System monitoring

SUNNY WEBBOX RPC

User Manual

Remote Procedure Call Description
Interface an dAPI Definition

SWebBoxRPC-BEN092713 | Version 1.3

EN

Table of Contents

1 Introduction

This document specifies a standard software interface with corresponding data transfer formats for the Sunny WebBox which makes system data available to downstream consumers.

1.1 System Overview

The Sunny WebBox data logger continually records all process data of a photovoltaic system. The data are averaged over a configurable interval and buffered. They are transferred to the Sunny Portal at regular intervals for analysis and visualization.

The Sunny WebBox makes raw process data available (non-averaged spot values) via the data interface described here for external data processing systems as a basis for company-specific processing.

For this purpose, the Sunny WebBox provides a pool of service procedures (see section 6 „Service Procedures“ (14)), which can be called up via a Remote Procedure Call (RPC) protocol by the receiver using a network or an RAS connection. JavaScript Object Notation (JSON, see section 2 „JavaScript Object Notation“ (7)) is used as the data transfer format.

1.2 Acronyms and Abbreviations

API	Application Programmers' Interface
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
UDP	User Data Protocol
URL	Uniform Resource Locator
RAS	Remote Access Service
RFC	Request for Comment
RPC	Remote Procedure Call

1.3 Referenced Documents and Sources

[1]
RFC 4627: The application/json Media Type for Javascript Object Notation (JSON)
http://www.ietf.org/rfc/rfc4627.txt?number=4627

[2]
JSON-RPC 1.1 Working Draft August 2006
http://json-rpc.org/wd/JSON-RPC-1-1-WD-20060807.html

[3]
Introducing JSON
http://www.json.org

[4]
The MD5 Message Digest Algorithm
http://www.ietf.org/rfc/rfc4627.txt?number=4627

2 JavaScript Object Notation

JavaScript Object Notation (JSON) is described and explained on the Internet at http://www.json.org.

2.1 Example

The following example illustrates a device list. It defines an object which consists of the values "totalDevicesReturned" and "devices".

"totalDevicesReturned" is a number and has the value 4. The array "devices" has 2 fields each with a device object (see section 5.1 „Device Object“ (12)) which also contains nested device objects.

```
{
  "totalDevicesReturned":4,
  "devices":
  [
    {
      "key":"SCC250H9:1390148531",
      "name":"Sunny Central E1",
      "children":
      [
        {
          "key":"SCC250H9:8945",
          "name":"Sunny BFS E1",
          "children":null
        },
        {
          "key":"SMU8b004:2567",
          "name":"String Monitoring Unit E1",
          "children":null
        }
      ]
    },
    {
      "key":"SCC250H9:1390148538",
      "name":"Sunny Central E2",
      "children":
      [
        {
          "key":"SCBFS016:8956",
          "name":"Sunny BFS E2",
          "children":null
        },
        {
          "key":"SMU8b004:2534",
          "name":"String Monitoring Unit E2",
          "children":null
        }
      ]
    }
  ]
}
```

3 Procedure Conventions

All identifiers used are case-sensitive. I.e. "Power" and "power" refer to two distinct objects. All characters are transferred as Unicode in UTF 8 format.

3.1 Procedure Call (Request)

Each request consists of a serialized JSON object which contains the following compulsory elements:

- **version** – A character string which specifies the version on which the RPC is based.
- **proc** – A character string containing the name of the procedure to be called.
- **id** – A random character string (max. 16 characters) which serves to assign a response to the request.
- **format** – A character string which specifies the data transfer format of the procedure result (see section 3.2 „Returned Value (Response)“ (9)).
- **passwd** – A character string which contains the hash value of the password for the required access level (user, installer). The hash value is calculated by means of a MD5 algorithm (see [4]). Passwords are assigned in the security settings for the WebBox. If the object is not transferred, the level is automatically assumed to be the user level.
- **params** – An object whose elements are passed on to the procedure as arguments. Each parameter must be available as a named JSON object. The order is therefore not important. The number of parameters depends on the corresponding service procedure (see section 7). The entry is not used if the procedure called does not expect arguments.

3.2 Returned Value (Response)

The data transfer format of the return is determined by the format character string sent in the procedure call.

