SHIFTING THE LIMITS

Fronius Solar API V1

EN

Operating Instructions

System monitoring

42,0410,2012,EN 013-02062020

Contents

1 Introduction

The Fronius Solar API is a means for third parties to obtain data from various Fronius devices (inverters, Sensor-Cards, StringControls) in a defined format through a central facility which acts as a proxy (e.g. Fronius Datalogger Web or Fronius Solar.web).

Currently, the only way to interact with this API is by making a HTTP request to a specific CGI. The URLs for the particular requests and the devices supporting them are listed at the beginning of each request description.
The API is versioned, meaning that multiple versions of this API may be available on the same device. The URLs in this document always point to the version of the API which this document describes. The highest supported version on the device can be queried. See 2.3.1 for details.
In order to check your product for compatibility with this version of the API specification, please see the separate document provided for this purpose.

The API divides roughly into realtime and archive requests: Realtime requests will obtain the data directly from the devices and can therefore only be used when the devices are not in standby or unavailable in any other matter. Archive requests will use the data stored in a central logging facility to obtain the results and are of course not subjected to the former limitation.

1.1 Fronius Generation 24

☞ Products from Fronius Generation 24 are described here but be aware that this interface is a temporary solution and will be removed.

2 General Considerations

2.1 Output Formats

Currently, the only output format supported is JSON, a lightweight data interchange format. It is easy to read and write for both humans and machines and it offers some advantages over XML, like basic typing and a leaner structure.

☞ It is strongly recommended to use appropriate frameworks or tools to parse json objects properly

2.2 Data Types

2.2.1 Numeric Types

JSON only knows one kind of numeric types, which can represent both floating point and integer values. It is however possible to specify a type in JSON description, but it is always in the hands of the interpreting system into which datatype a numeric node is converted.
Which range a certain numeric node actually can have is often determined by the device providing the value, and may also vary depending on the type of device (e.g. ”UAC” can be an integer value on older inverters, but a floating point value on newer ones).

This means we cannot reliably specify value ranges for all requests. So it is the responsibility of the API user to determine whether a value fits into a certain datatype in his language of choice.
What we can do is to specify whether a certain value is a floating point value (marked as ”number”) or an integer value (marked as ”integer”), where ”integer” must not be interpreted as the datatype ”int” like available in C/C++, it just means it is a value without decimal places.
For these specifications, please refer to the sections discussing the respective request.

Examples

number 1, -2, 0, 4, 4.0, 0.001, -10.002, ...

integer 1, -2, 0, 4, -10

unsigned integer 1, 0, 4, 10

unsigned number 1, 0, 4, 10, 0.001, 14.1234

2.2.2 Date/Time

Information on date/time is always (and can only be) represented by a string. The format for these strings inside this API has been defined as follows.

- Strings which include information on both date and time are always in RFC3339 format with time zone offset or Zulu marker.
  See Section 5.6 of RFC3339
  Example 1: `2011-10-20T10:23:17+02:00` (UTC+2)
  Example 2: `2011-10-20T08:23:17Z` (UTC)

- Strings which only include information on the date are of the format `yyyy-MM-dd`.
- Strings which only include information on the time are of the format `hh:mm:ss`.
- If no information on the time zone is given, any date/time specification is considered to be in local time of the PV system.

2.3 Requests

Currently, the only request protocol supported is HTTP.

☞ Use HTTP-GET requests to query data from Solar API

2.3.1 Querying of API version

The highest supported version on the device can be queried using the URL `/solar_api/GetAPIVersion.cgi`.

Listing 1: Object structure of GetAPIVersion response

```
object {

    # Numeric version of the API.
    # all Datamanager and Hybridmanager support only APIVersion 1
    unsigned integer APIVersion;

    # URL under which the CGIs for the requests can be reached.
    string BaseURL;

    # Compatibility version of current implementation (except GetPowerFlowRealtimeData)
    # THIS FIELD IS AVAILABLE AND MANDATORY SINCE
    #   Datamanager 3.9.1-x
    #   Hybridmanager 1.7.1-x
    # FORMAT: MAJOR.MINOR-BUILD
    #   Major: compatibility range (something big changed)
    #   Minor: feature range (new features added)
    #   Build: bugfix revision (only bugfixes applied)
    string CompatibilityRange;
}
```

Listing 2: Example: Complete response for GetAPIVersion request

```
{
    "APIVersion" : 1,
    "BaseURL" : "/solar_api/v1/",
    "CompatibilityRange" : "1.5-9"
}
```

2.3.2 Addressing of devices

A specific device is identified by the string parameter *DeviceId*.
For Fronius Solar Net devices this string shall contain the numeric address of the targeted device.
Future generations of Fronius devices may also use non numerical addresses, so this API is designed to allow for both.

2.4 Responses

The response will always be a valid JSON string ready to be evaluated by standard libraries.
If the response is delivered through HTTP, the `Content-Type` Header shall be either `text/javascript` or `application/json`.

All JSON structures are described using *Orderly JSON*, a textual format for describing JSON data. Please refer to the online documentation on *Orderly* for details.
Note that the definitions of some response bodies are not totally accurate, because there's no (known) way to express nodes named after values/channels (e.g. objects which are named "PAC" or "Power"). But each description is accompanied by an example which should clear up any uncertainty.

The contents of the response object will vary depending on the preceding request but it always contains a common response header and a request body.

Listing 3: Object structure of valid response

```
object {

  object Head: {}*;

  object Body: {}*;

}
```

Listing 4: Example: Complete response for GetInverterRealtimeData request on non hybrid system

```
{
   "Body" : {
      "Data" : {
         "DAY_ENERGY" : {
            "Unit" : "Wh",
            "Value" : 16390
         },
         "DeviceStatus" : {
            "ErrorCode" : 0,
            "LEDColor" : 2,
            "LEDState" : 0,
            "MgmtTimerRemainingTime" : -1,
            "StateToReset" : false,
            "StatusCode" : 7
         },
         "FAC" : {
            "Unit" : "Hz",
            "Value" : 49.990000000000002
         },
         "IAC" : {
            "Unit" : "A",
            "Value" : 17.890000000000001
         },
         "IDC" : {
            "Unit" : "A",
            "Value" : 6.7400000000000002
         },
         "PAC" : {
            "Unit" : "W",
            "Value" : 4097
         },
         "TOTAL_ENERGY" : {
            "Unit" : "Wh",
            "Value" : 8612942
         },
         "UAC" : {
            "Unit" : "V",
            "Value" : 229.90000000000001
         },
         "UDC" : {
```

```
            "Unit" : "V",
            "Value" : 674
         },
         "YEAR_ENERGY" : {
            "Unit" : "Wh",
            "Value" : 775271
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DataCollection" : "CommonInverterData",
         "DeviceClass" : "Inverter",
         "DeviceId" : "1",
         "Scope" : "Device"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2018-03-01T12:49:42+01:00"
   }
}
```


2.4.1 Availability

A request is listed as ”Available” if the response http code differs to 404 (not found). It does not relay to technical compatibility nor functionality.

2.4.2 Common Response Header

The common response header (CRH) is present in every response. It indicates, among other things, whether the request has been successful and the body of the response is valid.

Listing 5: Object Structure of Common Response Header

```
object {

  # Repetition of the parameters which produced this response.
  object {
    # Filled with properties named like the given parameters.
  }* RequestArguments;

  # Information about the response.
  object {

    # Indicates if the request went OK or gives a hint about what went wrong.
    # 0 means OK, any other value means something went wrong (e.g. Device not available,
    # invalid params, no data in logflash for given time, ...).
    integer Code;

    # Error message, may be empty.
    string Reason;

    # Error message to be displayed to the user, may be empty.
    string UserMessage;

  } Status;

  # RFC3339 timestamp in localtime of the datalogger.
  # This is the time the request was answered - NOT the time when the data
  # was queried from the device.
  string Timestamp;

};
```

Value	Status	Description
0	OKAY	Request successfully finished, Data are valid
1	NotImplemented	The request or a part of the request is not implemented yet
2	Uninitialized	Instance of APIRequest created, but not yet configured
3	Initialized	Request is configured and ready to be sent
4	Running	Request is currently being processed (waiting for response)
5	Timeout	Response was not received within desired time
6	Argument Error	Invalid arguments/combination of arguments or missing arguments
7	LNRequestError	Something went wrong during sending/receiving of LN-message
8	LNRequestTimeout	LN-request timed out
9	LNParseError	Something went wrong during parsing of successfully received LN-message
10	ConfigIOError	Something went wrong while reading settings from local config
11	NotSupported	The operation/feature or whatever is not supported
12	DeviceNotAvailable	The device is not available
255	UnknownError	undefined runtime error

Table 1: Error Code Table

2.4.3 Request Body

The request body contains the actual data produced by the request and is therefore different for each request. The object structures of the various response bodies will be detailed later in the description of the respective API request.

3 Realtime Requests

These requests will be provided where direct access to the realtime data of the devices is possible. This is currently the case for the Fronius Datalogger Web and the Fronius Datamanager.

In order to eliminate the need to specify each wanted value separately when making a request or querying each value separately, so called ”Data Collections” were defined.
The values in these collections are gathered from one or more Fronius Solar Net messages and supplied to the user in a single response to a certain request.
It may be the case that more values are queried from the device than the user is interested in, but the overhead caused by these superfluous values should be negligible compared to the advantages this strategy provides for the user.

If a device cannot provide some values of a DataCollection (e.g. because they are not implemented on the device) then those values are omitted from the response.

3.1 *GetInverterRealtimeData* request

This request does not care about the configured visibility of single inverters. All inverters are reported.

3.1.1 Availability

Platform	Since version
Fronius Hybrid	Not all DataCollections supported
Fronius Non Hybrid	ALWAYS
Fronius Generation 24	ALWAYS

3.1.2 Collection availability

DataCollection	supported on	
	Fronius Hybrid Systems	Fronius Generation 24
CumulationInverterData	Yes	Yes
CommonInverterData	Yes	Yes
3PInverterData	Yes	Yes
MinMaxInverterData	NO	NO

3.1.3 URL for HTTP requests

`/solar_api/v1/GetInverterRealtimeData.cgi`

3.1.4 Parameters

Parameter	Type	Range/Values/Pattern	Description
Scope	String	”Device” ”System”	Query specific device(s) or whole system (uses collection ”CumulationInverterData”)
DeviceId	String	*Solar Net:* 0 …99	*Only needed for Scope ”Device”* Which inverter to query.
DataCollection	String	”CumulationInverterData” ”CommonInverterData” ”3PInverterData” ”MinMaxInverterData”	*Only needed for Scope ”Device”* Selects the collection of data that should be queried from the device. See 3.1.5 for details.

3.1.5 Data Collections

CumulationInverterData Values which are cumulated to generate a system overview.

