System monitoring

SUNNY WEBBOX RPC

User Manual

Remote-Procedure-Call Description
Interfaces and API Definition

SWebBoxRPC-eng-BUS112713 | Version 1.3

US CA

SMA America, LLC
3801 N. Havana Street
Denver, CO 80239 U.S.A.

SMA Solar Technology Canada Inc.
2425 Matheson Blvd. E, 8th Floor
Mississauga, ON L4W 5K5, Canada

IMPORTANT SAFETY INSTRUCTIONS

SAVE THESE INSTRUCTIONS

This manual contains important instructions for the following products:

- Remote-Procedure-Call Description
  Interfaces and API Definition

This manual must be followed during installation and maintenance.

General warnings

General warnings

Before installing or using the product, read all of the instructions, cautions, and warnings in this manual.

Table of Contents

1 Information on this Manual 9
1.1 Validity 9
1.2 Target Group 9
1.3 Nomenclature 9
1.4 Abbreviations 9
1.5 System Overview 9
2 JavaScript Object Notation 10
2.1 Example 10
3 Procedure Conventions 12
3.1 Procedure Call (Request) 12
3.2 Return Value (Response) 12
3.2.1 JSON 13
3.3 Query Interval 13
4 Interfaces 14
4.1 RPC via UDP Stream 14
4.2 RPC via HTTP 14
5 Object Definitions 15
5.1 Device Object 15
5.2 Channel Object 16
6 Service Procedures 17
6.1 RPC_GET_PLANT_OVERVIEW 17
6.1.1 Version 1.0 17
6.2 RPC_GET_DEVICES 19
6.2.1 Version 1.0 19
6.3 RPC_GET_PROCESS_DATA_CHANNELS 21
6.3.1 Version 1.0 21

6.4 RPC_GET_PROCESS_DATA. 23
6.4.1 Version 1.0 23
6.5 RPC_GET_PARAMETER_CHANNELS 26
6.5.1 Version 1.0 26
6.6 RPC_GET_PARAMETER 28
6.6.1 Version 1.0 28
6.7 RPC_SET_PARAMETER. 33
6.7.1 Version 1.0 33

7 Sources 38

8 Contact 39

1 Information on this Manual

1.1 Validity

This manual describes a uniform software interface and the relevant data exchange formats for the Sunny WebBox.

This manual does not include any information on the devices communicating with the software interface. Information on such devices is obtainable from the device manufacturers.

1.2 Target Group

This manual is intended for software developers and end users.

1.3 Nomenclature

In this document, SMA America Production, LLC and SMA Solar Technology Canada Inc. are hereinafter referred to as SMA.

1.4 Abbreviations

Abbreviation	Description
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
PV	Photovoltaics
RPC	Remote Procedure Call

1.5 System Overview

The Sunny WebBox data logger continuously captures all the process data of a PV plant. This is then averaged over a configurable interval and cached. The data is transmitted at regular intervals to the Sunny Portal for analysis and visualization.

Via the data interface described here, the Sunny WebBox makes raw process data (unaveraged instantaneous values) available to external data processing systems for further processing.

To do this, the Sunny WebBox provides a pool of service procedures, see section 5 "Object Definitions" (page 15), which can be accessed from the remote terminal by means of a Remote Procedure Call protocol (RPC protocol) via a network or RAS connection. The data exchange format used here is the JavaScript Object Notation (JSON), see section 2 "JavaScript Object Notation" (page 10).

2 JavaScript Object Notation

A description and explanation of JavaScript Object Notation (JSON) is to be found on the website http://www.json.org.

2.1 Example

The following example shows an illustration of a device list. It defines an object made up of the values "totalDevicesReturned" and "devices".

"totalDevicesReturned" is a figure with the value 4. The "devices" array has 2 fields, each having one device object, see section 5.1 "Device Object" (page 15), which in turn contains nested device objects.

```
{
 "totalDevicesReturned": 4,
 "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
        }
      ]
    }
  ]
}
```

3 Procedure Conventions

All identifiers used are case-sensitive. This means that "Power" and "power", for instance, designate two different objects. All characters are transmitted using Unicode in UTF-8 format.