The following formats are currently available:

- JSON

3.2.1 JSON

The returned value of a procedure call consists of a serialized JSON object which contains the following compulsory objects:

- **version** – A character string which specifies the version on which the RPC is based.
- **proc** – A character string containing the name of the completed procedure.
- **id** – A character string which serves to assign the request and response. Contains the ID from the corresponding request.
- **result** – The result of the procedure implementation as a serialized JSON object. If an error occurs preventing the procedure being completed successfully, the error object is sent instead of this object.
- **error** – An object sent if an error occurs, which contains a character string describing the error which occurred. This object is not sent if the procedure was successfully executed.

3.3 Polling Interval

The period between two queries should not be less than 30 seconds.

4 Interfaces

The Sunny WebBox provides two different access methods. They differ in the amount of work required to implement them and the use of runtime resources.

4.1 RPC via UDP Stream

The procedure call is sent to the Sunny WebBox in the payload section of the UDP protocol on port 34268. Responses are also sent to port 34268.

UDP transport is relatively easy to implement on the client side and saves runtime resources. For communication beyond the limits of local networks, ports must generally be opened in the corresponding firewalls.

4.2 RPC via HTTP

Data are transferred via the Hypertext Transfer Protocol using a TCP/IP connection to the web server port which can be configured in the Sunny WebBox.

This is preset to port 80.

The URL for all requests is: http://IP-address/rpc

The IP address is the current IP address set for the Sunny WebBox. This is preset to 192.168.0.168.

Thus, the default URL is as follows: http://192.168.0.168/rpc

The procedure call is sent via HTTP POST in the body of the HTTP request as a serialized JSON object in accordance with the conventions specified in section 3.1 „Procedure Call (Request)“ (9).

This is more complex to implement on the client-side and also requires quite a high amount of resources. Communication is generally done via the standard port 80, which means that no changes must be made to firewalls which could be present.

5 Object Definitions

This section defines the structure of commonly used objects based on JSON syntax. The values of the object elements identify their data type in the description. All definitions apply accordingly for other data transfer formats.

5.1 Device Object

Describes a device within the system (e.g. Sunny Boy, Sunny Sensor Box).

```
{
  "key": "string",
  "name": "string" or null (optional),
  "channels": [array] or null (optional),
  "children": [array] or null (optional),
}
```

key:	A unique device key (e.g. "SB21TL06:2000106925").
name:	The user-defined name of the device (e.g. "Inverter left"). Specification of the element is optional. If the element is specified but no name is assigned, null is entered here.
channels:	An array of channel objects of the device. Specification of the element is optional. If the element is specified but no channels exist, null is entered here.
children:	An array of objects of hierarchically subordinate devices. Specification of the element is optional. If the element is specified but no sub-devices exist, null is entered here.

5.2 Channel object

Describes a process data or parameter channel of a device.

```
{
  "meta": "string",
  "name": "string" (optional),
  "value": "string",
  "unit": "string" (optional),
  "min": "string" (optional),
  "max": "string" (optional),
  "options": [array] (optional)
}
```

meta:	The meta name which identifies the channel uniquely (e.g. "ExtSolIrr").
name:	The translated display name (e.g. "External radiation"). Specification of the element is optional.
value:	The value of the channel (e.g. "843"). Specification of the element is optional.
unit:	The unit of the channel (e.g. "W/m^2). A null string is entered for channels without units.
min:	The minimum value a channel can assume. Specification of the element is optional.
max:	The maximum value a channel can assume. Specification of the element is optional.
options:	A list of possible values a parameter channel can assume. Specification of the element is optional.

6 Service Procedures

This section describes the structure of the service procedures available.

A brief description of the task of each procedure is provided. The structure of the request is then presented. Variable elements are represented via placeholders in capital letters. The placeholders VERSION, FORMAT and ID are not described here for each procedure as their meaning was already explained in section 3 „Procedure Conventions“ (9) and there are seldom differences in between these procedures.