Value name	specific data type	Description
PAC	integer	AC power (negative value for consuming power)
DAY_ENERGY	unsigned number	AC Energy generated on current day
YEAR_ENERGY	unsigned number	AC Energy generated in current year
TOTAL_ENERGY	unsigned number	AC Energy generated overall
DeviceStatus	object	Status information about inverter

CommonInverterData Values which are provided by all types of Fronius inverters.

Value name	specific data type	Description
PAC	integer	AC power (negative value for consuming power)
SAC	unsigned integer	AC power (absolute) *Currently not implemented because not handled correctly by all inverters.*
IAC	unsigned number	AC current (absolute, accumulated over all lines)
UAC	unsigned number	AC voltage
FAC	unsigned number	AC frequency
IDC	unsigned number	DC current
UDC	unsigned number	DC voltage
DAY_ENERGY	unsigned number	AC Energy generated on current day
YEAR_ENERGY	unsigned number	AC Energy generated in current year
TOTAL_ENERGY	unsigned number	AC Energy generated overall
DeviceStatus	object	Status information about inverter

3PInverterData Values which are provided by 3phase Fronius inverters.

Value name	specific data type	Description
IAC_L1	unsigned number	AC current Phase 1 (absolute)
IAC_L2	unsigned number	AC current Phase 2 (absolute)
IAC_L3	unsigned number	AC current Phase 3 (absolute)
UAC_L1	unsigned number	AC voltage Phase 1
UAC_L2	unsigned number	AC voltage Phase 2
UAC_L3	unsigned number	AC voltage Phase 3
T_AMBIENT	integer	Ambient temperature
ROTATION_SPEED_FAN_FL	unsigned integer	Rotation speed of front left fan
ROTATION_SPEED_FAN_FR	unsigned integer	Rotation speed of front right fan
ROTATION_SPEED_FAN_BL	unsigned integer	Rotation speed of back left fan
ROTATION_SPEED_FAN_BR	unsigned integer	Rotation speed of back right fan

MinMaxInverterData Minimum- and Maximum-values of various inverter values.

Value name	specific data type	Description
DAY_PMAX	unsigned integer	Maximum AC power of current day
DAY_UACMAX	number	Maximum AC voltage of current day
DAY_UACMIN	number	Minimum AC voltage of current day
DAY_UDCMAX	number	Maximum DC voltage of current day
YEAR_PMAX	unsigned integer	Maximum AC power of current year
YEAR_UACMAX	number	Maximum AC voltage of current year
YEAR_UACMIN	number	Minimum AC voltage of current year
YEAR_UDCMAX	number	Maximum DC voltage of current year
TOTAL_PMAX	unsigned integer	Maximum AC power of current year
TOTAL_UACMAX	number	Maximum AC voltage overall
TOTAL_UACMIN	number	Minimum AC voltage overall
TOTAL_UDCMAX	number	Maximum DC voltage overall

3.1.6 Object structure of request body (Scope ”Device”)

Listing 6: Object structure of request body for GetInverterRealtimeData request (Scope ”Device”)

```
object {

# Collection of named value-unit pairs according to selected DataCollection.
# Members of Data object are named according to the value they represent (e.g. "PAC").
  object {

    # Value-Unit pair.
    object {

      # Unscaled value.
      # value name based specific data type
      <specific data type> Value;

      # Base unit of the value, never contains any prefixes.
      string Unit;

    } __VALUE_NAME__;

  }* Data;
};
```

3.1.7 Example of request body (Scope ”Device”)

Listing 7: Reply body for GetInverterRealtimeData scope=”Device” and collection=”CommonInverterData”

```
{
   "Body" : {
      "Data" : {
         "DAY_ENERGY" : {
            "Unit" : "Wh",
            "Value" : 1393.2
         },
         "DeviceStatus" : {
            "ErrorCode" : 0,
            "LEDColor" : 2,
            "LEDState" : 0,
            "MgmtTimerRemainingTime" : -1,
            "StateToReset" : false,
            "StatusCode" : 7
         },
         "FAC" : {
            "Unit" : "Hz",
            "Value" : 49.969999999999999
         },
```

```
            "IAC" : {
               "Unit" : "A",
               "Value" : 0.35999999999999999
            },
            "IDC" : {
               "Unit" : "A",
               "Value" : 0.32000000000000001
            },
            "PAC" : {
               "Unit" : "W",
               "Value" : 84
            },
            "TOTAL_ENERGY" : {
               "Unit" : "Wh",
               "Value" : 1734796.1200000001
            },
            "UAC" : {
               "Unit" : "V",
               "Value" : 232.40000000000001
            },
            "UDC" : {
               "Unit" : "V",
               "Value" : 399.89999999999998
            },
            "YEAR_ENERGY" : {
               "Unit" : "Wh",
               "Value" : 322593.5
            }
         }
      },
      "Head" : {
         "RequestArguments" : {
            "DataCollection" : "CommonInverterData",
            "DeviceClass" : "Inverter",
            "DeviceId" : "1",
            "Scope" : "Device"
         },
         "Status" : {
            "Code" : 0,
            "Reason" : "",
            "UserMessage" : ""
         },
         "Timestamp" : "2019-06-12T15:31:03+02:00"
      }
}
```


Listing 8: Reply body for GetInverterRealtimeData scope=”Device” and collection=”3PInverterData” on Generation 24 Primo


```
{
   "Body" : {
      "Data" : {
         "IAC_L1" : {
            "Unit" : "A",
            "Value" : 0.55099999904632568
         },
         "T_AMBIENT" : {
            "Unit" : "C",
            "Value" : 51.384124755859375
         },
         "UAC_L1" : {
            "Unit" : "V",
            "Value" : 235.60000610351562
         }
      }
   },
   "Head" : {
```

```
      "RequestArguments" : {
         "DataCollection" : "3PInverterData",
         "DeviceClass" : "Inverter",
         "DeviceId" : "1",
         "Scope" : "Device"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-08-28T06:14:59+00:00"
   }
}
```


Listing 9: Reply body for GetInverterRealtimeData scope=”Device” and collection=”3PInverterData” on Generation 24 Symo


```
{
   "Body" : {
      "Data" : {
         "IAC_L1" : {
            "Unit" : "A",
            "Value" : 0.39000001549720764
         },
         "IAC_L2" : {
            "Unit" : "A",
            "Value" : 0.38900002837181091
         },
         "IAC_L3" : {
            "Unit" : "A",
            "Value" : 0.3970000147819519
         },
         "T_AMBIENT" : {
            "Unit" : "C",
            "Value" : 47.81353759765625
         },
         "UAC_L1" : {
            "Unit" : "V",
            "Value" : 233.10000610351562
         },
         "UAC_L2" : {
            "Unit" : "V",
            "Value" : 234.5
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DataCollection" : "3PInverterData",
         "DeviceClass" : "Inverter",
         "DeviceId" : "1",
         "Scope" : "Device"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-08-28T05:51:34+00:00"
   }
}
```


Listing 10: Reply body for GetInverterRealtimeData scope=”Device” and collection=”CommonInverterData” on Generation 24 Symo


```
{
```

```
  "Body" : {
     "Data" : {
        "DeviceStatus:" : {
           "InverterState" : "Running"
        },
        "FAC" : {
           "Unit" : "Hz",
           "Value" : 49.974002838134766
        },
        "IAC" : {
           "Unit" : "A",
           "Value" : 1.5640000402927399
        },
        "IAC_L1" : {
           "Unit" : "A",
           "Value" : 0.3880000114440918
        },
        "IAC_L2" : {
           "Unit" : "A",
           "Value" : 0.38600000739097595
        },
        "IAC_L3" : {
           "Unit" : "A",
           "Value" : 0.39500001072883606
        },
        "IDC" : {
           "Unit" : "A",
           "Value" : 0.66413545608520508
        },
        "PAC" : {
           "Unit" : "W",
           "Value" : 270.0
        },
        "SAC" : {
           "Unit" : "VA",
           "Value" : 274.0
        },
        "UAC" : {
           "Unit" : "V",
           "Value" : 234.60000101725259
        },
        "UAC_L1" : {
           "Unit" : "V",
           "Value" : 233.19999694824219
        },
        "UAC_L2" : {
           "Unit" : "V",
           "Value" : 234.5
        },
        "UDC" : {
           "Unit" : "V",
           "Value" : 449.9998779296875
        },
        "UDC_3" : {
           "Unit" : "V",
           "Value" : 409.69140625
        }
     }
  },
  "Head" : {
     "RequestArguments" : {
        "DataCollection" : "CommonInverterData",
        "DeviceClass" : "Inverter",
        "DeviceId" : "1",
        "Scope" : "Device"
     },
     "Status" : {
```

```
            "Code" : 0,
            "Reason" : "",
            "UserMessage" : ""
         },
         "Timestamp" : "2019-08-28T05:52:49+00:00"
      }
}
```

Listing 11: Reply body for GetInverterRealtimeData scope=”Device” and collection=”CumulationInverterData” on Generation 24 Primo

```
{
   "Body" : {
      "Data" : {
         "DeviceStatus:" : {
            "InverterState" : "Running"
         },
         "PAC" : {
            "Unit" : "W",
            "Value" : 8.4296154682294417e+252
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DataCollection" : "CumulationInverterData",
         "DeviceClass" : "Inverter",
         "DeviceId" : "1",
         "Scope" : "Device"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-08-28T05:59:13+00:00"
   }
}
```

3.1.8 Object structure of request body (Scope ”System”)

Listing 12: Object structure of request body for GetInverterRealtimeData request (Scope ”System”)

```
object {

# Collection of named object(s) containing values per device and metadata.
# Members of Data object are named according to the value they represent (e.g. "PAC").
  object {

    # Value-Unit pair.
    object {

      # Base unit of the value, never contains any prefixes.
      string Unit;

      # Unscaled values per device.
      # Property name is the DeviceId to which the value belongs.
      object {

        <specific data type> 1;  # value from device with index 1.
        <specific data type> 2;  # value from device with index 2.
        # .. and so on.

      }* Values;

    } __VALUE_NAME__;
```

```
  }* Data;

};
```

3.1.9 Example of request body (Scope ”System”)

Listing 13: Example of request body for GetInverterRealtimeData request (Scope ”System”)


```
{
   "Body" : {
      "Data" : {
         "DAY_ENERGY" : {
            "Unit" : "Wh",
            "Values" : {
               "1" : 1393,
               "2" : 1618,
               "3" : 1695,
               "55" : 1698
            }
         },
         "PAC" : {
            "Unit" : "W",
            "Values" : {
               "1" : 84,
               "2" : 109,
               "3" : 109,
               "55" : 108
            }
         },
         "TOTAL_ENERGY" : {
            "Unit" : "Wh",
            "Values" : {
               "1" : 1734796,
               "2" : 3026782,
               "3" : 3160499,
               "55" : 3275219
            }
         },
         "YEAR_ENERGY" : {
            "Unit" : "Wh",
            "Values" : {
               "1" : 322593,
               "2" : 385172,
               "3" : 399904,
               "55" : 403993
            }
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "Inverter",
         "Scope" : "System"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-12T15:31:04+02:00"
   }
}
```


Listing 14: Example of request body for GetInverterRealtimeData request (Scope ”System”) on Generation 24

```
{
   "Body" : {
      "Data" : {
         "PAC" : {
            "Unit" : "W",
            "Value" : {
               "1" : 271.0
            }
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "Inverter",
         "Scope" : "System"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-08-28T05:28:12+00:00"
   }
}
```


3.2 *GetSensorRealtimeData* request

This request provides data for all channels of a single Fronius Sensor Card.
Inactive channels and channels with damaged sensors are not included in the response.