3.1 Procedure Call (Request)

Each request consists of one serialized JSON object which possesses the following obligatory members:

- **version** – a character sequence which defines the underlying RPC version.
- **proc** – a character sequence which contains the procedure to be called.
- **id** – a random character sequence (max. 16 characters) which serves to assign a response to the request.
- **format** – a character sequence defining the data exchange format of the procedure's result, see Section 3.2 "Return Value (Response)" (page 12).
- **passwd** – a character sequence comprising the hash value of the password for the desired access level (user, installer). The hash value is calculated using the MD5 algorithm (see [4]). Password allocation is performed via the security settings in the WebBox. If the object is not transmitted, the user level is automatically assumed.
- **params** – an object, whose elements are transferred to the procedure as arguments. Each parameter must be available in the form of a named JSON object. Hence, the sequence is arbitrary. The number of parameters depends on the given service procedure (see section 7). If the requested procedure does not expect any arguments, the entry is omitted.

3.2 Return Value (Response)

The data exchange format of the response is defined by the character sequence transmitted in the request.

At present, the following format is available:

- JSON

3.2.1 JSON

The response to a request consists of a serialized JSON object made up of the following compulsory objects:

- **version** – a character sequence which defines the underlying RPC version.
- **proc** – a character sequence which contains the called procedure.
- **id** – a character sequence which assigns a request to a response. Contains the ID from the respective request.
- **result** – the result of the executed procedure as a serialized JSON object. If due to an error the procedure cannot be executed successfully, the "error" object will be transmitted instead.
- **error** – an object containing a character string with a description of the error encountered. If the procedure is executed without errors, this object will not be transmitted.

3.3 Query Interval

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

4 Interfaces

The Sunny WebBox provides two different access options which differ in the implementation effort required and their runtime utilization of resources.

4.1 RPC via UDP Stream

The procedure call is transmitted to port 34268 of the Sunny WebBox in the usable data part of the UDP protocol. Responses are also sent to port 34268.

UDP transport requires a relatively low implementation effort on the client side and saves runtime resources. For communication beyond the limits of local networks, the port will normally need to be opened by the appropriate firewalls.

4.2 RPC via HTTP

Data exchange takes place by means of the Hypertext Transfer Protocol via a TCP/IP connection to the webserver port configured in the Sunny WebBox.

The default port is 80.

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

The IP address in each case is the currently configured IP address of the Sunny WebBox. The default setting is 192.168.0.168.

Hence, the default URL is the following: http://192.168.0.168/rpc

The request is transmitted via HTTP POST in the body of the HTTP request as a serialized JSON object according to the conventions established in section 3.1 ”Procedure Call (Request)” (page 12).

Both the client-side implementation effort and the resource requirements are relatively high. As a general rule, communication takes place via the standard port 80, which means that there is no need to make any changes to active firewalls.

5 Object Definitions

This section defines the structure of frequently used objects using the JSON syntax. In the description, the values of the object members define the type of data used. All definitions apply in the same way to other data exchange formats.

5.1 Device Object

Describes a device within the plant (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. "WR21TL06:2000106925").
name:	The user-definable name of the device (e.g. "inverter left"). Defining this element is optional. If the element is used but no name was defined, enter zero.
channels:	An array of channel objects of the device. Defining this element is optional. If the element is used but no channels exist, enter zero.
children:	An array of objects containing sub-devices. Defining this element is optional. If the element is used but no sub-devices exist, enter zero.

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 uniquely defines the channel (e.g., "ExtSolIrr").
name:	The translated display name (e.g., "External irradiation"). Defining this element is optional.
value:	The value of the channel (e.g., "843"). Defining this element is optional.
unit:	The unit of the channel (e.g., "W/m^2). For channels not having any unit, an empty string must be entered.
min:	The minimum possible value of a channel. Defining this element is optional.
max:	The maximum possible value of a channel. Defining this element is optional.
options:	A list with the possible values of a parameter channel. Defining this element is optional.

6 Service Procedures

This section describes the structure of the available service procedures.

For each procedure, a brief description of its task is given. This is followed by the structure of the request. Variable elements are represented by placeholders in upper-case characters. In this manual, the placeholders VERSION, FORMAT and ID are not described for each procedure, since their meaning has already been described in section 3 "Procedure Conventions" (page 12) and it is not necessary to differentiate between all procedures here.

