API Dokumentation

Die Dokumentation der überarbeiteten Version der SmartDog Portal API (JSON)

- Allgemeines
- Allgemeine API-Aufrufe
  - getApiKey
  - getSmartDogs
- Photovoltaik-Daten
  - getInverters
  - getStringData
  - getPhotovoltaicBorders
  - getStringDayData
  - getStringMonthData
  - getStringYearData
- Sensordaten
  - getSensors
  - getSensorData
- Zählerdaten
  - getCounters
  - getCounterData
  - getCountersCount
- Antwortformat und Fehler
  - Antwortformat und Fehler
- API-Nutzung & Beispiele

- Javascript - Beispielhafte Anfrage
- Php - Beispielhafte Anfrage
- Windows Terminal - Beispielhafte Anfrage

Allgemeines

Revision History

Name	Datum	Grund für Änderungen
Jakob Hütter	24.02.2025	Erstellung der Dokumentation

Vertraulichkeitserklärung

Der Inhalt dieses Dokuments ist vertraulich und ausschließlich für die adressierte Person bestimmt. Eine Weitergabe an Dritte ist ohne vorherige Zustimmung des Geschäftsführers der ecodata solutions GmbH strengstens untersagt.

Es ist nicht gestattet, dieses Dokument ganz oder teilweise zu reproduzieren, zu verbreiten oder dessen Inhalte offenzulegen.

Die Informationen in diesem Dokument wurden nicht unabhängig überprüft und stellen keine umfassende oder vollständige Darstellung aller relevanten Informationen dar. ecodata solutions GmbH sowie der Geschäftsführer, Mitarbeiter oder Berater übernehmen keine Haftung für die Richtigkeit oder Vollständigkeit der bereitgestellten Informationen.

Jegliche Vervielfältigung, Verbreitung, Modifikation oder Veröffentlichung dieser Materialien ist strengstens untersagt.

Referenzen

- SmartDog®: www.smart-dog.eu

- SmartDog API: apiv2.smart-dog.eu
- ecodata GmbH: www.eco-data.de

Dokumentenspeicherort

Dieses Dokument kann unter folgender URL gefunden werden:
https://anleitung.smart-dog.eu/books/api-dokumentationn

© 2025 ecodata solutions GmbH

SmartDog Web API

Die API

Die SmartDog API ist eine serverseitige API, die es Benutzern ermöglicht, direkt auf die Daten ihrer PowerDog-Geräte zuzugreifen.

JSON API-Kommunikation

Die PowerDog API nutzt das JSON-Format für Anfragen und Antworten. API-Aufrufe erfolgen über HTTP-POST-Requests mit einem JSON-Request-Body, der die erforderlichen Parameter enthält. Die Antwort der API wird ebenfalls im JSON-Format zurückgegeben.

Die API ist unter folgender URL erreichbar:
➡ http://apiv2.smart-dog.eu/

Die API verarbeitet die JSON-Anfragen serverseitig und gibt strukturierte JSON-Daten zurück, die leicht in Anwendungen integriert werden können.

Allgemeine Nutzung

Um Daten von der API abzurufen, wird ein gültiger API-Schlüssel benötigt.

- Der API-Schlüssel ist eine eindeutige Identifikation für jeden Benutzer.
- Dieser kann über den API-Call `getApiKey(email, pass)` erhalten werden.
- Sobald der Schlüssel generiert wurde, bleibt er gültig, bis der Benutzer seine E-Mail-Adresse ändert.
- Alle weiteren API-Aufrufe verwenden den API-Schlüssel zur Authentifizierung, anstelle der ursprünglichen E-Mail und des Passworts.
- Dies bedeutet, dass einmal generierte API-Schlüssel sicher gespeichert und anstelle von Login-Daten in externen Anwendungen verwendet werden sollten.

> Jede Antwort die nicht valid = '1' enthält, kann als fehlerhaft gewertet werden

Verfügbare API-Endpunkte

1. Allgemeine Funktionen

1. `getApiKey(email,pass)`
2. `getSmartDogs(apikey)`

2. Photovoltaik-Daten

1. `getInverters(apikey, PowerDogID)`
2. `getStringData(apikey,SensorID,StringNum,UTC_TIMESTAMP_FROM,UTC_TIMESTAMP_TO)`
3. `getPhotovoltaicBorders(apikey,PowerDogID,month,year)`

4. `getStringDayData(apikey,SensorID,StringNum,day_from,day_to,month,year)`
5. `getStringMonthData(apikey,SensorID,StringNum,month_from,month_to,year)`
6. `getStringYearData(apikey,SensorID,StringNum,year_from,year_to)`

3. Sensor-Daten

1. `getSensors(apikey,PowerDogID)`
2. `getSensorData(apikey,SensorID,UTC_TIMESTAMP_FROM,UTC_TIMESTAMP_TO)`

4. Zähler-Daten

1. `getCounters(apikey,PowerDogID)`
2. `getCounterData(apikey,SensorID,UTC_TIMESTAMP_FROM,UTC_TIMESTAMP_TO)`

Antwortformat und Fehler

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

Allgemeine API-Aufrufe

getApiKey getSmartDogs

getApiKey

Beschreibung: Erstellt einen API-Schlüssel für einen Benutzer basierend auf seiner E-Mail-Adresse und seinem Passwort. Der API-Schlüssel wird für alle weiteren Anfragen zur Authentifizierung verwendet.
Endpunkt: `/api`
Methode: `POST`

Parameter:

- `email` (string) – E-Mail-Adresse des Benutzers (wird in Kleinbuchstaben umgewandelt)
- `pass` (string) – Passwort des Benutzers (wird als MD5-Hash verarbeitet)

Beispielanfrage:

```
{
 "action": "getApiKey",
 "email": "testkunde@eco-data.de",
 "pass": "test"
}
```

Beispielantwort:

```
{
 "valid": 1,
 "apikey": "6641d282073d76b625987af5141d3e2a"
}
```

Beschreibung der Antwortparameter:

- `valid` (number) – Status der Anfrage; `1` bedeutet Erfolg, `0` bedeutet Fehler.