3.2.1 Availability

Platform	Since version
Fronius Hybrid	ALWAYS
Fronius Non Hybrid	ALWAYS
Fronius Generation 24	ALWAYS

☞ This API is useless on Fronius Hybrid systems which are unable to get connected to sensor cards anyway.

☞ API is available but always return an error on Generation 24

3.2.2 URL for HTTP requests

`/solar_api/v1/GetSensorRealtimeData.cgi`

3.2.3 Parameters

Parameter	Type	Range/Values/Pattern	Description
Scope	String	”Device” ”System”	Query specific device(s) or whole system
DeviceId	String	*Solar Net:* 0 …9	Which card to query.
DataCollection	String	”NowSensorData” ”MinMaxSensorData”	Selects the collection of data that should be queried from the device. See 3.2.4 for details.

3.2.4 Data Collections

NowSensorData The presently measured values of every active channel.

MinMaxSensorData The minimum and maximum values for every time period (day, month, year, total) of every channel.
Some channels do not have a minimum value because it would always be zero. For these channels, the minimum value is not included.

3.2.5 Object structure of request body (DataCollection ”NowSensorData”)

Listing 15: Object structure of request body for GetSensorRealtimeData request (DataCollection ”NowSensorData”)

```
object {

  # Collection of named object(s) containing values per channel and metadata.
  # Members of Data object are named according to the channel index they represent (e.g.
      "0").
  object {

    # Value-Unit pair.
    object {

      # Value for the channel.
      number Value;

      # Base unit of the value, never contains any prefixes.
      string Unit;

    } __CHANNEL_INDEX__;

  }* Data;

};
```

3.2.6 Example of request body (DataCollection ”NowSensorData”)

Listing 16: Example of request body for GetSensorRealtimeData request (DataCollection ”NowSensorData”)

```
{
   "Body" : {
      "Data" : {
         "0" : {
            "Unit" : "°C",
            "Value" : -9
         },
         "1" : {
            "Unit" : "°C",
            "Value" : 24
         },
         "2" : {
            "Unit" : "W/m2",
            "Value" : 589
         },
         "4" : {
            "Unit" : "KWh/m2",
            "Value" : 0
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DataCollection" : "NowSensorData",
         "DeviceClass" : "SensorCard",
         "DeviceId" : "1",
         "Scope" : "Device"
      },
      "Status" : {
```

```
        "Code" : 0,
        "Reason" : "",
        "UserMessage" : ""
      },
      "Timestamp" : "2018-03-01T13:25:34+01:00"
   }
}
```

3.2.7 Object structure of request body (DataCollection ”MinMaxSensorData”)

Listing 17: Object structure of request body for GetSensorRealtimeData request (DataCollection ”MinMaxSensorData”)

```
object {

    # Collection of named object(s) containing min/max values per channel and metadata.
    # Members of Data object are named according to the channel index they represent (e.g.
        "0").
    object {

        # Object representing one channel.
        object {

            # Whether this channel is currently active.
            boolean SensorActive;

            # Object representing min/max values of current day.
            object {

                # Maximum value with unit.
                object {
                    number Value;
                    string Unit;
                } Max;

                # Minimum value with unit.
                # This object is only present in temperature channels (channel# 0 and 1)
                # as other channels do not have minimum values.
                object {
                    number Value;
                    string Unit;
                } Min;
            } Day;

            # Object representing min/max values of current month.
            object {
                object {
                    number Value;
                    string Unit;
                } Max;
                object {
                    number Value;
                    string Unit;
                } Min;
            } Month;

            # Object representing min/max values of current year.
            object {
                object {
                    number Value;
                    string Unit;
                } Max;
                object {
                    number Value;
                    string Unit;
                } Min;
```

```
            } Year;

            # Object representing total min/max values.
            object {
                object {
                    number Value;
                    string Unit;
                } Max;
                object {
                    number Value;
                    string Unit;
                } Min;
            } Total;

        } __CHANNEL_INDEX__;

    }* Data;

};
```

3.2.8 Example of request body (DataCollection ”MinMaxSensorData”)

Listing 18: Example of request body for GetSensorRealtimeData request (DataCollection ”MinMaxSensorData”)

```
{
   "Body" : {
      "Data" : {
         "0" : {
            "Day" : {
               "Max" : {
                  "Unit" : "°C",
                  "Value" : 66
               },
               "Min" : {
                  "Unit" : "°C",
                  "Value" : 46
               }
            },
            "Month" : {
               "Max" : {
                  "Unit" : "°C",
                  "Value" : 85
               },
               "Min" : {
                  "Unit" : "°C",
                  "Value" : 0
               }
            },
            "SensorActive" : true,
            "Total" : {
               "Max" : {
                  "Unit" : "°C",
                  "Value" : 85
               },
               "Min" : {
                  "Unit" : "°C",
                  "Value" : -35
               }
            },
            "Year" : {
               "Max" : {
                  "Unit" : "°C",
                  "Value" : 85
               },
               "Min" : {
                  "Unit" : "°C",
```

```
                "Value" : 0
            }
        }
    },
    "1" : {
        "Day" : {
            "Max" : {
                "Unit" : "°C",
                "Value" : 27
            },
            "Min" : {
                "Unit" : "°C",
                "Value" : 27
            }
        },
        "Month" : {
            "Max" : {
                "Unit" : "°C",
                "Value" : 77
            },
            "Min" : {
                "Unit" : "°C",
                "Value" : 27
            }
        },
        "SensorActive" : true,
        "Total" : {
            "Max" : {
                "Unit" : "°C",
                "Value" : 187
            },
            "Min" : {
                "Unit" : "°C",
                "Value" : -35
            }
        },
        "Year" : {
            "Max" : {
                "Unit" : "°C",
                "Value" : 77
            },
            "Min" : {
                "Unit" : "°C",
                "Value" : 27
            }
        }
    },
    "2" : {
        "Day" : {
            "Max" : {
                "Unit" : "W/m2",
                "Value" : 0
            }
        },
        "Month" : {
            "Max" : {
                "Unit" : "W/m2",
                "Value" : 159
            }
        },
        "SensorActive" : true,
        "Total" : {
            "Max" : {
                "Unit" : "W/m2",
                "Value" : 10036
            }
        },
```

```
            "Year" : {
               "Max" : {
                  "Unit" : "W/m2",
                  "Value" : 159
               }
            }
         },
         "3" : {
            "Day" : {
               "Max" : {
                  "Unit" : "Hz",
                  "Value" : 0
               }
            },
            "Month" : {
               "Max" : {
                  "Unit" : "Hz",
                  "Value" : 0
               }
            },
            "SensorActive" : false,
            "Total" : {
               "Max" : {
                  "Unit" : "Hz",
                  "Value" : 2975
               }
            },
            "Year" : {
               "Max" : {
                  "Unit" : "Hz",
                  "Value" : 0
               }
            }
         },
         "4" : {
            "Day" : {
               "Max" : {
                  "Unit" : "Hz",
                  "Value" : 0
               }
            },
            "Month" : {
               "Max" : {
                  "Unit" : "Hz",
                  "Value" : 0
               }
            },
            "SensorActive" : false,
            "Total" : {
               "Max" : {
                  "Unit" : "Hz",
                  "Value" : 2982
               }
            },
            "Year" : {
               "Max" : {
                  "Unit" : "Hz",
                  "Value" : 0
               }
            }
         },
         "5" : {
            "Day" : {
               "Max" : {
                  "Unit" : "A",
                  "Value" : 0
               }
```

```
            },
            "Month" : {
               "Max" : {
                  "Unit" : "A",
                  "Value" : 0
               }
            },
            "SensorActive" : true,
            "Total" : {
               "Max" : {
                  "Unit" : "A",
                  "Value" : 36934
               }
            },
            "Year" : {
               "Max" : {
                  "Unit" : "A",
                  "Value" : 0
               }
            }
         }
      }
   },
  "Head": {
    "RequestArguments": {
      "DataCollection": "MinMaxSensorData",
      "DeviceClass": "SensorCard",
      "DeviceId" : "1"
      "Scope": "Device"
    },
    "Status": {
      "Code": 0,
      "Reason": "",
      "UserMessage": ""
    },
    "Timestamp": "2018-03-01T13:25:34+01:00"
  }
}
```


3.3 *GetStringRealtimeData* request

3.3.1 Availability

Platform	Since version
Fronius Hybrid	ALWAYS
Fronius Non Hybrid	ALWAYS
Fronius Generation 24	ALWAYS

☞ This API is useless on Fronius Hybrid systems which are unable to get connected to sensor cards anyway.

☞ String Control does not exist for Generation 24

3.3.2 URL for HTTP requests

`/solar_api/v1/GetStringRealtimeData.cgi`

3.3.3 Parameters

Parameter	Type	Range/Values/Pattern	Description
Scope	String	”Device” ”System”	Query specific device or whole system
DeviceId	String	*Solar Net:* 0 …199	Which device to query.
DataCollection	String	”NowStringControlData” ”LastErrorStringControlData” ”CurrentSumStringControlData”	Selects the collection of data that should be queried from the device. See 3.3.5 for details.
TimePeriod	String	”Day” ”Year” ”Total”	*Only needed for Collection ”CurrentSumStringControlData”* For which time period the current sums should be requested.

3.3.4 Collection availability

DataCollection	supported on		
	Non Hybrid	Hybrid	Generation 24
NowStringControlData	YES	useless	YES
LastErrorStringControlData	YES	useless	NO
CurrentSumStringControlData	YES	useless	NO

3.3.5 Data Collections

NowStringControlData The presently measured currents of every channels.

LastErrorStringControlData Information about the last error which triggered a service message.

CurrentSumStringControlData Current sums of all channels for a selected time period (day, year or total).