6.1 RPC_GET_PLANT_OVERVIEW

6.1.1 Version 1.0

Returns an object with the following plant data:

- PAC
- E-TODAY
- E-TOTAL
- MODE
- 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": "current power",
    "value": "4 250",
    "unit": "W"
   },
   {
    "meta": "GriEgyTdy",
    "name": "Day energy",
    "value": "45.23",
    "unit": "kWh"
   },
   {
    "meta": "GriEgyTot",
    "name": "Total energy",
    "value": "7 821",
    "unit": "kWh"
   },
   {
    "meta": "OpStt",
    "name": "Mode",
    "value": "MPP",
    "unit": null
   },
      {
        "meta": "Msg",
        "name": "Error",
        "value": null,
        "unit": null
      }
    ]
  }
}
```

The following data was transmitted:

Pac = 4 250 W,
E-Today = 45.23 kWh,
E-total = 7 821 kWh,
Mode = MPP,
no error

6.2 RPC_GET_DEVICES

6.2.1 Version 1.0

Returns a hierarchical list of all detected plant devices.

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

Returns a list with the meta names of the available process data channels for a particular device type.

Structure:

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

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

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":
  [
   "Vpv-Soll",
   "h-Total",
   "Zac",
   "Mode",
   "E-total",
   "Vpv-Ist",
   "Riso",
   "Vac",
   "Pac",
   "Error-Cnt",
   "Ipv",
   "Power On",
   "Serial Number",
   "Fac"
   "Error"
```

```
    "Iac"
   ]
 }
}
```

6.4 RPC_GET_PROCESS_DATA

6.4.1 Version 1.0

Returns 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 parameters you should submit a list of device keys from which process data is to be returned. You can submit a selection of required process data to each device. If a selection is omitted, all process data will be transmitted.

DEVICES:	An array containing objects with the device keys of those devices from which process data is to be returned, and optional CHANNELS.
DEVICE_KEY:	The corresponding device key. See section 5.1 "Device Object" (page 15).
CHANNELS:	An array containing the meta names of the required process data. The available meta names can be identified with the command RPC_GET_PROCESS_DATA_CHANNELS.

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": "1 160.987",
          "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

Returns a list with the meta names of the available parameter channels for a particular device type, depending on access level. The level is determined by transmitting the MD5 hash value of the respective password in the request header.

Assembly:

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

PASSWORD:	The MD5-coded hash value of the password for the required access level (e.g., the user or installation password of the Sunny WebBox).
DEVICE_KEY:	The device key of a device for whose type the parameter channels are to be returned.