- `apikey` (string) – Generierter API-Schlüssel für den Benutzer. Dieser Schlüssel sollte sicher gespeichert und für zukünftige API-Aufrufe verwendet werden.

Hinweise:

- Der API-Key bleibt gültig, bis der Benutzer seine E-Mail-Adresse ändert.
- Die API speichert Passwörter nicht im Klartext, sondern verarbeitet sie als MD5-Hash.
- Der API-Key ersetzt die Login-Daten und sollte sicher gespeichert werden.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

getSmartDogs

Beschreibung: Gibt eine Liste der registrierten SmartDog-Geräte des Benutzers zurück.

Endpunkt: `/api`

Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers

Beispielanfrage:

```
{
 "action": "getSmartDogs",
 "apikey": "90f47b75edc159ba8333a16ef37bd431"
}
```

Beispielantwort:

```
{
 "valid": 1,
 "powerdogs": [
  {
   "id": 11,
   "name": "Familie Hütter",
   "description": "Anlage Familie Hütter",
   "address_city": "Braunau",
   "address_country": "AUT"
  }
 ]
}
```

Beschreibung der Antwortparameter:

- `valid` (number) – Status der Anfrage; `1` bedeutet Erfolg
- `powerdogs` (array) – Liste der registrierten SmartDog-Geräte.
  - `id` (int) – Eindeutige Kennung des Geräts.
  - `name` (string) – Name des SmartDog-Geräts.
  - `description` (string) – Beschreibung oder Zusatzinformation zum Gerät.
  - `address_city` (string) – Stadt, in der das Gerät lokalisiert ist.
  - `address_country` (string) – Ländercode des Gerätestandorts.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

Photovoltaik-Daten

getInverters

Beschreibung: Ruft eine Liste der Wechselrichter (Inverters) für das angegebene SmartDog-Gerät ab.

Endpunkt: `/api`

Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers
- `PowerDogID` (number) – Eindeutige Kennung des SmartDog-Geräts

Beispielanfrage:

```
{
  "action": "getInverters",
  "apikey": "6641d282073d76b625987af5141d3e2a",
  "PowerDogID": "36789"
}
```

Beispielantwort:


```
{
  "valid": 1,
  "inverters": {
    "B1_A2": {
      "BUS": 1,
      "ADDRESS": 2,
      "Manufactor": "sma",
      "Modulfield": "1",
      "Modulfield_Name": "Haus1",
```

```
        "Monitoring": "on",
        "Capacity": "3000",
        "SerialNo": "2000049568",
        "Type": "WR25-014",
        "Strings": "1",
        "desc": "SMA",
        "StringList": {
          "1": {
            "STRING": 1,
            "Capacity": "3000"
          }
        },
        "id": "551902"
      },
      "B8_A1": {
        "BUS": 8,
        "ADDRESS": 1,
        "Manufactor": "sma",
        "Modulfield": "2",
        "Modulfield_Name": "Garage",
        "Monitoring": "on",
        "Capacity": "2000",
        "SerialNo": "1930203053",
        "Type": "SB2.0",
        "Strings": "1",
        "desc": "SMA Bus8",
        "StringList": {
          "1": {
            "STRING": 1,
            "Capacity": "2000"
          }
        },
        "id": "551903"
      }
    },
    "id": 36789
}
```


Beschreibung der Antwortparameter:

- valid (number):

  Status der API-Anfrage. Ein Wert von `1` signalisiert, dass der Aufruf erfolgreich war.

- inverters (object):

  Ein Objekt, das die Wechselrichter (Inverters) enthält. Jeder Schlüssel in diesem Objekt (z. B. `B1_A2`, `B8_A1`) repräsentiert einen spezifischen Wechselrichter.

  Für jeden Wechselrichter gelten folgende Parameter:

  - BUS (number):

    Die Busnummer, die dem Wechselrichter zugeordnet ist.

  - ADDRESS (number):

    Die Adresse des Wechselrichters im jeweiligen Bus.

  - Manufactor (string):

    Der Hersteller des Wechselrichters (z. B. "sma").

  - Modulfield (string):

    Eine Kennzeichnung oder Identifikation des Modulfelds.

  - Modulfield_Name (string):

    Der Name des Modulfelds, beispielsweise "Haus1" oder "Garage".

  - Monitoring (string):

    Der Überwachungsstatus des Wechselrichters (z. B. "on").

  - Capacity (string):

    Die Kapazität des Wechselrichters, in Watt angegeben.

  - SerialNo (string):

    Die Seriennummer des Wechselrichters.

  - Type (string):

    Die Modellbezeichnung oder der Typ des Wechselrichters.

  - Strings (string):

    Die Anzahl der zugeordneten Strings (Photovoltaik-Stränge).

- desc (string):
  Eine zusätzliche Beschreibung oder ein Hinweis zum Wechselrichter.
- StringList (object):
  Ein Objekt, das detaillierte Informationen zu den einzelnen Strings enthält. Jeder Schlüssel repräsentiert eine String-Nummer.
  - STRING (number):
    Die Nummer des Strings.
  - Capacity (string):
    Die Kapazität des jeweiligen Strings in Watt.
- id (string):
  Die eindeutige Kennung des Wechselrichters.

- id (number):
  Die eindeutige Kennung des übergeordneten Geräts (z. B. des SmartDog-Geräts), zu dem die Wechselrichter gehören.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

getStringData

Beschreibung: Ruft Daten eines spezifischen Strings (Photovoltaik-Daten) für einen Sensor im angegebenen Zeitraum ab.