3.3.6 Object structure of request body (DataCollection ”NowStringControlData” and ”CurrentSumStringControlData”)

Listing 19: Object structure of request body for GetStringRealtimeData request (DataCollection ”NowStringControlData” and ”CurrentSumStringControlData”)

```
object {

  # Collection of named object(s) containing values per channel and metadata.
  # Members of Data object are named according to the channel index they represent (e.g.
      "0").
  object {

    # Value-Unit pair.
    object {

      # Value for the channel.
      number Value;

      # Base unit of the value, never contains any prefixes.
      string Unit;

    } __CHANNEL_INDEX__;

  }* Data;

};
```

3.3.7 Example of request body (DataCollection ”CurrentSumStringControlData”)

Listing 20: Example of request body for GetStringRealtimeData request (DataCollection ”CurrentSumStringControlData”)


```
{
   "Body" : {
      "Data" : {
         "1" : {
            "Unit" : "Ah",
            "Value" : 0
         },
         "2" : {
            "Unit" : "Ah",
            "Value" : 0
         },
         "3" : {
            "Unit" : "Ah",
            "Value" : 0
         },
         "4" : {
            "Unit" : "Ah",
            "Value" : 0
         },
         "5" : {
            "Unit" : "Ah",
            "Value" : 0
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DataCollection" : "CurrentSumStringControlData",
         "DeviceClass" : "StringControl",
         "DeviceId" : "8",
         "Scope" : "Device",
         "TimePeriod" : "Day"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-13T15:06:54+02:00"
   }
}
```


Listing 21: Reply body for GetStringRealtimeData DataCollection=”NowStringControlData” on Generation 24


```
{
   "Body" : {
      "Data" : {}
   },
   "Head" : {
      "RequestArguments" : {
         "DataCollection" : "NowStringControlData",
         "DeviceClass" : "StringControl",
         "Scope" : "System"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-08-28T07:20:58+00:00"
   }
}
```

3.3.8 Object structure of request body (DataCollection ”LastErrorStringControlData”)

Listing 22: Object structure of request body for GetStringRealtimeData request (DataCollection ”LastErrorString-ControlData”)

```
object {

    object {

        # Timestamp when the error was detected.
        string TimeOfError;

        # Average value of all channels
        # at the time the error was detected.
        object {

            number Value;

            # Base unit of the value, never contains any prefixes.
            string Unit;

        } StringAverage;

        # Contains information about every channel
        # at the time the error was detected.
        object {

            # Object representing one channel.
            object {

                # Deviation from string average.
                object {

                    number Value;

                    # Base unit of the value, never contains any prefixes.
                    string Unit;

                } Deviation;

                # Current sum
                object {

                    number Value;

                    # Base unit of the value, never contains any prefixes.
                    string Unit;

                } Sum;

            } __CHANNEL_INDEX__;

        }* Channels;

    } Data;
}
```

3.3.9 Example of request body (DataCollection ”LastErrorStringControlData”)

Listing 23: Example of request body for GetStringRealtimeData request (DataCollection ”LastErrorStringControl-Data”)

```
{
   "Body" : {
      "Data" : {
         "Channels" : {
```

```
            "1" : {
               "Deviation" : {
                  "Unit" : "%",
                  "Value" : 5.7000000000000002
               },
               "Sum" : {
                  "Unit" : "Ah",
                  "Value" : 0.84999999999999998
               }
            },
            "2" : {
               "Deviation" : {
                  "Unit" : "%",
                  "Value" : -12.6
               },
               "Sum" : {
                  "Unit" : "Ah",
                  "Value" : 0.69999999999999996
               }
            },
            "3" : {
               "Deviation" : {
                  "Unit" : "%",
                  "Value" : 7.0999999999999996
               },
               "Sum" : {
                  "Unit" : "Ah",
                  "Value" : 0.85999999999999999
               }
            },
            "4" : {
               "Deviation" : {
                  "Unit" : "%",
                  "Value" : 0
               },
               "Sum" : {
                  "Unit" : "Ah",
                  "Value" : 0
               }
            },
            "5" : {
               "Deviation" : {
                  "Unit" : "%",
                  "Value" : 0
               },
               "Sum" : {
                  "Unit" : "Ah",
                  "Value" : 0
               }
            }
         },
         "StringAverage" : {
            "Unit" : "Ah",
            "Value" : 0.81000000000000005
         },
         "TimeOfError" : "2010-10-23T09:32:00+02:00"
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DataCollection" : "LastErrorStringControlData",
         "DeviceClass" : "StringControl",
         "DeviceId" : "8",
         "Scope" : "Device"
      },
      "Status" : {
         "Code" : 0,
```

```
            "Reason" : "",
            "UserMessage" : ""
         },
         "Timestamp" : "2019-06-13T15:06:56+02:00"
      }
}
```

3.3.10 Object structure of request body (DataCollection ”NowStringControlData”)

Listing 24: Object structure of request body for GetStringRealtimeData request (DataCollection ”NowStringControlData”)

```
object {
    object {
        # Object representing one channel.
        object {

          number Value;

          # Base unit of the value, never contains any prefixes.
          string Unit;

        } __CHANNEL_INDEX__;
    } Data;
}
```

3.3.11 Example of request body (DataCollection ”NowStringControlData”)

Listing 25: Example of request body for GetStringRealtimeData request (DataCollection ”NowStringControlData”)

```
{
   "Body" : {
      "Data" : {
         "1" : {
            "Unit" : "A",
            "Value" : 0
         },
         "2" : {
            "Unit" : "A",
            "Value" : 0
         },
         "3" : {
            "Unit" : "A",
            "Value" : 0
         },
         "4" : {
            "Unit" : "A",
            "Value" : 0
         },
         "5" : {
            "Unit" : "A",
            "Value" : 0
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DataCollection" : "NowStringControlData",
         "DeviceClass" : "StringControl",
         "DeviceId" : "8",
         "Scope" : "Device"
      },
      "Status" : {
         "Code" : 0,
```

```
            "Reason" : "",
            "UserMessage" : ""
        },
        "Timestamp" : "2019-06-13T15:06:57+02:00"
    }
}
```

3.4 *GetLoggerInfo* request

This request provides information about the logging device which provides this API.

3.4.1 Availability

Platform	Since version
Fronius Hybrid	ALWAYS
Fronius Non Hybrid	ALWAYS
Fronius Generation 24	ALWAYS

☞ API is available but always return an error on Generation 24

3.4.2 URL for HTTP requests

`/solar_api/v1/GetLoggerInfo.cgi`

3.4.3 Object structure of request body

Listing 26: Object structure of request body for GetLoggerInfo request

```
object {

  object {

    # Unique ID of the logging device.
    string UniqueID;

    # String identifying the exact product type.
    # examples: "fronius-hybrid" or "fronius-datamanager-card"
    string ProductID;

    # String identifying the exact hardware platform.
    string PlatformID;

    # Hardware version of the logging device.
    string HWVersion;

    # Software version of the logging device. (Major.Minor.Revision-Build)
    string SWVersion;

    # Name of city/country which the user
    # selected as time zone.
    string TimezoneLocation/[a-zA-Z]+|/;

    # Name of the selected time zone.
    # May be empty if information not available.
    string TimezoneName/[a-zA-Z]+|/;

    # UTC offset in seconds east of UTC,
    # including adjustments for daylight saving.
    integer UTCOffset;

    # Default language set on the logging device
    # as a two letter abbreviation (e.g. "en").
    # NOTE: This attribute will be REMOVED soon
```

```
    string DefaultLanguage;

    # Grid supply tariff
    # This field is mandatory only for all Fronius Hybrid inverter
    # and Fronius Non Hybrid since 3.3.3-1
    number DeliveryFactor;

    # The cash factor set on the logging device,
    # NOT the factor set on the inverters.
    number CashFactor;

    # Currency of cash factor set on the logging device,
    # NOT the currency set on the inverters.
    string CashCurrency;

    # The CO2 factor set on the logging device,
    # NOT the factor set on the inverters.
    number CO2Factor;

    # Unit of CO2 factor set on the logging device,
    # NOT the unit set on the inverters.
    string CO2Unit;

  } LoggerInfo;

};
```

☞ Item ”DefaultLanguage” will be removed soon

3.4.4 Example of request body

Listing 27: Example of request body for GetLoggerInfo request


```
{
   "Body" : {
      "LoggerInfo" : {
         "CO2Factor" : 0.52999997138977051,
         "CO2Unit" : "kg",
         "CashCurrency" : "EUR",
         "CashFactor" : 0.11999999731779099,
         "DefaultLanguage" : "en",
         "DeliveryFactor" : 0.25,
         "HWVersion" : "2.4D",
         "PlatformID" : "wilma",
         "ProductID" : "fronius-datamanager-card",
         "SWVersion" : "3.14.1-2",
         "TimezoneLocation" : "Paris",
         "TimezoneName" : "CEST",
         "UTCOffset" : 7200,
         "UniqueID" : "240.107620"
      }
   },
   "Head" : {
      "RequestArguments" : {},
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-12T15:31:06+02:00"
   }
}
```

3.5 *GetLoggerLEDInfo* request

This request provides information about the LED states and colors on the device which provides this API.

3.5.1 Availability

Platform	Since version
Fronius Hybrid	ALWAYS
Fronius Non Hybrid	ALWAYS
Fronius Generation 24	ALWAYS

☞ API is available but always return an error on Generation 24

3.5.2 URL for HTTP requests

`/solar_api/v1/GetLoggerLEDInfo.cgi`

3.5.3 Object structure of request body

Listing 28: Object structure of request body for GetLoggerLEDInfo request

```
object {

  object {

    # State of one LED.
    object {

      # Color ("red", "green" or "none").
      string Color;

      # State ("on", "off", "blinking" or "alternating").
      string State;

    } __LED_NAME__ ;

  }* Data;

};
```

3.5.4 Example of request body

Listing 29: Example of request body for GetLoggerLEDInfo request

```
{
   "Body" : {
      "Data" : {
         "PowerLED" : {
            "Color" : "green",
            "State" : "on"
         },
         "SolarNetLED" : {
            "Color" : "green",
            "State" : "on"
         },
         "SolarWebLED" : {
            "Color" : "green",
            "State" : "on"
         },
         "WLANLED" : {
            "Color" : "red",
            "State" : "on"
         }
      }
```

```
   },
   "Head" : {
      "RequestArguments" : {},
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-12T15:31:07+02:00"
   }
}
```

3.6 *GetInverterInfo* request

This request provides information about all inverters that are currently being monitored by the logging device. So this means that inverters which are currently not online are also reported by this request, provided these inverters have been seen by the logging device within the last 24 hours.

If information about devices currently online is needed, the *GetActiveDeviceInfo* request should be used. This request also provides information about device classes other than inverters.