6.1 RPC_GET_PLANT_OVERVIEW

6.1.1 Version 1.0

Provides an object with the following system data:

- POWER
- DAILY-YIELD
- TOTAL-YIELD
- STATUS
- ERROR

Structure:

```
{
  "version": "1.0",
  "proc": "GetPlantOverview",
  "id": "ID",
  "format": "FORMAT"
}
```

Sample request:

```
{
  "version": "1.0",
  "proc": "GetPlantOverview",
  "id": "1",
  "format": "JSON"
}
```

Sample response:

```
{
 "version": "1.0",
 "proc": "GetPlantOverview",
 "id": "1",
 "result":
 {
  "overview":
  [
   {
    "meta": "GriPwr",
    "name": "Momentanleistung",
    "value": "4250",
    "unit": "W"
   },
   {
    "meta": "GriEgyTdy",
    "name": "Tagesenergie",
    "value": "45.23",
    "unit": "kWh"
   },
   {
    "meta": "GriEgyTot",
    "name": "Gesamtenergie",
    "value": "7821",
    "unit": "kWh"
   },
   {
    "meta": "OpStt",
    "name": "Status",
    "value": "MPP",
    "unit": null
   },
   {
    "meta": "Msg",
        "name": "Fehler",
        "value": null,
        "unit": null
      }
    ]
  }
}
```

The following data is sent:

POWER = 4250 W,

DAILY-YIELD = 45.23 kWh,

TOTAL-YIELD = 7821 kWh,

Status = MPP,

no error

6.2 RPC_GET_DEVICES

6.2.1 Version 1.0

Provides a hierarchical list of all system devices recorded.

Structure:

```
{
  "version": "1.0",
  "proc": "GetDevices",
  "id": "ID",
  "format": "FORMAT"
}
```

Sample request:

```
{
  "version": "1.0",
  "proc":   "GetDevices",
  "id": "1",
  "format": "JSON"
}
```

Sample response:

```
{
 "version": "1.0",
 "proc": "GetDevices",
 "id": "1",
 "result":
 {
  "totalDevicesReturned": 6,
  "devices":
  [
   {
    "key":"SCC250H9: 1390148531",
    "name": Sunny Central E1",
    "children":
    [
     {
      "key": "SCBFS016:8945",
      "name": "Sunny BFS E1",
      "children": null
     },
     {
      "key": "SMU8b004:2567",
      "name": "String Monitoring Unit E1",
      "children": null
     }
    ]
   },
   {
    "key": "SCC250H9:1390148538",
    "name": "Sunny Central E2",
    "children":
    [
     {
      "key": "SCBFS016:8956",
      "name": "Sunny BFS E2",
          "children": null
        },
        {
          "key": "SMU8b004:2534",
          "name": "String Monitoring Unit E2",
          "children": null
        }
      ]
    }
  ]
 }
}
```

6.3 RPC_GET_PROCESS_DATA_CHANNELS

6.3.1 Version 1.0

Provides a list with the meta names of the available process data channels for a specific device type.

Structure:

```
{
 "version": "1.0",
 "proc": "GetProcessDataChannels",
 "id": "ID",
 "format": "FORMAT",
 "params":
 {
   DEVICE_KEY
 }
}
```

DEVICE_KEY:	The device key of a device for the type whose process data channels are to be provided.

Sample request:

```
{
  "version": "1.0",
  "proc": "GetProDataChannels",
  "id": "1",
  "format": "JSON",
  "params":
  {
    "device": "WR715-19:263415747"
  }
}
```

Sample response:

```
{
  "version": "1.0",
  "proc": "GetProcessDataChannels",
  "id": "1",
  "result":
  {
    "WR715-19:263415747":
    [
      "Upv-Soll",
      "h-Total",
      "Zac",
      "Status",
      "E-Total",
      "Upv-Ist",
      "Riso",
      "Uac",
      "Pac",
      "Fehler-Cnt",
      "Ipv",
      "Netz-Ein",
      "Seriennummer",
      "Fac"
      "Fehler"
    "Iac-Ist"
  ]
 }
}
```

6.4 RPC_GET_PROCESS_DATA

6.4.1 Version 1.0

Provides process data for up to 5 devices per request.

Structure:

```
{
 "version": "1.0",

 "proc": "GetProcessData",
 "id": "ID",
 "format": "FORMAT",
 "params":
 {
  "DEVICES":
  [
   {
    "key": DEVICE_KEY,
    "channels": [CHANNELS]
   }
  ]
 }
}
```

As a parameter, a list with the device keys whose process data is to be provided must be transferred. A selection of required process data can be sent to each device. If no selection is made, all process data are sent.