Endpunkt: `/api`

Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers
- `SensorID` (number) – Eindeutige Kennung des Sensors
- `StringNum` (number) – Nummer des Strings (Photovoltaik-String)
- `UTC_TIMESTAMP_FROM` (number) – Startzeitpunkt im UTC-Timestamp-Format
- `UTC_TIMESTAMP_TO` (number) – Endzeitpunkt im UTC-Timestamp-Format

Beispielanfrage:

```
{
 "action": "getStringData",
 "apikey": "6641d282073d76b625987af5141d3e2a",
 "SensorID": "551902",
 "StringNum": "1",
 "UTC_TIMESTAMP_FROM": "1739867939",
 "UTC_TIMESTAMP_TO": "1739877939"
}
```

Beispielantwort:

```
{
   "sensor_id": 551902,
```

```
  "string_num": 1,
  "valid": 1,
  "test": "bcde",
  "datasets": {
    "1739868002.000000": {
      "PAC": "844",
      "PDC": "907",
      "UDC": "358",
      "TEMPERATURE": "29",
      "TIMESTAMP_LOCAL": 1739871602,
      "TIMESTAMP_UTC": "1739868002.000000"
    },
    "1739868302.000000": {
      "PAC": "878",
      "PDC": "944",
      "UDC": "359",
      "TEMPERATURE": "31",
      "TIMESTAMP_LOCAL": 1739871902,
      "TIMESTAMP_UTC": "1739868302.000000"
    }
  }
}
```


Beschreibung der Antwortparameter:

- sensor_id (number):

  Die eindeutige ID des Sensors, für den die Daten abgefragt wurden.

- string_num (number):

  Die Nummer des angeforderten Photovoltaik-Strings.

- valid (number):

  Gibt an, ob die Anfrage erfolgreich war (`1` = Erfolg, `0` = Fehler).

- test (string):

  Testwert oder Debug-Information (kann variieren).

- datasets (object):

  Ein Objekt mit den Zeitstempeln als Schlüssel, das Messwerte zu jedem Zeitpunkt enthält.

  - PAC (string):

    Wechselstromleistung (AC) in Watt.

  - PDC (string):

    Gleichstromleistung (DC) in Watt.

  - UDC (string):

    Gleichspannung in Volt.

  - TEMPERATURE (string):

    Temperatur in Grad Celsius.

  - TIMESTAMP_LOCAL (number):

    Zeitstempel in der lokalen Zeitzone.

  - TIMESTAMP_UTC (string):

    Zeitstempel im UTC-Format.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

getPhotovoltaicBorders

Beschreibung:
Ruft die früheste und späteste Erzeugungszeit eines bestimmten Wechselrichter-Strings für einen angegebenen Monat und ein Jahr ab.
Diese Funktion wird verwendet, um die Start- und Endzeiten des Tagesdiagramms zu bestimmen.
Es werden alle verwendeten Wechselrichter analysiert, um die kleinste und größte Zeit zu finden.

Endpunkt: `/api`
Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers, aus `getApiKey()`
- `PowerDogID` (number) – Eindeutige Kennung des PowerDog-Geräts, aus `getPowerDogs()`
- `month` (number) – Angeforderter Monat als Ganzzahl (1-12)
- `year` (number) – Angefordertes Jahr als Ganzzahl

Beispielanfrage:

```
{
   "action": "getPhotovoltaicBorders",
   "apikey": "90f47b75edc159ba8333a16ef37bd431",
   "PowerDogID": 11,
   "month": 1,
   "year": 2013
}
```

Beispielantwort:

```
{
   "valid": 1,
   "borders": {
      "min": "7",
      "max": "17"
   }
}
```


Beschreibung der Antwortparameter:

- valid (number):
  Gibt an, ob die Anfrage erfolgreich war (`1` = Erfolg, `0` = Fehler).
- borders (object):
  Enthält die früheste und späteste Erzeugungszeit in Stunden (24h-Format) für den angefragten Monat.
  - min (string):
    Die früheste Startzeit (Stunde), zu der im angegebenen Monat Strom erzeugt wurde.
  - max (string):
    Die späteste Endzeit (Stunde), zu der im angegebenen Monat Strom erzeugt wurde.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

getStringDayData

Beschreibung:
Ruft die Tageserzeugungsdaten eines spezifischen Strings für einen Sensor über einen angegebenen Zeitraum ab.
Diese Funktion liefert tägliche Verbrauchs- und Leistungswerte für einen bestimmten Wechselrichter-String.

Endpunkt: `/api`
Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers, aus `getApiKey()`
- `SensorID` (number) – Eindeutige Kennung des Sensors (Wechselrichter-ID), aus `getInverters()`
- `StringNum` (number) – String-ID (normalerweise 1-3), MMP-Tracker-ID, aus `StringList` in `getInverters()`
- `day_from` (number) – Starttag der Abfrage (1-31)
- `day_to` (number) – Endtag der Abfrage (1-31)
- `month` (number) – Angefragter Monat als Ganzzahl (1-12)
- `year` (number) – Angefragtes Jahr als Ganzzahl

Beispielanfrage:

```
{
  "action": "getStringDayData",
  "apikey": "6641d282073d76b625987af5141d3e2a",
  "SensorID": "551902",
  "StringNum": "1",
```

```
  "day_from": "1",
  "day_to": "12",
  "month": "1",
  "year": "2024"
}
```


Beispielantwort:


```
{
   "sensor_id": 551902,
   "string_num": 1,
   "valid": 1,
   "datasets": {
      "2024-1-01": {
         "WH": "4160",
         "PAC_MAX": "1189",
         "DAY": "01",
         "MONTH": 1,
         "YEAR": 2024
      },
      "2024-1-02": {
         "WH": "348",
         "PAC_MAX": "173",
         "DAY": "02",
         "MONTH": 1,
         "YEAR": 2024
      },
      "2024-1-03": {
         "WH": "2204",
         "PAC_MAX": "897",
         "DAY": "03",
         "MONTH": 1,
         "YEAR": 2024
      }
   }
}
```


Beschreibung der Antwortparameter:

- sensor_id (number):

  Eindeutige Kennung des Sensors, für den die Daten abgefragt wurden.