3.6.1 Availability

Platform	Since version
Fronius Hybrid	ALWAYS
Fronius Non Hybrid	ALWAYS
Fronius Generation 24	ALWAYS

3.6.2 URL for HTTP requests

`/solar_api/v1/GetInverterInfo.cgi`

3.6.3 Object structure of request body

Listing 30: Object structure of request body for GetInverterInfo request

```
object {

  # Collection of objects with infos about one inverter,
  # mapped by inverter index.
  object {

    # Info about a single inverter.
    # Name of object is the inverter index.
    object {

      # Device type of the inverter.
      integer DT;

      # PV power connected to this inverter (in watts).
      # If none defined, default power for this DT is used.
      integer PVPower;

      # Custom name of the inverter, assigned by the customer.
      string CustomName;

      # Whether the device shall be displayed in visualizations according
      # to customer settings. (0 do not show; 1 show)
      # visualization settings.
      unsigned integer Show;

      # Unique ID of the inverter (e.g. serial number).
      string UniqueID;
```

```
      # Error code that is currently present on inverter.
      # A value of -1 means that there is no valid error code.
      integer ErrorCode;

      # Status code reflecting the operational state of the inverter.
      integer StatusCode;

    } __INVERTER_INDEX__;

  }* Data;

};
```

3.6.4 Example of request body

Listing 31: Example of request body for GetInverterInfo request

```
{
   "Body" : {
      "Data" : {
         "1" : {
            "CustomName" : "
                &#80;&#114;&#105;&#109;&#111;&#32;&#56;&#46;&#50;&#45;&#49;&#32;&#40;&#49;&#41;
                ",
            "DT" : 102,
            "ErrorCode" : 0,
            "PVPower" : 500,
            "Show" : 1,
            "StatusCode" : 7,
            "UniqueID" : "38183"
         },
         "2" : {
            "CustomName" : "
                &#80;&#114;&#105;&#109;&#111;&#32;&#53;&#46;&#48;&#45;&#49;&#32;&#50;&#48;&#56;&#45;&#
                ",
            "DT" : 86,
            "ErrorCode" : 0,
            "PVPower" : 500,
            "Show" : 1,
            "StatusCode" : 7,
            "UniqueID" : "16777215"
         },
         "3" : {
            "CustomName" : "
                &#71;&#97;&#108;&#118;&#111;&#32;&#51;&#46;&#49;&#45;&#49;&#32;&#50;&#48;&#56;&#45;&#5
                ",
            "DT" : 106,
            "ErrorCode" : 0,
            "PVPower" : 500,
            "Show" : 1,
            "StatusCode" : 7,
            "UniqueID" : "7262"
         },
         "55" : {
            "CustomName" : "
                &#71;&#97;&#108;&#118;&#111;&#32;&#51;&#46;&#48;&#45;&#49;&#32;&#40;&#53;&#53;&#41;
                ",
            "DT" : 224,
            "ErrorCode" : 0,
            "PVPower" : 500,
            "Show" : 1,
            "StatusCode" : 7,
            "UniqueID" : "100372"
         }
      }
   },
```

```
   "Head" : {
      "RequestArguments" : {},
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-12T15:31:02+02:00"
   }
}
```


Listing 32: Reply body for GetInverterInfo on Generation 24


```
{
   "Body" : {
      "Data" : {
         "1" : {
            "CustomName" : "tr-3pn-01",
            "DT" : 1,
            "PVPower" : 0,
            "Show" : 1,
            "StatusCode" : "Running",
            "UniqueID" : "29301000987160033"
         }
      }
   },
   "Head" : {
      "RequestArguments" : {},
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-08-28T07:47:43+00:00"
   }
}
```


☞ The items 'DT' and 'PVPower' are invalid on Generation 24

☞ The item 'StatusCode' (specified as number) is implemented as string on Generation 24

3.6.5 Meaning of numerical status codes

The `StatusCode` Field is only reported as numerical value. The meaning of the numbers is shown in the table below.

Value	Description	provided by	
		Datamanager and Hybridmanager	Generation 24
0 - 6	Startup	YES	Yes
7	Running	YES	Yes
8	Standby	YES	Yes
9	Bootloading	YES	No
10	Error	YES	Yes
	Idle	No	Yes
	Ready	No	Yes
	Sleeping	No	Yes
	Unknown	No	Yes
	INVALID	No	Yes

3.7 *GetActiveDeviceInfo* request

This request provides information about which devices are currently online.

3.7.1 Availability

Platform	Since version
Fronius Hybrid	ALWAYS
Fronius Non Hybrid	ALWAYS
Fronius Generation 24	ALWAYS

3.7.2 URL for HTTP requests

`/solar_api/v1/GetActiveDeviceInfo.cgi`

3.7.3 Parameters

Parameter	Type	Range/Values/Pattern	Description
DeviceClass	String	”Inverter” ”Storage” ”Ohmpilot”[2] ”SensorCard”[3] ”StringControl”[3] ”Meter”[1] ”System”[1]	Which kind of device class to search for active devices. Uses different response format

1 2 3

3.7.4 DeviceClass is not System

Listing 33: Object structure of request body for GetActiveDeviceInfo request

```
object {

  # Collection of objects with infos about one inverter,
  # mapped by inverter index.
  object {

    # Info about a single device.
    # Name of object is the device index.
    object {

      # Mandatory Device type of the device.
      #  (only for Inverter, SensorCard or StringControl; others have -1)
      integer DT;

      # Optional attribute: serialnumber
      # usually supported by new Inverters, OhmPilots, Batteries and Smart Meters
      string  Serial;

      # Channel listing for Sensor Cards
      array {
        string;
      } ChannelNames;

    } __DEVICE_INDEX__;

  }* Data;

};
```

Listing 34: Example of request body for GetActiveDeviceInfo Inverter request

```
{
   "Body" : {
      "Data" : {
         "1" : {
```

[1] Supported since version 3.3.4-5
[2] Supported since version 3.6.1-3
[3] Not listed and provided on Generation 24

```
            "DT" : 102,
                  "Serial": "27135399"
         },
         "2" : {
            "DT" : 86
         },
         "3" : {
            "DT" : 106
         },
         "55" : {
            "DT" : 224
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "Inverter"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-12T15:30:57+02:00"
   }
}
```

Listing 35: Example of request body for GetActiveDeviceInfo SensorCard request

```
{
   "Body" : {
      "Data" : {
         "1" : {
            "ChannelNames" : [
                "Temperature␣1",
                "Temperature␣2",
                "Irradiation",
                "Digital␣1",
                "Digital␣2",
                "Current"
             ],
            "DT" : 254
         },
         "2" : {
            "ChannelNames" : [
               "Temperature␣1",
               "Temperature␣2",
               "Irradiation",
               "Digital␣1",
               "Digital␣2",
               "Current"
             ],
            "DT" : 254
         }
      }
   },
  "Head" : {
    "RequestArguments" : {
      "DeviceClass" : "SensorCard"
    },
    "Status" : {
      "Code" : 0,
      "Reason" : "",
      "UserMessage" : ""
    },
    "Timestamp" : "2018-03-01T14:41:12+01:00"
  }
```

```
}
```

Listing 36: Reply body for GetActiveDeviceInfo deviceclass=Inverter on Generation 24

```
{
   "Body" : {
      "Data" : {
         "1" : {
            "DT" : 1,
            "Serial" : "29091000975090007"
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "Inverter"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-08-28T09:02:48+00:00"
   }
}
```

☞ The item 'DT' is not valid for Inverters on Generation 24

3.7.5 DeviceClass is System

Listing 37: Object structure of request body for GetActiveDeviceInfo request

```
object {

  # Collection of objects with infos about one inverter,
  # mapped by inverter index.
  object {

    #name of DeviceClass
    object {

      # Info about a single device.
      # Name of object is the device index.
      object {

        # Device type only for Inverter, SensorCard or StringControl. others have -1
        integer DT;

        # Optional attribute: serialnumber
        string  Serial;

        # This object only exists for SensorCard device class
        array { string } ChannelNames;

      } __DEVICE_INDEX__;

    } __DEVICE_CLASS__;

  }* Data;

};
```

Listing 38: Example of request body for GetActiveDeviceInfo request on non hybrid inverter systems

```
{
```

```
"Body" : {
   "Data" : {
     "Inverter" : {
        "1" : {
           "DT" : 102,
           "Serial" : "27135399"
        },
        "2" : {
           "DT" : 86
        },
        "3" : {
           "DT" : 106
        },
        "55" : {
           "DT" : 224
        }
     },
     "Meter" : {
        "0" : {
           "DT" : -1,
           "Serial" : "16420055"
        },
        "2" : {
           "DT" : -1,
           "Serial" : "475619"
        },
        "3" : {
           "DT" : -1,
           "Serial" : "17362721"
        }
     },
     "Ohmpilot" : {
        "0" : {
           "DT" : -1,
           "Serial" : "12345678"
        }
     },
     "SensorCard" : {
       "1" : {
          "ChannelNames" : [
             "Temperature␣1",
             "Temperature␣2",
             "Irradiation",
             "Digital␣1",
             "Digital␣2",
             "Current"
          ],
          "DT" : 254
       }
     },
     "Storage" : {},
     "StringControl" : {
        "3" : {
           "DT" : 253
        }
     }
   }
},
"Head" : {
   "RequestArguments" : {
      "DeviceClass" : "System"
   },
   "Status" : {
      "Code" : 0,
      "Reason" : "",
      "UserMessage" : ""
   },
```

```
      "Timestamp" : "2019-06-12T15:30:59+02:00"
   }
}
```

Listing 39: Example of request body for GetActiveDeviceInfo request on hybrid inverter systems

```
{
   "Body" : {
      "Data" : {
         "Inverter" : {
            "1" : {
               "DT" : 99
            }
         },
         "Meter" : {
            "0" : {
               "DT" : -1,
               "Serial" : "16250161"
            }
         },
         "Ohmpilot" : {},
         "SensorCard" : {},
         "Storage" : {
            "0" : {
               "DT" : -1,
               "Serial" : "26175063"
            }
         },
         "StringControl" : {}
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "System"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-12T15:30:59+02:00"
   }
}
```

Listing 40: Reply body for GetActiveDeviceInfo deviceclass=Inverter on Generation 24

```
{
   "Body" : {
      "Data" : {
         "1" : {
            "DT" : 1,
            "Serial" : "29091000975090007"
         },
         "Meter" : {
            "0" : {
               "DT" : -1,
               "Serial" : "18142251"
            }
         },
         "Ohmpilot" : {
            "0" : {
               "DT" : -1,
               "Serial" : "28136344"
            }
         },
         "Storage" : {}
      }
```

```
    },
    "Head" : {
        "RequestArguments" : {
            "DeviceClass" : "System"
        },
        "Status" : {
            "Code" : 0,
            "Reason" : "",
            "UserMessage" : ""
        },
        "Timestamp" : "2019-08-28T08:59:41+00:00"
    }
}
```


☞ The item 'DT' is not valid for Inverters on Generation 24

3.8 *GetMeterRealtimeData* request

This request provides detailed information about Meter devices. Inactive channels are not included in the response and may vary depending on used metering device and software version. Take care about permanently or temporary missing channels when processing this response.

