das Skript erstellt eine HTTP-Response mit entsprechenden Headern und den JSON-Daten, welche vom NRGkick interpretiert werden.
6. Verbindung schließen: Nach dem Senden der Response wird die Verbindung geschlossen.
7. Socket aufräumen: Sobald die Schleife beendet ist (z. B. durch Beenden des Skripts), wird der Server-Socket geschlossen.

Ausführen des Skripts

1. Konfiguriere die Variablen deviceValues, deviceInfo, webserverHost und webserverPort entsprechend Ihren Anforderungen.
2. Führe das Skript mit Python aus. Öffne ein Terminal oder eine Eingabeaufforderung, navigiere zum Verzeichnis mit dem Skript und führe folgenden Befehl aus:

Terminal

```
python3 Custom_API_Webserver.py
```

3. Das Skript startet den Webserver auf dem angegebenen Host und Port. Folgend wird der Status des Servers angezeigt.

2. Kontaktmöglichkeiten/Support



Solltest du weitere Fragen haben oder Hilfe benötigen, kontaktiere uns gerne telefonisch oder per E-Mail:

DiniTech GmbH
+43 664 537 62 51
office@nrgkick.com

Support
+43 3116 21777 223
support@nrgkick.com