- string_num (number):

  Die Nummer des Photovoltaik-Strings.

- valid (number):

  Gibt an, ob die Anfrage erfolgreich war (`1` = Erfolg, `0` = Fehler).

- datasets (object):

  Enthält Tageswerte für den angeforderten Zeitraum.

  Jeder Schlüssel repräsentiert ein Datum (`YYYY-MM-DD`).

  - WH (string):

    Gesamtenergieerzeugung des Tages in Wattstunden (Wh).

  - PAC_MAX (string):

    Maximale Wechselstromleistung (AC) des Tages in Watt.

  - DAY (string):

    Der Tag innerhalb des abgefragten Zeitraums.

  - MONTH (number):

    Der Monat der Messung.

  - YEAR (number):

    Das Jahr der Messung.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

getStringMonthData

Beschreibung:
Ruft die monatlichen Erzeugungsdaten eines bestimmten Wechselrichter-Strings für einen angegebenen Zeitraum ab.
Diese Funktion liefert monatliche Verbrauchs- und Leistungswerte für einen bestimmten Wechselrichter-String.

Endpunkt: `/api`
Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers, aus `getApiKey()`
- `SensorID` (number) – Eindeutige Kennung des Sensors (Wechselrichter-ID), aus `getInverters()`
- `StringNum` (number) – String-ID (normalerweise 1-3), MMP-Tracker-ID, aus `StringList` in `getInverters()`
- `month_from` (number) – Startmonat der Abfrage (1-12)
- `month_to` (number) – Endmonat der Abfrage (1-12)
- `year` (number) – Angefragtes Jahr als Ganzzahl

Beispielanfrage:

```
{
  "action": "getStringMonthData",
  "apikey": "6641d282073d76b625987af5141d3e2a",
  "SensorID": "551902",
  "StringNum": "1",
  "month_from": "1",
  "month_to": "12",
```

```
    "year": "2023"
}
```


Beispielantwort:


```
{
    "sensor_id": 551902,
    "string_num": 1,
    "valid": 1,
    "datasets": {
        "2023-01": {
            "WH": "55147",
            "MONTH": "01",
            "YEAR": 2023
        },
        "2023-02": {
            "WH": "146404",
            "MONTH": "02",
            "YEAR": 2023
        },
        "2023-03": {
            "WH": "233747",
            "MONTH": "03",
            "YEAR": 2023
        }
    }
}
```


Beschreibung der Antwortparameter:

- sensor_id (number):

  Eindeutige Kennung des Sensors, für den die Daten abgefragt wurden.

- string_num (number):

  Die Nummer des Photovoltaik-Strings.

- valid (number):

  Gibt an, ob die Anfrage erfolgreich war (`1` = Erfolg, `0` = Fehler).

- datasets (object):

  Enthält monatliche Erzeugungswerte für den angeforderten Zeitraum.

  Jeder Schlüssel repräsentiert ein Monat (`YYYY-MM`).

  - WH (string):

    Gesamtenergieerzeugung des Monats in Wattstunden (Wh).

  - MONTH (string):

    Der Monat der Messung.

  - YEAR (number):

    Das Jahr der Messung.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

getStringYearData

Beschreibung:
Ruft die jährlichen Erzeugungsdaten eines bestimmten Wechselrichter-Strings für einen angegebenen Zeitraum ab.
Diese Funktion liefert Jahresverbrauchs- und Leistungswerte für einen bestimmten Wechselrichter-String.

Endpunkt: `/api`
Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers, aus `getApiKey()`
- `SensorID` (number) – Eindeutige Kennung des Sensors (Wechselrichter-ID), aus `getInverters()`
- `StringNum` (number) – String-ID (normalerweise 1-3), MMP-Tracker-ID, aus `StringList` in `getInverters()`
- `year_from` (number) – Startjahr der Abfrage (z. B. 2010)
- `year_to` (number) – Endjahr der Abfrage (z. B. aktuelles Jahr)

Beispielanfrage:

```
{
  "action": "getStringYearData",
  "apikey": "6641d282073d76b625987af5141d3e2a",
  "SensorID": "551902",
  "StringNum": "1",
  "year_from": "2022",
  "year_to": "2023"
}
```

Beispielantwort:


```
{
   "sensor_id": 551902,
   "string_num": 1,
   "valid": 1,
   "datasets": {
      "2022": {
         "WH": "3205440",
         "YEAR": "2022"
      },
      "2023": {
         "WH": "3066610",
         "YEAR": "2023"
      }
   }
}
```


Beschreibung der Antwortparameter:

- sensor_id (number):

  Eindeutige Kennung des Sensors, für den die Daten abgefragt wurden.

- string_num (number):

  Die Nummer des Photovoltaik-Strings.

- valid (number):

  Gibt an, ob die Anfrage erfolgreich war (`1` = Erfolg, `0` = Fehler).

- datasets (object):

  Enthält jährliche Erzeugungswerte für den angeforderten Zeitraum.

  Jeder Schlüssel repräsentiert ein Jahr (`YYYY`).

  - WH (string):

    Gesamtenergieerzeugung des Jahres in Wattstunden (Wh).

- YEAR (string):

  Das Jahr der Messung.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

Sensordaten

getSensors getSensorData

getSensors

Beschreibung:

Ruft eine Liste aller Sensoren und deren Informationen für ein bestimmtes PowerDog-Gerät ab.