3.8.1 Availability

Platform	Since version
Fronius Hybrid	ALWAYS
Fronius Non Hybrid	3.3.4-8
Fronius Generation 24	ALWAYS

3.8.2 URL for HTTP requests

`/solar_api/v1/GetMeterRealtimeData.cgi`

3.8.3 Parameters

Parameter	Type	Range/Values/Pattern	Description
Scope	String	"System" "Device"	Mandatory
DeviceId	String	0..65535	Mandatory on non system scope

3.8.4 Devicetypes and provided channels

Group	Fronius			CCS WattNode					Generic
Model	IME 63A	IME 63A-1	IME 5kA-3	WNC-3D-240-MB	WNC-3D-480-MB	WND-3D-240-MB	WND-3D-480-MB	WND-3Y-600-MB	SunsSpec Meter
Compatibility	hw=0 and sw=2.7 hw=1 and (sw=2.9 or sw=3.0)	hw=1 and sw is 3.00, 3.01, 3.03 or 3.04)	hw=0 and sw=1.9	sw=26	sw=26	hw=16 and sw is 21 or 22 hw=17 and sw is 29 or 30			model=307 or model=70307
Channel	provided (m…mandatory, -…never, o…optional)								
Details / Manufacturer	m	m	m	m	m	m	m	m	o
Details / Model	m	m	m	m	m	m	m	m	o
Details / Serial	m	m	m	m	m	m	m	m	o
Current_AC_Phase_1	m	m	m	m	m	m	m	m	o
Current_AC_Phase_2	m	-	m	o	o	o	o	o	o
Current_AC_Phase_3	m	-	m	o	o	o	o	o	o
Current_AC_Sum	-	m	-	m	m	m	m	m	o
Enable	m	m	m	m	m	m	m	m	m
EnergyReactive_VArAC_Phase_1_Consumed	-	m	-	-	-	-	-	-	-
EnergyReactive_VArAC_Phase_1_Produced	-	m	-	-	-	-	-	-	-
EnergyReactive_VArAC_Sum_Consumed	m	m	m	-	-	-	-	-	-
EnergyReactive_VArAC_Sum_Produced	m	m	m	-	-	-	-	-	-
EnergyReal_WAC_Minus_Absolute	m	m	m	m	m	m	m	m	m
EnergyReal_WAC_Phase_1_Consumed	-	m	-	m	m	m	m	m	o
EnergyReal_WAC_Phase_1_Produced	-	m	-	m	m	m	m	m	o
EnergyReal_WAC_Phase_2_Consumed	-	-	-	o	o	o	o	o	o
EnergyReal_WAC_Phase_2_Produced	-	-	-	o	o	o	o	o	o
EnergyReal_WAC_Phase_3_Consumed	-	-	-	o	o	o	o	o	o
EnergyReal_WAC_Phase_3_Produced	-	-	-	o	o	o	o	o	o
EnergyReal_WAC_Plus_Absolute	m	m	m	m	m	m	m	m	m
EnergyReal_WAC_Sum_Consumed	m	m	m	m	m	m	m	m	m
EnergyReal_WAC_Sum_Produced	m	m	m	m	m	m	m	m	m
Frequency_Phase_Average	m	m	m	m	m	m	m	m	m
Meter_Location_Current	m	m	m	m	m	m	m	m	m
PowerApparent_S_Phase_1	m	m	m	m	m	m	m	m	o
PowerApparent_S_Phase_2	m	-	m	o	o	o	o	o	o
PowerApparent_S_Phase_3	m	-	m	o	o	o	o	o	o
PowerApparent_S_Sum	m	m	m	m	m	m	m	m	m
PowerFactor_Phase_1	m	m	m	m	m	m	m	m	o
PowerFactor_Phase_2	m	-	m	o	o	o	o	o	o
PowerFactor_Phase_3	m	-	m	o	o	o	o	o	o
PowerFactor_Sum	m	m	m	m	m	m	m	m	m
PowerReactive_Q_Phase_1	m	m	m	m	m	m	m	m	o
PowerReactive_Q_Phase_2	m	-	m	o	o	o	o	o	o
PowerReactive_Q_Phase_3	m	-	m	o	o	o	o	o	o
PowerReactive_Q_Sum	m	m	m	m	m	m	m	m	m

Group	Fronius			CCS WattNode					Generic
Model	IME 63A	IME 63A-1	IME 5kA-3	WNC-3D-240-MB	WNC-3D-480-MB	WND-3D-240-MB	WND-3D-480-MB	WND-3Y-600-MB	SunsSpec Meter
PowerReal_P_Phase_1	m	m	m	m	m	m	m	m	o
PowerReal_P_Phase_2	m	-	m	o	o	o	o	o	o
PowerReal_P_Phase_3	m	-	m	o	o	o	o	o	o
PowerReal_P_Sum	m	m	m	m	m	m	m	m	m
TimeStamp	m	m	m	m	m	m	m	m	m
Visible	m	m	m	m	m	m	m	m	m
Voltage_AC_PhaseToPhase_12	m	-	m	o	o	o	o	o	o
Voltage_AC_PhaseToPhase_23	m	-	m	o	o	o	o	o	o
Voltage_AC_PhaseToPhase_31	m	-	m	o	o	o	o	o	o
Voltage_AC_Phase_1	m	m	m	m	m	m	m	m	o
Voltage_AC_Phase_2	m	-	m	o	o	o	o	o	o
Voltage_AC_Phase_3	m	-	m	o	o	o	o	o	o
Voltage_AC_Phase_Average	-	-	-	m	m	m	m	m	o

Some values are optional since meter is able to operate on one or three phases.

3.8.5 Channel Descriptions

Name	Description
Enable	1...enabled, 0...disabled
Visible	1...use values, 0...incomplete or outdated values
Current_AC_Phase_*	absolute values
Meter_Location_Current	0...grid interconnection point (primary meter) 1...load (primary meter) 3...external generator (secondary meters)(multiple) 256-511 subloads (secondary meters)(unique)
EnergyReal_WAC_Plus/Minus_Absolute	system specific view
EnergyReal_WAC_Sum_Consumed/Produced	meter specific view

The values EnergyReal_WAC_Sum_Produced and EnergyReal_WAC_Sum_Consumed represent the values for the Smart Meter itself. The values EnergyReal_WAC_Minus_Absolute and EnergyReal_WAC_Plus_Absolute represent the values for Solar.web. Now it depends where the Smart Meter is installed (Feed-In-Point or Consumption-Path), so that either EnergyReal_WAC_Minus_Absolute is the same as EnergyReal_WAC_Sum_Produced or EnergyReal_WAC_Sum_Consumed.

3.8.6 System-Request

Listing 41: Object structure of response body for GetMeterRealtimeData request

```
# Collection of objects with infos about multiple Meters,
# mapped by serial number.
object {

  #list of single device objects
  object {

    #optional detailed information about device
    #supported since:
    #    Fronius Symo Hybrid     : with version greater than or equal to 1.1.2-14
    #    Non Fronius Symo Hybrid : with version greater than or equal to 3.3.6-14
    object {
       string Serial;

       string Model;
```

```
        string Manufacturer;

    } Details;

    #channels of device (textual name and value)
    number * __CHANNEL_NAME__;

  } * DeviceId;

} Data ;
```

