Plant Monitoring

SUNNY WEBBOX RPC

User Manual

Remote Procedure Call Description

Interfaces and API Definition

SWebBoxRPC-BA-en-14 | Version 1.4

EN

Table of Contents

1 Introduction

1.1 Validity

This document applies to the RPC interface of the Sunny WebBox and the Sunny WebBox with *Bluetooth*® Wireless Technology.

1.2 Nomenclature

In this document, the Sunny WebBox with *Bluetooth*® Wireless Technology is referred to as Sunny WebBox with *Bluetooth*.

1.3 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

1.4 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/rfc1321.txt?number=1321

2 System Overview

2.1 Sunny WebBox RPC

The data loggers Sunny WebBox and Sunny WebBox with *Bluetooth* continuously record all the data of a PV plant. This is then averaged over a configurable interval and cached. The data can be transmitted at regular intervals to the Sunny Portal for analysis and visualization.

Via the Sunny WebBox and Sunny WebBox with *Bluetooth* RPC interface, selected data from the PV plant can be transmitted to a remote terminal by means of an RPC protocol (Remote Procedure Call protocol).

To do this, the Sunny WebBox provides a pool of service procedures (see Section 6 "Object Definitions", page 12) that 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 3 "JavaScript Object Notation", page 7).

2.2 Differences between Sunny WebBox and Sunny WebBox with *Bluetooth*

Different representation for Sunny WebBox and Sunny WebBox with *Bluetooth*

The presentation of the plant data transmitted via the RPC interface is different for the Sunny WebBox and the Sunny WebBox with *Bluetooth*.

This document only contains sample responses for the Sunny WebBox and none from the Sunny WebBox with *Bluetooth*.

You will find a list of measured values and parameters for the Sunny WebBox with *Bluetooth* in the technical description "Measured Values and Parameters" at www.SMA-Solar.com.

Available service procedures	**Sunny WebBox**	**Sunny WebBox with *Bluetooth***
RPC_GET_PLANT_OVERVIEW	+	+
RPC_GET_DEVICES	+	+
RPC_GET_PROCESS_DATA_CHANNELS	+	+
RPC_GET_PROCESS_DATA	+	+
RPC_GET_PARAMETER_CHANNELS	+	–
RPC_GET_PARAMETER	+	–
RPC_SET_PARAMETER	+	–

3 JavaScript Object Notation

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

3.1 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 array "device" has 2 fields, each having one device object (see Section 6.1 "Device Object", page 12), which in turn contains nested device objects.

```
RPC={
    "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
         }
        ]
      }
    ]
  }
```

4 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.

4.1 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 4.2 "Return Value (Response)", page 10).
- **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 Sunny 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.

4.2 Return Value (Response)

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

The following formats are currently available:

- JSON

4.2.1 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 serves to assign the request and 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.

4.3 Query Interval

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

5 Interfaces

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

5.1 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.

5.2 RPC via HTTP

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

The default setting is port 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 4.1.

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.

6 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 accordingly to all other data transmission formats.

6.1 Device Object

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

```
RPC={
    "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-definable name of the device (e.g. "INV left"). Defining this element is optional. If the element is used but no name was defined, "null" is entered.
channels:	An array of channel objects of the device. Defining this element is optional. If the element is used but no channels exist, "null" is entered.
children:	An array of objects containing sub-devices. Defining this element is optional. If the element is used but no sub-devices exist, "null" is entered.

6.2 Channel Object

Describes a process data or parameter channel of a device.

```
RPC={
        "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 which have no 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.

7 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 4 and it is not necessary to differentiate between all procedures here.