DEVICES:	An array which contains objects with the device keys of the devices whose process data are to be provided, as well as optional CHANNELS.
DEVICE_KEY:	The corresponding device key. See section 5.1 „Device Object“ (12).

CHANNELS:	An array which contains the meta names of the required process data. The meta names available can be determined using the RPC_GET_PROCESS_DATA_CHANNELS command.

Sample request:

```
{
 "version": "1.0",
 "proc": "GetProcessData",
 "id": "1",
 "format": "JSON",
 "params":
 {
  "devices":
  [
   {
    "key": "WR715-19:263415747",
    "channels": null
   },
   {
    "key": "WR715-19:263415748",
    "channels":
    [
     "Pac"
    ]
   }
  ]
 }
}
```

Sample response:

```
{
 "version": "1.0",
 "proc": "GetProcessData",
 "id": "1",
 "result":
 {
  "devices":
```

```
[
  {
    "key": "WR715-19:263415747",
    "channels":
    [
      {
        "meta": "E-Total",
        "name": null,
        "value": "1160.987",
        "unit": "kWh"
      },
      {
        "meta": "Fac",
        "name": null,
        "value": "49.98",
        "unit": "Hz"
      },
      {
        "meta": "Zac",
        "name": null,
        "value": "1.346",
        "unit": "Ohm"
      }
    ]
  },
  {
    "key": "WR715-19:263415748",
    "channels":
    [
      {
        "meta": "Pac",
        "name": null,
        "value": "630",
        "unit": "W"
      }
```

```
        ]
      }
    ]
  }
}
```

6.5 RPC_GET_PARAMETER_CHANNELS

6.5.1 Version 1.0

Provides a list with the meta names of the available parameter channels for a specific device type depending on the access level. The level is defined by the transfer of the MD5 hash values of the corresponding password in the request header.

Structure

```
{
  "version": "1.0",
  "proc": "GetParameterChannels",
  "id": "ID",
  "format": "FORMAT",
  "passwd" : "PASSWORD",
  "params":
  {
    "key": DEVICE_KEY
  }
}
```

PASSWORT:	The MD5 coded hash value of the password for the desired access level.
DEVICE_KEY:	The device key of a device for the type whose parameter channels are to be provided.

Sample request

```
{
  "version": "1.0",
  "proc": "GetParameterChannels",
  "id": "1",
  "format": "JSON",
  "passwd" : "a289fa4252ed5af8e3e9f9bee545c172",
  "params":
```

```
  {
    "device": "WR715-19:263415747"
  }
}
```

Sample response:

```
{
  "version": "1.0",
  "proc": "GetParameterChannels",
  "id": "1",
  "result":
  {
    "WR715-19:263415747":
    [
      "Plimit",
      "SMA-Grid-Guard",
      "SMA-SN",
      "Betriebsart",
      "Control",
      "Ripple-Ctl-Frq",
      "PowerBalancer",
      "Usoll-Konst",
      "Upv-Start",
      "Default",
      "T-Start",
      "Ripple-Ctl-Lev",
      "Storage",
      "Ripple-Ctl-Rcvr",
      "Software-SRR",
      "T-Stop",
      "Software-BFR",
      "Hardware-BFS"
    ]
  }
}
```

6.6 RPC_GET_PARAMETER

6.6.1 Version 1.0

Provides parameter values for up to 5 devices depending on the access level. The level is defined by the transfer of the MD5 hash values of the corresponding password in the request header.

Structure

```
{
  "version": "1.0",
  "proc": "GetParameter",
  "id": "ID",
  "format": "FORMAT",
  "passwd" : "PASSWORT",
  "params":
  {
    "DEVICES":
    [
      {
        "key": DEVICE_KEY,
        "channels": [CHANNELS]
      }
    ]
  }
}
```

As a parameter, a list with the device objects whose parameters are to be provided must be transferred. A selection of required parameters can be sent to each device. If no selection is made, all parameters are sent.