Diese Funktion gibt detaillierte Sensordaten zurück, darunter Typ, Name und Messbereich.

Endpunkt: `/api`

Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers, aus `getApiKey()`
- `PowerDogID` (number) – Eindeutige Kennung des PowerDog-Geräts, aus `getPowerDogs()`

Beispielanfrage:

```
{
  "action": "getSensors",
  "apikey": "6641d282073d76b625987af5141d3e2a",
  "PowerDogID": "36789"
}
```

Beispielantwort:


```
{
  "valid": 1,
  "sensors": {
    "onewire_1494659203": {
      "KEY": "onewire_1494659203",
      "Type": "Temperature",
      "Max": "100",
```

```
        "Name": "boilertemp",
        "id": "551929"
      },
      "adcsensor_1522075451": {
        "KEY": "adcsensor_1522075451",
        "Type": "Global_Radiation",
        "Max": "1200",
        "Name": "radiation",
        "id": "551931"
      }
    },
    "id": 36789
}
```


Beschreibung der Antwortparameter:

- valid (number):

  Gibt an, ob die Anfrage erfolgreich war (`1` = Erfolg, `0` = Fehler).

- sensors (object):

  Ein Objekt, das eine Liste aller Sensoren enthält. Jeder Sensor hat eine eindeutige KEY-ID.

  - KEY (string):

    Eindeutige Kennung des Sensors.

  - Type (string):

    Typ des Sensors, z. B. `Temperature`, `Voltage`, `Energy`.

  - Max (string):

    Maximale Messkapazität des Sensors.

  - Name (string):

    Name des Sensors.

  - id (string):

    Interne ID des Sensors.

- id (number):

  Eindeutige Kennung des PowerDog-Geräts, zu dem die Sensoren gehören.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

getSensorData

Beschreibung:

Ruft Sensordaten oder Zählerdaten für einen bestimmten Zeitraum ab.

Die Werte werden in 5-Minuten-Intervallen geliefert.

Endpunkt: `/api`

Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers, aus `getApiKey()`
- `SensorID` (number) – Eindeutige Kennung des Sensors/Zählers, aus `getSensors()` oder `getCounters()`
- `UTC_TIMESTAMP_FROM` (number) – Startzeitpunkt im UTC-Timestamp-Format
- `UTC_TIMESTAMP_TO` (number) – Endzeitpunkt im UTC-Timestamp-Format

Beispielanfrage:

```
{
   "action": "getSensorData",
   "apikey": "6641d282073d76b625987af5141d3e2a",
   "SensorID": "551941",
   "UTC_TIMESTAMP_FROM": "1739867939",
   "UTC_TIMESTAMP_TO": "1739877939"
}
```

Beispielantwort:

```
{
   "sensor_id": 551941,
   "valid": 1,
   "datasets": {
      "1739868002": {
         "DATA": "251",
         "TIMESTAMP_UTC": 1739868002,
         "TIMESTAMP_LOCAL": 1739871602
      },
      "1739868302": {
         "DATA": "257",
         "TIMESTAMP_UTC": 1739868302,
         "TIMESTAMP_LOCAL": 1739871902
      }
}
```


Beschreibung der Antwortparameter:

- sensor_id (number):

  Eindeutige Kennung des Sensors oder Zählers, für den die Daten abgefragt wurden.

- valid (number):

  Gibt an, ob die Anfrage erfolgreich war (`1` = Erfolg, `0` = Fehler).

- datasets (object):

  Enthält die aufgezeichneten Messwerte im 5-Minuten-Intervall.

  Jeder Schlüssel stellt einen UTC-Timestamp dar.

  - DATA (string):

    Messwert des Sensors oder Zählers.

  - TIMESTAMP_UTC (int):

    Zeitstempel der Messung im UTC-Format.

  - TIMESTAMP_LOCAL (int):

    Zeitstempel der Messung in der lokalen Zeitzone.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

Zählerdaten

getCounters getCounterData

Zählerdaten

getCounters

Beschreibung:

Ruft eine Liste aller Zähler und deren Informationen für ein bestimmtes PowerDog-Gerät ab.

Diese Funktion gibt detaillierte Informationen zu den angeschlossenen Energiezählern zurück.

Endpunkt: `/api`

Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers, aus `getApiKey()`
- `PowerDogID` (number) – Eindeutige Kennung des PowerDog-Geräts, aus `getPowerDogs()`

Beispielanfrage:

```
{
  "action": "getCounters",
  "apikey": "6641d282073d76b625987af5141d3e2a",
  "PowerDogID": "36789"
}
```

Beispielantwort:


```
{
  "valid": 1,
  "counters": {
    "pv_global_1499321681": {
      "KEY": "pv_global_1499321681",
      "Type": "Energy",
      "Max": 5000,
```

```
        "Name": "PV",
        "Hardware": "pv_global",
        "Medium": "ELECTRIC",
        "id": "551904"
      },
      "buscounter_1579943663": {
        "KEY": "buscounter_1579943663",
        "Type": "Energy",
        "Max": "5000",
        "Name": "GRIDIN",
        "Hardware": "buscounter",
        "Medium": "ELECTRIC",
        "id": "551911"
      }
    },
    "id": 36789
}
```


Beschreibung der Antwortparameter:

- valid (number):

  Gibt an, ob die Anfrage erfolgreich war (`1` = Erfolg, nicht vorhanden = Fehler).

- counters (object):

  Ein Objekt, das eine Liste aller Zähler enthält. Jeder Zähler hat eine eindeutige KEY-ID.

  - KEY (string):

    Eindeutige Kennung des Zählers.

  - Type (string):

    Typ des Zählers, z. B. `Energy`.

  - Max (number):

    Maximale Messkapazität des Zählers.

  - Name (string):

    Name des Zählers.