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",
   "Operating Mode",
   "Control",
   "Ripple-Ctl-Frq",
   "PowerBalancer",
   "Vconst-Setpoint",
   "Vpv-Start",
   "Default",
   "T-Start",
   "Ripple-Ctl-Lev",
   "Storage",
   "Ripple-Ctl-Rcvr",
   "Firmware-SRR",
   "T-Stop",
   "Firmware-BFR",
   "Hardware-BFS"
  ]
 }
}
```

6.6 RPC_GET_PARAMETER

6.6.1 Version 1.0

Returns the parameter values of up to 5 devices, depending on the access level. The level is determined by transmitting the MD5 hash value of the respective password in the request header.

Assembly

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

You should submit as parameters a list of those device objects for which the parameters are to be returned. You can submit a selection of requested parameters to each device. If this selection is omitted, all parameters will be transmitted.

PASSWORD:	The MD5-coded hash value of the password for the required access level (e.g., the user or installation password of the Sunny WebBox).
DEVICES:	An array containing the device objects for which parameters are to be returned, and an optional selection of certain channels.
DEVICE_KEY:	The corresponding device key. See section 5.1 ”Device Object” (page 15).
CHANNELS:	An array containing the meta names of the required process data. The available meta names can be identified with the command RPC_GET_PROCESS_DATA_CHANNELS.

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": "Mode",
  "options":
  [
    "Stop",
    "Konstantspg.",
    "Mpp-Betrieb",
    "Res1",
    "Res2",
    "Res3",
    "Res4",
    "Res5"
  ],
  "value": "Mpp Operation",
  "name": "Mode",
  "unit": ""
},
{
  "min": "2 150",
  "max": "2 150",
  "meta": "Plimit",
  "value": "2 150",
  "name": "Plimit",
  "unit": "W"
},
{
  "min": "0",
  "max": "4294900000",
  "meta": "SMA-SN",
  "value": "2 000101000",
  "name": "SMA-SN",
  "unit": ""
},
{
  "min": "125",
  "max": "600",
  "meta": "Vpv-Start",
  "value": "150",
  "name": "Vpv-Start",
  "unit": "V"
},
{
  "min": "1",
  "max": "300",
  "meta": "T-Stop",
  "value": "4",
  "name": "T-Stop",
  "unit": "s"
},
{
  "min": "125",
  "max": "600",
  "meta": "Vconst-Setpoint",
  "value": "600",
  "name": "Vconst-Setpoint",
  "unit": "V"
},
{
  "min": "5",
  "max": "300",
  "meta": "T-Start",
  "value": "10",
  "name": "T-Start",
  "unit": "s"
},
{
  "min": "0",
  "max": "100",
  "meta": "Firmware-SRR",
  "value": "2",
  "name": "Firmware-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": "Firmware-BFR",
  "value": "2",
  "name": "Firmware-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 returns the device list submitted in the request with the apropriate current parameter values. The calling application verifies whether each parameter value has been set correctly.

Assembly:

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

Submit as parameters a list with the device objects whose parameter values are to be changed. Each device object contains a list of the parameters to be set.

Configuration of the parameters is performed synchronously. Response times depend on the number of parameters to be set. For the next example, the response time is approx. 10 seconds.

PASSWORD:	The MD5-coded hash value of the password for the required access level (e.g., the user or installation password of the Sunny WebBox).
DEVICES:	An array containing objects with the device keys of the devices, and an array with the CHANNELS whose parameter values are to be set.
DEVICE_KEY:	The corresponding device key. See section 5.1 "Device Object" (page 15).
CHANNELS:	An array containing the channel objects to be set for the respective device. A list with the available parameter channels can be obtained by means of the command RPC_GET_PARAMETER_CHANNELS.

Sample request:

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

Sample response:

```
{
  "version": "1.0",
  "id": "1",
  "format": "JSON",
  "proc": "SetParameter",
  "result":
  {
    "devices":
    [
      {
        "key": "WR21TL06:2000101000",
        "channels":
        [
          {
            "min": "0",
            "max": "7",
            "meta": "Mode",
            "options":
```

```
          [
            "Stop",
            "Konstantspg.",
            "Mpp-Betrieb",
            "Res1",
            "Res2",
            "Res3",
            "Res4",
            "Res5"
          ],
          "value": "Mpp Operation",
          "name": "Mode",
          "unit": ""
        }
      ]
    },
    {
      "key": "WR21TL06:2000101001",
      "channels":
      [
        {
          "min": "0",
          "max": "7",
          "meta": "Mode",
          "options":
          [
            "Stop",
            "Konstantspg.",
            "Mpp-Betrieb",
            "Res1",
            "Res2",
            "Res3",
            "Res4",
            "Res5"
          ],
            "value": "Stop",
            "name": "Mode",
            "unit": ""
          }
        ]
      }
    ]
  }
}
```

7 Sources

[1]
RFC 4 627: The application/json Media Type for Javascript Object Notation (JSON)
http://www.ietf.org/rfc/rfc4627.txt?number=4 627

[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/rfc1321.txt?number=1 321

8 Contact

If you have technical problems concerning our products, contact the SMA Serviceline. We need the following information in order to provide you with the necessary assistance:

- Inverter Serial Number and Type
- Serial Number and firmware version of the Sunny WebBox

SMA Solar Technology America, LLC

6020 West Oaks Blvd, Ste 300

Rocklin, CA 95765

Tel. +1 916 625 0870

Tel. +1 877-MY SMA TECH

Tel. +1 877 697 6283 (Toll free, available for USA, Canada and Puerto Rico)

Fax +1 916 625-0871

Service@SMA-America.com

www.SMA-America.com

SMA Solar Technology Canada Inc.

2425 Matheson Blvd E , 8th Floor

Mississauga, ON L4W 5K5 , Canada

Tel. +1 877 506 1756 (Toll free, available for Canada)

Service@SMA-Canada.ca

www.SMA-Canada.ca