PASSWORT:	The MD5 coded hash value of the password for the desired access level.
DEVICES:	An array which contains device objects whose parameters are to be provided, and an optional selection of certain channels.
DEVICE_KEY:	The corresponding device key. See section 5.1 „Device Object“ (12).
CHANNELS:	An array which contains the meta names of the required process data. The meta names available can be determined using the RPC_GET_PROCESS_DATA_CHANNELS command.

Sample request

```
{
 "version": "1.0",
 "proc": "GetParameter",
 "id": "1",
 "format": "JSON",
 "passwd" : "a289fa4252ed5af8e3e9f9bee545c172",
 "params":
 {
  "devices":
  [
   {
    "key": "WR715-19:263415747"
   }
  ]
 }
}
```

Sample response:

```
{
 "version": "1.0",
 "id": "1",
 "format": "JSON",
 "proc": "GetParameter",
 "result":
 {
  "devices":
  [
   {
    "key": "WR21TL06:2000101000"
    "channels":
    [
     {
      "min": "0",
      "max": "7",
      "meta": "Betriebsart",
  "options":
  [
    "Stop",
    "Konstantspg.",
    "Mpp-Betrieb",
    "Res1",
    "Res2",
    "Res3",
    "Res4",
    "Res5"
  ],
  "value": "Mpp-Betrieb",
  "name": "Betriebsart",
  "unit": ""
},
{
  "min": "2150",
  "max": "2150",
  "meta": "Plimit",
  "value": "2150",
  "name": "Plimit",
  "unit": "W"
},
{
  "min": "0",
  "max": "4294900000",
  "meta": "SMA-SN",
  "value": "2000101000",
  "name": "SMA-SN",
  "unit": ""
},
{
  "min": "125",
  "max": "600",
  "meta": "Upv-Start",
  "value": "150",
  "name": "Upv-Start",
  "unit": "V"
},
{
  "min": "1",
  "max": "300",
  "meta": "T-Stop",
  "value": "4",
  "name": "T-Stop",
  "unit": "s"
},
{
  "min": "125",
  "max": "600",
  "meta": "Usoll-Konst",
  "value": "600",
  "name": "Usoll-Konst",
  "unit": "V"
},
{
  "min": "5",
  "max": "300",
  "meta": "T-Start",
  "value": "10",
  "name": "T-Start",
  "unit": "s"
},
{
  "min": "0",
  "max": "100",
  "meta": "Software-SRR",
  "value": "2",
  "name": "Software-SRR",
  "unit": "Version"
```

```
},
{
  "min": "0.005",
  "max": "4",
  "meta": "dFac-Max",
  "value": "0",
  "name": "dFac-Max",
  "unit": "Hz/s"
},
{
  "min": "0",
  "max": "7",
  "meta": "Storage",
  "options":
  [
    "permanent",
    "volatile",
    "Res1",
    "Res2",
    "Res3",
    "Res4",
    "Res5",
    "Res6"
  ],
  "value": "permanent",
  "name": "Storage",
  "unit": ""
},
{
  "min": "0",
  "max": "100",
  "meta": "Software-BFR",
  "value": "2",
  "name": "Software-BFR",
  "unit": "Version"
```

```
          },
          {
            "min": "0",
            "max": "100",
            "meta": "Hardware-BFS",
            "value": "1",
            "name": "Hardware-BFS",
            "unit": "Version"
          }
        ]
      }
    ]
  }
}
```

6.7 RPC_SET_PARAMETER

6.7.1 Version 1.0

Sets parameter values of up to 5 devices and sends back the device list with the corresponding current parameter values transferred in the request. A check whether every parameter value was set successfully must be carried out by the application used.

Structure

```
{
  "version": "1.0",
  "proc": "SetParameter",
  "id": "ID",
  "format": "FORMAT",
  "passwd" : "PASSWORT",
  "params":
  {
    "DEVICES":
    [
      {
        key,
        "channels": [CHANNELS]
      }
    ]
  }
}
```

As a parameter, a list with the device objects whose parameters are to be changed must be transferred. Every device object contains a list with the parameters that must be set.

The parameters are set synchronically. Thus, the response time depends on the number of parameters to be set. For the following example, the response time is about 10 seconds.