- Hardware (string):

  Bezeichnung der Zählerhardware.

- Medium (string):

  Medium des Zählers (z. B. `ELECTRIC`, `MBUS_HEATMETER`).

- id (string):

  Interne ID des Zählers.

- id (number):

  Eindeutige Kennung des PowerDog-Geräts, zu dem die Zähler gehören.

getCounterData

Beschreibung:

Ruft Zählerdaten (Counter Data) oder Sensordaten für einen bestimmten Zeitraum ab.

Die Werte werden in 5-Minuten-Intervallen bereitgestellt.

Endpunkt: `/api`

Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers, aus `getApiKey()`
- `SensorID` (number) – Eindeutige Kennung des Sensors/Zählers, aus `getSensors()` oder `getCounters()`
- `UTC_TIMESTAMP_FROM` (number) – Startzeitpunkt im UTC-Timestamp-Format
- `UTC_TIMESTAMP_TO` (number) – Endzeitpunkt im UTC-Timestamp-Format

Beispielanfrage:

```
{
   "action": "getCounterData",
   "apikey": "6641d282073d76b625987af5141d3e2a",
   "SensorID": "551907",
   "UTC_TIMESTAMP_FROM": "1739867939",
   "UTC_TIMESTAMP_TO": "1739877939"
}
```

Beispielantwort:

```
{
  "sensor_id": 551907,
  "valid": 1,
  "datasets": {
    "1739868002": {
      "DATA": "1165",
      "TIMESTAMP_UTC": 1739868002,
      "TIMESTAMP_LOCAL": 1739871602
    },
    "1739868302": {
      "DATA": "1428",
      "TIMESTAMP_UTC": 1739868302,
      "TIMESTAMP_LOCAL": 1739871902
    },
  }
}
```


Beschreibung der Antwortparameter:

- sensor_id (number):

  Eindeutige Kennung des Zählers oder Sensors, für den die Daten abgefragt wurden.

- valid (number):

  Gibt an, ob die Anfrage erfolgreich war (`1` = Erfolg, `0` = Fehler).

- datasets (object):

  Enthält die aufgezeichneten Messwerte im 5-Minuten-Intervall.

  Jeder Schlüssel stellt einen UTC-Timestamp dar.

  - DATA (string):

    Erfasster Messwert des Zählers oder Sensors.

  - TIMESTAMP_UTC (int):

    Zeitstempel der Messung im UTC-Format.

  - TIMESTAMP_LOCAL (int):

    Zeitstempel der Messung in der lokalen Zeitzone.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

getCountersCount

Beschreibung:

Ruft den Zählerstand eines bestimmten Zählers oder Sensors zu einem angegebenen Zeitpunkt ab.

Dabei wird der zum angefragten UTC-Zeitpunkt passende verfügbare Zählerwert zurückgegeben.

Endpunkt: `/api`

Methode: `POST`

Parameter:

- `apikey` (string) – API-Schlüssel des Benutzers, aus `getApiKey()`
- `SensorID` (number) – Eindeutige Kennung des Sensors/Zählers, aus `getSensors()` oder `getCounters()`
- `UTC_TIMESTAMP` (number) – Zeitpunkt im UTC-Timestamp-Format, für den der Zählerstand abgefragt werden soll

Beispielanfrage:

```
{
   "action": "getCounterCount",
   "apikey": "6641d282073d76b625987af5141d3e2a",
   "SensorID": "551911",
   "UTC_TIMESTAMP": "1739862041"
}
```

Beispiel mit cURL:

```
$body = @{
   action = "getCounterCount"
   apikey = "6641d282073d76b625987af5141d3e2a"
```

```
    SensorID = "551911"
    UTC_TIMESTAMP = "1739862041"
} | ConvertTo-Json -Compress

Invoke-RestMethod `
    -Uri "https://apiv2.smart-dog.eu/index.php" `
    -Method Post `
    -ContentType "application/json" `
    -Body $body
```

Beispielantwort:

```
{
    "CounterCount": "4592390",
    "TIMESTAMP_UTC": "1739833200.000000"
}
```

Beschreibung der Antwortparameter:

- CounterCount (string):
  Zählerstand des angefragten Sensors oder Zählers zum ermittelten Zeitpunkt.
- TIMESTAMP_UTC (string):
  UTC-Zeitstempel des zurückgegebenen Zählerstands.
  Dieser kann vom angefragten `UTC_TIMESTAMP` abweichen, wenn der nächstpassende verfügbare Messzeitpunkt verwendet wird.

Nähere Informationen zu Antwortformat und Fehlermeldungen:

Antwortformat und Fehler

Antwortformat und Fehler

Alle API-Antworten werden im JSON-Format zurückgegeben. Fehler folgen einem konsistenten Schema mit: error_code (Numerischer Fehlercode) error_message (Beschreibung des Fehlers)

Antwortformat und Fehler

1. Erfolgreiche Antwort (Correct Response)

Ein erfolgreicher API-Aufruf liefert eine JSON-Antwort mit den abgefragten Daten. Das folgende Beispiel zeigt eine Antwort für die Abfrage von PowerDog-Geräten eines bestimmten Eigentümers.


```
{
   "query": "SELECT * FROM powerdog WHERE pdowner=29812;",
   "valid": 1,
   "powerdogs": [
      {
         "timezone": "Europe/Vienna",
         "id": "36789",
         "name": "Testanlage4Hütter",
         "description": "",
         "address_zip": "5280",
         "address_city": "Braunau am Inn",
         "address_country": "AUT",
         "address_street": "Mozartstraße",
         "address_no": "33",
         "longitude": "13.04096985",
         "latitude": "48.24743652",
         "published": "0",
         "owner": "29812",
         "uid": "PD2402-0019",
         "server_id": "31"
      },
      {
```

```
            "timezone": "Etc/GMT+0",
            "id": "37641",
            "name": "TestGerät",
            "description": null,
            "address_zip": "",
            "address_city": "",
            "address_country": null,
            "address_street": "",
            "address_no": "",
            "longitude": null,
            "latitude": null,
            "published": null,
            "owner": "29812",
            "uid": "0815",
            "server_id": "101"
        },
        {
            "timezone": "Europe/Vienna",
            "id": "38586",
            "name": "Testanlage5Hütter cm4s",
            "description": null,
            "address_zip": "",
            "address_city": "",
            "address_country": null,
            "address_street": "",
            "address_no": "",
            "longitude": null,
            "latitude": null,
            "published": "0",
            "owner": "29812",
            "uid": "PD2502-0033",
            "server_id": "31"
        }
    ]
}
```