Listing 42: Reply body for GetMeterRealtimeData System request


```
{
   "Body" : {
      "Data" : {
         "0" : {
            "Current_AC_Phase_1" : 0.74199999999999999,
            "Current_AC_Phase_2" : 0.63200000000000001,
            "Current_AC_Phase_3" : 0.65400000000000003,
            "Details" : {
               "Manufacturer" : "Fronius",
               "Model" : "Smart␣Meter␣63A",
               "Serial" : "15160189"
            },
            "Enable" : 1,
            "EnergyReactive_VArAC_Sum_Consumed" : 9156420,
            "EnergyReactive_VArAC_Sum_Produced" : 87894450,
            "EnergyReal_WAC_Minus_Absolute" : 1642802,
            "EnergyReal_WAC_Plus_Absolute" : 19838697,
            "EnergyReal_WAC_Sum_Consumed" : 19838697,
            "EnergyReal_WAC_Sum_Produced" : 1642802,
            "Frequency_Phase_Average" : 50,
            "Meter_Location_Current" : 0,
            "PowerApparent_S_Phase_1" : 172.36660000000001,
            "PowerApparent_S_Phase_2" : 147.00319999999999,
            "PowerApparent_S_Phase_3" : 152.57820000000001,
            "PowerApparent_S_Sum" : 31,
            "PowerFactor_Phase_1" : 0,
            "PowerFactor_Phase_2" : 0.97999999999999998,
            "PowerFactor_Phase_3" : 1,
            "PowerFactor_Sum" : 0.81999999999999995,
            "PowerReactive_Q_Phase_1" : 12.550000000000001,
            "PowerReactive_Q_Phase_2" : 5.8099999999999996,
            "PowerReactive_Q_Phase_3" : 0,
            "PowerReactive_Q_Sum" : 18.359999999999999,
            "PowerReal_P_Phase_1" : 0,
            "PowerReal_P_Phase_2" : -40.560000000000002,
            "PowerReal_P_Phase_3" : 15.029999999999999,
            "PowerReal_P_Sum" : -25.530000000000001,
            "TimeStamp" : 1561364909,
            "Visible" : 1,
            "Voltage_AC_PhaseToPhase_12" : 402.60000000000002,
            "Voltage_AC_PhaseToPhase_23" : 403.5,
            "Voltage_AC_PhaseToPhase_31" : 403.19999999999999,
            "Voltage_AC_Phase_1" : 232.30000000000001,
            "Voltage_AC_Phase_2" : 232.59999999999999,
            "Voltage_AC_Phase_3" : 233.30000000000001
         },
         "1" : {
            "Current_AC_Phase_1" : -0.58310449123382568,
            "Current_AC_Phase_2" : -0.67854827642440796,
            "Current_AC_Phase_3" : -0.7008516788482666,
            "Current_AC_Sum" : -1.9625044465065002,
            "Details" : {
               "Manufacturer" : "Fronius",
               "Model" : "CCS␣WattNode␣WNC-3D-480-MB",
```

```
            "Serial" : "186477"
         },
         "Enable" : 1,
         "EnergyReal_WAC_Minus_Absolute" : 7336854,
         "EnergyReal_WAC_Phase_1_Consumed" : 1320806,
         "EnergyReal_WAC_Phase_1_Produced" : 1933071,
         "EnergyReal_WAC_Phase_2_Consumed" : 158238,
         "EnergyReal_WAC_Phase_2_Produced" : 3043466,
         "EnergyReal_WAC_Phase_3_Consumed" : 179872,
         "EnergyReal_WAC_Phase_3_Produced" : 2912264,
         "EnergyReal_WAC_Plus_Absolute" : 1106969,
         "EnergyReal_WAC_Sum_Consumed" : 1106969,
         "EnergyReal_WAC_Sum_Produced" : 7336854,
         "Frequency_Phase_Average" : 50.116844177246094,
         "Meter_Location_Current" : 256,
         "PowerApparent_S_Phase_1" : 135.04127502441406,
         "PowerApparent_S_Phase_2" : 160.43267822265625,
         "PowerApparent_S_Phase_3" : 163.04228210449219,
         "PowerApparent_S_Sum" : 458.5162353515625,
         "PowerFactor_Phase_1" : 1,
         "PowerFactor_Phase_2" : 1,
         "PowerFactor_Phase_3" : 1,
         "PowerFactor_Sum" : 1,
         "PowerReactive_Q_Phase_1" : 0,
         "PowerReactive_Q_Phase_2" : 0,
         "PowerReactive_Q_Phase_3" : 0,
         "PowerReactive_Q_Sum" : 0,
         "PowerReal_P_Phase_1" : -135.04127502441406,
         "PowerReal_P_Phase_2" : -160.43267822265625,
         "PowerReal_P_Phase_3" : -163.04228210449219,
         "PowerReal_P_Sum" : -458.5162353515625,
         "TimeStamp" : 1561364987,
         "Visible" : 1,
         "Voltage_AC_PhaseToPhase_12" : 405.32907104492188,
         "Voltage_AC_PhaseToPhase_23" : 406.23068237304688,
         "Voltage_AC_PhaseToPhase_31" : 402.03070068359375,
         "Voltage_AC_Phase_1" : 231.59017944335938,
         "Voltage_AC_Phase_2" : 236.43516540527344,
         "Voltage_AC_Phase_3" : 232.63450622558594,
         "Voltage_AC_Phase_Average" : 233.55328369140625
      },
      "2" : {
         "Current_AC_Phase_1" : 0.57899999999999996,
         "Current_AC_Sum" : 0.57899999999999996,
         "Details" : {
            "Manufacturer" : "Fronius",
            "Model" : "Smart␣Meter␣63A-1",
            "Serial" : "15160009"
         },
         "Enable" : 1,
         "EnergyReactive_VArAC_Phase_1_Consumed" : 260,
         "EnergyReactive_VArAC_Phase_1_Produced" : 8261790,
         "EnergyReactive_VArAC_Sum_Consumed" : 260,
         "EnergyReactive_VArAC_Sum_Produced" : 8261790,
         "EnergyReal_WAC_Minus_Absolute" : 0,
         "EnergyReal_WAC_Phase_1_Consumed" : 5670793,
         "EnergyReal_WAC_Phase_1_Produced" : 0,
         "EnergyReal_WAC_Plus_Absolute" : 5670793,
         "EnergyReal_WAC_Sum_Consumed" : 5670793,
         "EnergyReal_WAC_Sum_Produced" : 0,
         "Frequency_Phase_Average" : 50,
         "Meter_Location_Current" : 257,
         "PowerApparent_S_Phase_1" : 135.19,
         "PowerApparent_S_Sum" : 135.19,
         "PowerFactor_Phase_1" : 0.96999999999999997,
         "PowerFactor_Sum" : 0.96999999999999997,
         "PowerReactive_Q_Phase_1" : -22.629999999999999,
```

```
            "PowerReactive_Q_Sum" : -22.629999999999999,
            "PowerReal_P_Phase_1" : 132.09999999999999,
            "PowerReal_P_Sum" : 132.09999999999999,
            "TimeStamp" : 1561365038,
            "Visible" : 1,
            "Voltage_AC_Phase_1" : 233.5
         },
         "3" : {
            "Details" : {
               "Manufacturer" : "Fronius",
               "Model" : "SO Meter at inverter 40",
               "Serial" : "n.a."
            },
            "Enable" : 1,
            "Meter_Location_Current" : 258,
            "TimeStamp" : 1560942897,
            "EnergyReal_WAC_Minus_Relative": 0,
            "EnergyReal_WAC_Plus_Relative": 0,
            "PowerReal_P_Sum": 0,
            "Visible" : 1
         },
        "4": {
          "Current_AC_Phase_1": 0,
          "Current_AC_Phase_2": 0,
          "Current_AC_Phase_3": 0,
          "Current_AC_Sum": 0,
          "Details": {
            "Manufacturer": "Fronius",
            "Model": "CCS WattNode WND-3Y-600-MB",
            "Serial": "475619"
          },
          "Enable": 1,
          "EnergyReal_WAC_Minus_Absolute": 3321,
          "EnergyReal_WAC_Phase_1_Consumed": 3321,
          "EnergyReal_WAC_Phase_1_Produced": 10996,
          "EnergyReal_WAC_Phase_2_Consumed": 0,
          "EnergyReal_WAC_Phase_2_Produced": 0,
          "EnergyReal_WAC_Phase_3_Consumed": 0,
          "EnergyReal_WAC_Phase_3_Produced": 14,
          "EnergyReal_WAC_Plus_Absolute": 11010,
          "EnergyReal_WAC_Sum_Consumed": 3321,
          "EnergyReal_WAC_Sum_Produced": 11010,
          "Frequency_Phase_Average": 49.9833869934082,
          "Meter_Location_Current": 259,
          "PowerApparent_S_Phase_1": 0,
          "PowerApparent_S_Phase_2": 0,
          "PowerApparent_S_Phase_3": 0,
          "PowerApparent_S_Sum": 0,
          "PowerFactor_Phase_1": 1,
          "PowerFactor_Phase_2": 1,
          "PowerFactor_Phase_3": 1,
          "PowerFactor_Sum": 1,
          "PowerReactive_Q_Phase_1": 0,
          "PowerReactive_Q_Phase_2": 0,
          "PowerReactive_Q_Phase_3": 0,
          "PowerReactive_Q_Sum": 0,
          "PowerReal_P_Phase_1": 0,
          "PowerReal_P_Phase_2": 0,
          "PowerReal_P_Phase_3": 0,
          "PowerReal_P_Sum": 0,
          "TimeStamp": 1519911921,
          "Visible": 1,
          "Voltage_AC_PhaseToPhase_12": 238.15383911132812,
          "Voltage_AC_PhaseToPhase_23": 0,
          "Voltage_AC_PhaseToPhase_31": 232.91676330566406,
          "Voltage_AC_Phase_1": 404.52679443359375,
          "Voltage_AC_Phase_2": 231.70884704589844,
```

```
          "Voltage_AC_Phase_3": 232.72479248046875,
          "Voltage_AC_Phase_Average": 289.6534729003906
        },
        "5": {
          "Current_AC_Phase_1": 0,
          "Current_AC_Phase_2": 0,
          "Current_AC_Phase_3": 0,
          "Details": {
            "Manufacturer": "Fronius",
            "Model": "Smart␣Meter␣50kA-3",
            "Serial": "17362721"
          },
          "Enable": 1,
          "EnergyReactive_VArAC_Sum_Consumed": 34,
          "EnergyReactive_VArAC_Sum_Produced": 174,
          "EnergyReal_WAC_Minus_Absolute": 3940,
          "EnergyReal_WAC_Plus_Absolute": 434,
          "EnergyReal_WAC_Sum_Consumed": 3940,
          "EnergyReal_WAC_Sum_Produced": 434,
          "Frequency_Phase_Average": 49.900000743567944,
          "Meter_Location_Current": 3,
          "PowerApparent_S_Phase_1": 0,
          "PowerApparent_S_Phase_2": 0,
          "PowerApparent_S_Phase_3": 0,
          "PowerApparent_S_Sum": 0,
          "PowerFactor_Phase_1": 0.9999999776482582,
          "PowerFactor_Phase_2": 0.9999999776482582,
          "PowerFactor_Phase_3": 0.9999999776482582,
          "PowerFactor_Sum": 0.9999999776482582,
          "PowerReactive_Q_Phase_1": 0,
          "PowerReactive_Q_Phase_2": 0,
          "PowerReactive_Q_Phase_3": 0,
          "PowerReactive_Q_Sum": 0,
          "PowerReal_P_Phase_1": 0,
          "PowerReal_P_Phase_2": 0,
          "PowerReal_P_Phase_3": 0,
          "PowerReal_P_Sum": 0,
          "TimeStamp": 1519911921,
          "Visible": 1,
          "Voltage_AC_PhaseToPhase_12": 404.90001923171803,
          "Voltage_AC_PhaseToPhase_23": 404.50001921271905,
          "Voltage_AC_PhaseToPhase_31": 404.4000192079693,
          "Voltage_AC_Phase_1": 233.70001110015437,
          "Voltage_AC_Phase_2": 233.80001110490412,
          "Voltage_AC_Phase_3": 233.3000110811554
        }
      }
  },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "Meter",
         "Scope" : "System"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-24T10:31:54+02:00"
   }
}
```


Examples for Generation 24 are ident.

3.8.7 Device-Request

Listing 43: Object structure of response body for GetMeterRealtimeData request

```
# object with detailed informations about one Meter,
object {

    object {
      string Serial;

      string Model;

      string Manufacturer;
    } Details;

    #channels of device (textual name and value)
    number * __CHANNEL_NAME__;

} Data ;
```

Listing 44: Reply body for GetMeterRealtimeData Device request

```
{
   "Body" : {
      "Data" : {
         "Current_AC_Phase_1" : 0.61899999999999999,
         "Current_AC_Phase_2" : 0.68799999999999994,
         "Current_AC_Phase_3" : 0.55100000000000005,
         "Details" : {
            "Manufacturer" : "Fronius",
            "Model" : "Smart␣Meter␣63A",
            "Serial" : "15480258"
         },
         "Enable" : 1,
         "EnergyReactive_VArAC_Sum_Consumed" : 2183700,
         "EnergyReactive_VArAC_Sum_Produced" : 47100,
         "EnergyReal_WAC_Minus_Absolute" : 4075753,
         "EnergyReal_WAC_Plus_Absolute" : 941840,
         "EnergyReal_WAC_Sum_Consumed" : 941840,
         "EnergyReal_WAC_Sum_Produced" : 4075753,
         "Frequency_Phase_Average" : 50,
         "Meter_Location_Current" : 0,
         "PowerApparent_S_Phase_1" : 143.9794,
         "PowerApparent_S_Phase_2" : 159.5472,
         "PowerApparent_S_Phase_3" : 127.44630000000002,
         "PowerApparent_S_Sum" : 211.36000000000001,
         "PowerFactor_Phase_1" : 0.97999999999999998,
         "PowerFactor_Phase_2" : 1,
         "PowerFactor_Phase_3" : 1,
         "PowerFactor_Sum" : 1,
         "PowerReactive_Q_Phase_1" : 9.9000000000000004,
         "PowerReactive_Q_Phase_2" : 0,
         "PowerReactive_Q_Phase_3" : 4.7999999999999998,
         "PowerReactive_Q_Sum" : 14.699999999999999,
         "PowerReal_P_Phase_1" : -75,
         "PowerReal_P_Phase_2" : -74.280000000000001,
         "PowerReal_P_Phase_3" : -62.079999999999998,
         "PowerReal_P_Sum" : -211.36000000000001,
         "TimeStamp" : 1560430330,
         "Visible" : 1,
         "Voltage_AC_PhaseToPhase_12" : 402.30000000000001,
         "Voltage_AC_PhaseToPhase_23" : 401.10000000000002,
         "Voltage_AC_PhaseToPhase_31" : 401.69999999999999,
         "Voltage_AC_Phase_1" : 232.59999999999999,
         "Voltage_AC_Phase_2" : 231.90000000000001,
         "Voltage_AC_Phase_3" : 231.30000000000001
      }
   },
   "Head" : {
      "RequestArguments" : {
```

```
            "DeviceClass" : "Meter",
            "DeviceId" : "0",
            "Scope" : "Device"
         },
         "Status" : {
            "Code" : 0,
            "Reason" : "",
            "UserMessage" : ""
         },
         "Timestamp" : "2019-06-13T14:52:10+02:00"
      }
}
```


Examples for Generation 24 are ident.