PASSWORT:	The MD5 coded hash value of the password for the desired access level.
DEVICES:	An array which contains objects with the device keys of the devices, as well as an array with the CHANNELS whose parameter values are to be set.
DEVICE_KEY:	The corresponding device key. See section 5.1 „Device Object“ (12).
CHANNELS:	An array which contains the channel objects which are to be set for the respective device. A list of the parameter channels available can be determined using the RPC_GET_PARAMETER_CHANNELS command.

Sample request:

```
{
  "version": "1.0",
  "proc": "SetParameter",
  "id": "1",
  "format": "JSON",
  "passwd": "a289fa4252ed5af8e3e9f9bee545c172",
  "params":
  {
    "devices":
    [
      {
        "key": "WR21TL06:2000101000",
        "channels":
        [
          {
            "meta": "Betriebsart",
            "value": "Mpp-Betrieb"
          }
        ]
      },
      {
        "key": "WR21TL06:2000101001",
        "channels":
        [
          {
            "meta": "Betriebsart",
            "value": "Stop"
          }
        ]
      }
    ]
  }
}
```

Sample response:

```
{
  "version": "1.0",
  "id": "1",
  "format": "JSON",
  "proc": "SetParameter",
  "result":
  {
    "devices":
    [
      {
        "key": "WR21TL06:2000101000",
        "channels":
        [
          {
            "min": "0",
            "max": "7",
            "meta": "Betriebsart",
            "options":
            [
              "Stop",
        "Konstantspg.",
        "Mpp-Betrieb",
        "Res1",
        "Res2",
        "Res3",
        "Res4",
        "Res5"
      ],
      "value": "Mpp-Betrieb",
      "name": "Betriebsart",
      "unit": ""
    }
  ]
},
{
  "key": "WR21TL06:2000101001",
  "channels":
  [
    {
      "min": "0",
      "max": "7",
      "meta": "Betriebsart",
      "options":
      [
        "Stop",
        "Konstantspg.",
        "Mpp-Betrieb",
        "Res1",
        "Res2",
        "Res3",
        "Res4",
        "Res5"
      ],
      "value": "Stop",
      "name": "Betriebsart",
            "unit": ""
          }
        ]
      }
    ]
  }
}
```

Exclusion of liability

The general terms and conditions of delivery of SMA Solar Technology AG shall apply.

The content of these documents is continually checked and amended, where necessary. However, discrepancies cannot be excluded. No guarantee is made for the completeness of these documents. The latest version is available online at www.SMA.de or from the usual sales channels.

Guarantee or liability claims for damages of any kind are excluded if they are caused by one or more of the following:

- Damages during transportation
- Improper or inappropriate use of the product
- Operating the product in an unintended environment
- Operating the product whilst ignoring relevant, statutory safety regulations in the deployment location
- Ignoring safety warnings and instructions contained in all documents relevant to the product
- Operating the product under incorrect safety or protection conditions
- Altering the product or supplied software without authority
- The product malfunctions due to operating attached or neighboring devices beyond statutory limit values
- In case of unforeseen calamity or force majeure

The use of supplied software produced by SMA Solar Technology AG is subject to the following conditions:

- SMA Solar Technology AG rejects any liability for direct or indirect damages arising from the use of software developed by SMA Solar Technology AG. This also applies to the provision or non-provision of support activities.
- Supplied software not developed by SMA Solar Technology AG is subject to the respective licensing and liability agreements of the manufacturer.

SMA Factory Warranty

The current guarantee conditions come enclosed with your device. These are also available online at www.SMA.de and can be downloaded or are available on paper from the usual sales channels if required.

Trademarks

All trademarks are recognized even if these are not marked separately. Missing designations do not mean that a product or brand is not a registered trademark.

SMA Solar Technology AG

Sonnenallee 1

34266 Niestetal

Germany

Tel. +49 561 9522-0

Fax +49 561 9522-100

www.SMA.de

E-Mail: info@SMA.de

SMA Solar Technology AG

www.SMA.de

Sonnenallee 1
34266 Niestetal, Germany
Tel.: +49 561 9522 4000
Fax: +49 561 9522 4040
E-Mail: Vertrieb@SMA.de
Freecall: 0800 SUNNYBOY
Freecall: 0800 78669269