Erklärung der Felder

- `query`: Die SQL-Abfrage, die der API-Server ausgeführt hat.
- `valid`: Gibt an, ob die Anfrage gültig war (`1` = gültig, wenn nicht vorhanden = ungültig)
- `powerdogs`: Liste der PowerDog-Geräte

2. Fehlerhafte Antwort (Faulty Response)

Falls eine Anfrage fehlschlägt, gibt die API eine Fehlermeldung mit einem Fehlercode zurück.

```
{
   "faultString": "Authentication failed",
   "faultCode": 100
}
```

Fehlermeldungen und Codes

Fehlercode	Beschreibung
`100`	Authentifizierung fehlgeschlagen (z. B. ungültiger `apikey`).
`101`	Keine Rechte auf den SmartDog der angegebenen ID
`102`	Kann keine Verbindung zur Datenbank herstellen
`103`	Datenbankanfrage fehlerhaft
`104`	Fehlende Parameter

Falls eine API-Anfrage fehlschlägt, sollte überprüft werden:

1. Ob der `apikey` korrekt ist.
2. Ob alle benötigten Parameter vorhanden sind.

3. Ob die Werte in der richtigen Form übergeben wurden.

> Jede Antwort die nicht valid = '1' enthält, kann als fehlerhaft gewertet werden

3. Zusätzliche Hinweise

- Alle API-Antworten werden im JSON-Format zurückgegeben.
- Fehlermeldungen enthalten eine `faultString`-Beschreibung und einen `faultCode` zur schnellen Identifikation des Problems.
- Erfolgreiche Antworten enthalten immer das Feld `valid: 1`, während fehlerhafte Antworten eine `faultString`-Nachricht enthalten.

© 2025 PowerDog API – Alle Rechte vorbehalten

API-Nutzung & Beispiele

Javascript - Beispielhafte Anfrage

Beschreibung:

Diese JavaScript-Funktion ruft Sensordaten von der SmartDog API v2 ab.

Die API liefert Sensordaten für einen bestimmten Zeitraum im JSON-Format zurück.

Programmiersprache: JavaScript (async/await)

API-URL: `https://apiv2.smart-dog.eu/index.php`

Code:

```
async function getSensorData() {
    const url = "https://apiv2.smart-dog.eu/index.php";

    const requestData = {
        action: "getSensorData",
        apikey: "6641d282073d76b625987af5141d3e2a",
        SensorID: "551941",
        UTC_TIMESTAMP_FROM: "1739867939",
        UTC_TIMESTAMP_TO: "1739877939"
    };

    try {
        const response = await fetch(url, {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
```

```
            body: JSON.stringify(requestData)
        });

        if (!response.ok) {
            throw new Error(`HTTP-Fehler! Status: ${response.status}`);
        }

        const data = await response.json();
        document.getElementById("sensorOutput").textContent = JSON.stringify(data, null, 2);
    } catch (error) {
        console.error("Fehler beim Abrufen der Sensordaten:", error);
        document.getElementById("sensorOutput").textContent = "Fehler beim Abrufen der Daten.";
    }
}
```

Beschreibung der Funktion:

1. API-Request vorbereiten:
    - Erstellt eine Anfrage an die API mit den Parametern:
        - `action`: `"getSensorData"`
        - `apikey`: `"6641d282073d76b625987af5141d3e2a"`
        - `SensorID`: `"551941"`
        - `UTC_TIMESTAMP_FROM`: `"1739867939"`
        - `UTC_TIMESTAMP_TO`: `"1739877939"`
2. Daten abrufen mit `fetch()`:
    - Sendet die Anfrage per `POST`-Methode an die API.
    - Setzt den `Content-Type`-Header auf `"application/json"`.
    - Wandelt das `requestData`-Objekt in JSON um und sendet es als `body`.
3. Fehlermanagement:

- Falls der Server keine gültige Antwort sendet (`!response.ok`), wird eine `Error`-Exception geworfen.
- Falls ein Fehler auftritt, wird dieser in der Konsole (`console.error`) ausgegeben und eine Fehlermeldung im HTML-Element mit der ID `"sensorOutput"` angezeigt.

Beispiel für eine erfolgreiche API-Antwort:


```
{
  "sensor_id": 551941,
  "valid": 1,
  "datasets": {
    "1739868002": {
      "DATA": "251",
      "TIMESTAMP_UTC": 1739868002,
      "TIMESTAMP_LOCAL": 1739871602
    },
    "1739868302": {
      "DATA": "257",
      "TIMESTAMP_UTC": 1739868302,
      "TIMESTAMP_LOCAL": 1739871902
    }
  }
}
```

Php - Beispielhafte Anfrage

Beschreibung:

Dieses PHP-Skript ruft Sensordaten von der SmartDog API v2 ab.

Die API liefert Sensordaten für einen bestimmten Zeitraum im JSON-Format zurück.