7.1 RPC_GET_PLANT_OVERVIEW

7.1.1 Version 1.0

Returns an object with the following plant data:

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

Structure:

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

Sample request:

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

Sample response:

```
RPC={
    "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 was transmitted:

Power = 4,250 W,
E-Today = 45.23 kWh,
E-Total = 7,821 kWh,
Status = MPP,
No error

7.2 RPC_GET_DEVICES

7.2.1 Version 1.0

Returns a hierarchical list of all detected plant devices.

Structure:

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

Sample request:

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

Sample response:

```
RPC={
        "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
                }
              ]
            }
          ]
        }
      }
```

7.3 RPC_GET_PROCESS_DATA_CHANNELS

7.3.1 Version 1.0

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

Structure:

```
RPC={
    "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:

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

Sample response:

```
RPC={
        "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"
          ]
        }
      }
```

7.4 RPC_GET_PROCESS_DATA

7.4.1 Version 1.0

Returns process data for up to 5 devices per request.

Structure:

```
RPC={
    "version": "1.0",
    "proc": "GetProcessData",
    "id": "ID",
    "format": "FORMAT",
    "params":
    {
      "DEVICES":
      [
        {
          "key": DEVICE_KEY,
          "channels": [CHANNELS]
        }
      ]
    }
  }
```

A list with the device keys whose process data is to be provided must be transferred as a parameter. 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 for which process data is to be returned, and optional CHANNELS.
DEVICE_KEY:	The corresponding device key (see Section 6.1 "Device Object", page 12).
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:

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

Sample response:

```
RPC={
        "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","key": "WR715-19:263415748",
        "channels":
        [
          {
            "meta": "Pac",
            "name": null,
            "value": "630",
            "unit": "W"
          }
        ]
      }
    ]
  }
}
```

7.5 RPC_GET_PARAMETER_CHANNELS

7.5.1 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.

Structure:

```
RPC={
        "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 desired access level.
DEVICE_KEY:	The device key of a device for whose type the parameter channels are to be returned.

Sample request

```
RPC={
        "version": "1.0",
        "proc": "GetParameterChannels",
        "id": "1",
        "format": "JSON",
        "passwd": "a289fa4252ed5af8e3e9f9bee545c172",
        "params":
        {
          "device": "WR715-19:263415747"
        }
      }
```

Sample response:

```
RPC={
        "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"
          ]
        }
      }
```

7.6 RPC_GET_PARAMETER

7.6.1 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.

Structure:

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

A list with the device objects whose parameters are to be provided must be transferred as a parameter. 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 desired access level.
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 6.1 "Device Object", page 12).
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

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

Sample response:

```
RPC={
        "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"
                }
              ]
            }
          ]
         }
        }
```

7.7 RPC_SET_PARAMETER

7.7.1 Version 1.0

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

Structure:

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

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

Configuration of the parameters is performed synchronously. Thus, the response time depends 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 desired access level.
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 6.1 "Device Object", page 12).
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:

```
RPC={
        "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:

```
RPC={
     "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": ""
             }
           ]
         }
       ]
     }
   }
```

Legal Provisions

Declaration of Conformity

SMA Solar Technology AG hereby declares that this equipment is in compliance with the essential requirements and other relevant provisions of Directive 1999/5/EC. You can find the entire CE declaration of conformity at www.SMA-Solar.com.

Trademarks

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

www.SMA-Solar.com

SMA Solar Technology AG www.SMA.de	**SMA Solar India Pvt. Ltd.** www.SMA-India.com
SMA Australia Pty. Ltd. www.SMA-Australia.com.au	**SMA Italia S.r.l.** www.SMA-Italia.com
SMA Benelux bvba/sprl www.SMA-Benelux.com	**SMA Japan K.K.** www.SMA-Japan.com
SMA Beijing Commercial Company Ltd. www.SMA-China.com.cn	**SMA Technology Korea Co., Ltd.** www.SMA-Korea.com
SMA Central & Eastern Europe s.r.o. www.SMA-Czech.com	**SMA Middle East LLC** www.SMA-Me.com
SMA France S.A.S. www.SMA-France.com	**SMA Portugal - Niestetal Services Unipessoal Lda** www.SMA-Portugal.com
SMA Hellas AE www.SMA-Hellas.com	**SMA Solar (Thailand) Co., Ltd.** www.SMA-Thailand.com
SMA Ibérica Tecnología Solar, S.L.U. www.SMA-Iberica.com	**SMA Solar UK Ltd.** www.SMA-UK.com