Programmiersprache: PHP

API-URL: `https://apiv2.smart-dog.eu/index.php`

Code:

```
<?php

function getSensorData() {
    $url = "https://apiv2.smart-dog.eu/index.php";

    $requestData = [
        "action" => "getSensorData",
        "apikey" => "6641d282073d76b625987af5141d3e2a",
        "SensorID" => "551941",
        "UTC_TIMESTAMP_FROM" => "1739867939",
        "UTC_TIMESTAMP_TO" => "1739877939"
    ];

    $options = [
        "http" => [
            "header"  => "Content-Type: application/json\r\n",
            "method"  => "POST",
            "content" => json_encode($requestData)
        ]
    ];
```

```
    $context  = stream_context_create($options);
    $response = file_get_contents($url, false, $context);

    if ($response === FALSE) {
        die("Fehler beim Abrufen der Sensordaten");
    }

    $data = json_decode($response, true);
    echo "<pre>" . print_r($data, true) . "</pre>";
}

getSensorData();

?>
```

Beschreibung der Funktion:

1. API-Request vorbereiten:
    - Erstellt eine Anfrage an die API mit den Parametern:
        - `action`: `"getSensorData"`
        - `apikey`: `"6641d282073d76b625987af5141d3e2a"`
        - `SensorID`: `"551941"`
        - `UTC_TIMESTAMP_FROM`: `"1739867939"`
        - `UTC_TIMESTAMP_TO`: `"1739877939"`
2. Datenabruf mit `file_get_contents()`:
    - Sendet eine `POST`-Anfrage mit JSON-Daten an die API.
    - Nutzt `stream_context_create()` zum Setzen der `Content-Type`-Header.
    - Falls die Antwort fehlschlägt (`FALSE`), gibt das Skript eine Fehlermeldung aus.
3. Antwortverarbeitung:

- Dekodiert die JSON-Antwort mit `json_decode()`.
- Gibt die Sensordaten formatiert (`print_r()`) als HTML-`<pre>`-Block aus.

Beispiel für eine erfolgreiche API-Antwort:


```
{
   "sensor_id": 551941,
   "valid": 1,
   "datasets": {
      1739868002.000000: {
         "DATA": "251",
         "TIMESTAMP_UTC": 1739868002,
         "TIMESTAMP_LOCAL": 1739871602
      },
      1739868302.000000: {
         "DATA": "257",
         "TIMESTAMP_UTC": "1739868302.000000",
         "TIMESTAMP_LOCAL": 1739871902
      }
   }
}
```


Alternative mit cURL (empfohlen für größere API-Anfragen):

```
<?php

function getSensorData() {
   $url = "https://apiv2.smart-dog.eu/index.php";

   $requestData = [
```

```
        "action" => "getSensorData",
        "apikey" => "6641d282073d76b625987af5141d3e2a",
        "SensorID" => "551941",
        "UTC_TIMESTAMP_FROM" => "1739867939",
        "UTC_TIMESTAMP_TO" => "1739877939"
    ];

    $ch = curl_init($url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, ["Content-Type: application/json"]);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($requestData));

    $response = curl_exec($ch);

    if (curl_errno($ch)) {
        die("Fehler beim Abrufen der Sensordaten: " . curl_error($ch));
    }

    curl_close($ch);

    $data = json_decode($response, true);
    echo "<pre>" . print_r($data, true) . "</pre>";
}

getSensorData();

?>
```

Diese Variante verwendet `cURL`, was für größere API-Anfragen besser geeignet ist.

Einsatz in einer HTML-Seite:

```
<button onclick="location.reload();">Sensordaten abrufen</button>
<pre><?php getSensorData(); ?></pre>
```

Beim Klicken auf den Button `"Sensordaten abrufen"` wird die PHP-Funktion ausgeführt, und die Daten erscheinen im `<pre>`-Tag.

```
```

Windows Terminal - Beispielhafte Anfrage

Beschreibung:
Dieser `cURL`-Befehl sendet eine `POST`-Anfrage an die SmartDog API v2, um Sensordaten für einen bestimmten Zeitraum abzurufen.

Tool: cURL
API-URL: `https://apiv2.smart-dog.eu/index.php`
Methode: `POST`

Code:

```
curl -X POST "https://apiv2.smart-dog.eu/index.php" \
    -H "Content-Type: application/json" \
    -d '{
        "action": "getSensorData",
        "apikey": "6641d282073d76b625987af5141d3e2a",
        "SensorID": "551941",
        "UTC_TIMESTAMP_FROM": "1739867939",
        "UTC_TIMESTAMP_TO": "1739877939"
       }'
```

Beschreibung der Parameter:

- `-X POST` → Die Anfrage wird per `POST`-Methode gesendet.

- `-H "Content-Type: application/json"` → Setzt den Header auf JSON-Daten.
- `-d '{...}'` → Enthält die zu sendenden JSON-Daten.

Beispiel für eine erfolgreiche API-Antwort:


```
{
   "sensor_id": 551941,
   "valid": 1,
   "datasets": {
      "1739868002": {
         "DATA": "251",
         "TIMESTAMP_UTC": 1739868002,
         "TIMESTAMP_LOCAL": 1739871602
      },
      "1739868302": {
         "DATA": "257",
         "TIMESTAMP_UTC": 1739868302,
         "TIMESTAMP_LOCAL": 1739871902
      }
   }
}
```


Alternative mit Ausgabe in einer Datei:

Falls du die Antwort direkt in einer Datei speichern möchtest:

```
curl -X POST "https://apiv2.smart-dog.eu/index.php" \
    -H "Content-Type: application/json" \
    -d '{
        "action": "getSensorData",
        "apikey": "6641d282073d76b625987af5141d3e2a",
        "SensorID": "551941",
        "UTC_TIMESTAMP_FROM": "1739867939",
```

```
  "UTC_TIMESTAMP_TO": "1739877939"
}' > response.json
```

Die Antwort wird dann in `response.json` gespeichert.