3.9 *GetStorageRealtimeData* request

This request provides detailed information about batteries. Inactive channels are not included in the response and may vary depended on used battery and software version. Take care about permanently or temporary missing channels when processing this response.

3.9.1 Availability

Platform	Since version
Fronius Hybrid	1.1.2-13
Fronius Non Hybrid	NOT AVAILABLE
Fronius Generation 24	ALWAYS

☞ API is available but always return an error on Generation 24

3.9.2 3rd Party Batteries

will be displayed since HybridManager version 1.13.1-x and SolarAPI Version 1.5-17. Older versions reported an error:

Listing 45: Former response body for GetStorageRealtimeData request using BYD Box


```
{
   "Body" : {
      "Data" : {}
   },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "Storage",
         "DeviceId" : "0",
         "Scope" : "Device"
      },
      "Status" : {
         "Code" : 255,
         "Reason" : "battery␣type␣'BYD'␣is␣not␣supported",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-03T17:01:01+02:00"
   }
}
```


3.9.3 Supported

manufacturer	model	versions
Fronius	Fronius Solar Battery	bms sw 0x18XX
BYD	BYD Battery-Box HV	protocol 0x0 - 0x1ffff
LG-Chem	Resu H	dcdc sw 0x5046 - 0x50ff dcdc sw 0x7046 - 0x70ff

☞ If storage version is incompatible, it will be operative and an inconsistency warning will be shown to update the storage if possible.

3.9.4 URL for HTTP requests

`/solar_api/v1/GetStorageRealtimeData.cgi`

3.9.5 Parameters

Parameter	Type	Range/Values/Pattern	Description
Scope	String	”System” ”Device”	Mandatory
DeviceId	String	0..65535	Mandatory on non system scope

3.9.6 Reference to manual

Reference to Fronius Energy Package and 3rd Party support can be found here:
`https://www.fronius.com/~/downloads/Solar%20Energy/Operating%20Instructions/42%2C0426%2C0222%2CEN.pdf`

3.9.7 Channel Descriptions

Table 3: Channel and description for control section

Name of channel	Description	Available		
		Fronius Solar Battery	LG Chem Resu H	BYD Box
Details / Manufacturer	name of manufacturer	always	always	always
Details / Model	model of battery	always	always	always
Details / Serial	unique identification serial	always	always	always
TimeStamp	last timestamp data has been refrehsed	always	always	always
Enable	device is managed (1.0) or disconnected (0.0)	always	always	always
StateOfCharge_Relative	relative charged capacity in %	always	always	always
Capacity_Maximum	current max capacity	always	always	always
DesignedCapacity	max designed capacity	always	always	always
Current_DC	battery output current (+ charging)	always	always	always
Voltage_DC	battery output voltage	always	always	always
Temperature_Cell	temperature in degree celsius	always	always	always

Table 2: Channel and value description

<table>
<tr><th>Name of channel</th><th>Description</th></tr>
<tr><td>Status_BatteryCell</td><td>Fronius Solar Battery at section Modules

Previuos and current state of a battery cell. One Byte printed in hexadecimal. 0xYX (Y: Current status, X: Previous status)
Meaning of numerical status codes:
<table>
<tr><th>Status value</th><th>Description</th></tr>
<tr><td>0_{16}</td><td>RESERVED</td></tr>
<tr><td>1_{16}</td><td>Pre Charge</td></tr>
<tr><td>2_{16}</td><td>Initial</td></tr>
<tr><td>3_{16}</td><td>Normal Charge</td></tr>
<tr><td>4_{16}</td><td>Charge Terminate</td></tr>
<tr><td>5_{16}</td><td>Normal Discharge</td></tr>
<tr><td>6_{16}</td><td>Over Voltage</td></tr>
<tr><td>7_{16}</td><td>Over Discharge</td></tr>
<tr><td>8_{16}</td><td>RESERVED</td></tr>
<tr><td>9_{16}</td><td>Over Temp Charge</td></tr>
<tr><td>A_{16}</td><td>Over Current Charge</td></tr>
<tr><td>B_{16}</td><td>Over Temp Discharge</td></tr>
<tr><td>C_{16}</td><td>Over Current Discharge</td></tr>
<tr><td>D_{16}</td><td>Cell Unbalance</td></tr>
<tr><td>E_{16}</td><td>Charge Suspend</td></tr>
<tr><td>F_{16}</td><td>RESERVED</td></tr>
</table></td></tr>
<tr><td>Status_BatteryCell</td><td>for LG-Chem Resu H at section Controller

<table>
<tr><th>Status value</th><th>Description</th></tr>
<tr><td>1</td><td>STANDBY</td></tr>
<tr><td>3</td><td>ENABLED</td></tr>
<tr><td>5</td><td>FAULTED</td></tr>
<tr><td>10</td><td>SLEEP</td></tr>
</table></td></tr>
<tr><td>Status_BatteryCell</td><td>for BYD Box at section Controller

<table>
<tr><th>Status value</th><th>Description</th></tr>
<tr><td>0</td><td>STANDBY</td></tr>
<tr><td>1</td><td>INACTIVE</td></tr>
<tr><td>2</td><td>DARKSTART</td></tr>
<tr><td>3</td><td>ACTIVE</td></tr>
<tr><td>4</td><td>FAULT</td></tr>
<tr><td>5</td><td>UPDATING</td></tr>
</table></td></tr>
</table>

Table 4: Channel and description for modul section (only Solar Battery provides modul informations)

Name of channel	Description	Available		
		Fronius Solar Battery	LG Chem Resu H	BYD Box
Details / Manufacturer	manufacturer	always	-	-
Details / Model	model identifier	always	-	-
Details / Serial	unique identifier	always	-	-
Capacity_Maximum		always	-	-
Current_DC		always	-	-
CycleCount_BatteryCell		always	-	-
DesignedCapacity		always	-	-
Enable		always	-	-
StateOfCharge_Relative		always	-	-
Status_BatteryCell		always	-	-
Temperature_Cell		always	-	-
Temperature_Cell_Maximum		always	-	-
Temperature_Cell_Minimum		always	-	-
TimeStamp		always	-	-
Voltage_DC		always	-	-
Voltage_DC_Maximum_Cell		always	-	-
Voltage_DC_Minimum_Cell		always	-	-

3.9.8 System-Request

Listing 46: Object structure of response body for GetStorageRealtimeData request

```
# object with detailed informations about one Battery
object {

  object {

    object {
      object {
          # serial number of battery
          string Serial;

          # device type identifier
          string Model;

          # Solar battery manufacturer changed from "Fronius International" to "Fronius"
          #    Solar API Version       : 1.5-16
          #    Fronius Symo Hybrid     : 1.12.1-3
          string Manufacturer;
      } Details;

      #channels of device (textual name and value)
      number * __CHANNEL_NAME__;

    } Controller;

    array {

      object {
        string Serial;

        string Model;

        string Manufacturer;
      } Details;

      #channels of device (textual name and value)
```

```
      number * __CHANNEL_NAME__;

    } * Modules;

  } * DeviceId;   // 0...65535

} Data ;
```

Manufacturer has been updated at version HM-1.12.1-X from ”Fronius International” to ”Fronius”

Listing 47: Reply body for GetStorageRealtimeData System request (Solar Battery)


```
{
   "Body" : {
      "Data" : {
         "0" : {
            "Controller" : {
               "Capacity_Maximum" : 7200,
               "Current_DC" : 1.1200000000000001,
               "DesignedCapacity" : 7200,
               "Details" : {
                  "Manufacturer" : "Fronius",
                  "Model" : "Fronius␣Solar␣Battery",
                  "Serial" : "26175063"
               },
               "Enable" : 1,
               "StateOfCharge_Relative" : 55,
               "Temperature_Cell" : 26.150000000000034,
               "TimeStamp" : 1560346272,
               "Voltage_DC" : 318.80000000000001,
               "Voltage_DC_Maximum_Cell" : 3.3290000000000002,
               "Voltage_DC_Minimum_Cell" : 3.3159999999999998
            },
            "Modules" : [
               {
                  "Capacity_Maximum" : 1200,
                  "Current_DC" : 1.1100000000000001,
                  "CycleCount_BatteryCell" : 255,
                  "DesignedCapacity" : 1200,
                  "Details" : {
                     "Manufacturer" : "Sony",
                     "Model" : "unknown",
                     "Serial" : "S012002885␣"
                  },
                  "Enable" : 1,
                  "StateOfCharge_Relative" : 55,
                  "Status_BatteryCell" : 53,
                  "Temperature_Cell" : 27.25,
                  "Temperature_Cell_Maximum" : 27.75,
                  "Temperature_Cell_Minimum" : 26.950000000000045,
                  "TimeStamp" : 1560346263,
                  "Voltage_DC" : 53.142000000000003,
                  "Voltage_DC_Maximum_Cell" : 3.3239999999999998,
                  "Voltage_DC_Minimum_Cell" : 3.3140000000000001
               },
               {
                  "Capacity_Maximum" : 1200,
                  "Current_DC" : 1.1200000000000001,
                  "CycleCount_BatteryCell" : 257,
                  "DesignedCapacity" : 1200,
                  "Details" : {
                     "Manufacturer" : "Sony",
                     "Model" : "unknown",
                     "Serial" : "S012002843␣"
                  },
                  "Enable" : 1,
                  "StateOfCharge_Relative" : 55,
                  "Status_BatteryCell" : 53,
```

```
            "Temperature_Cell" : 26.650000000000034,
            "Temperature_Cell_Maximum" : 27.150000000000034,
            "Temperature_Cell_Minimum" : 26.350000000000023,
            "TimeStamp" : 1560346263,
            "Voltage_DC" : 53.137,
            "Voltage_DC_Maximum_Cell" : 3.3279999999999998,
            "Voltage_DC_Minimum_Cell" : 3.3159999999999998
         },
         {
            "Capacity_Maximum" : 1200,
            "Current_DC" : 1.1299999999999999,
            "CycleCount_BatteryCell" : 257,
            "DesignedCapacity" : 1200,
            "Details" : {
               "Manufacturer" : "Sony",
               "Model" : "unknown",
               "Serial" : "S012002844␣"
            },
            "Enable" : 1,
            "StateOfCharge_Relative" : 55,
            "Status_BatteryCell" : 53,
            "Temperature_Cell" : 26.450000000000045,
            "Temperature_Cell_Maximum" : 27.050000000000011,
            "Temperature_Cell_Minimum" : 26.25,
            "TimeStamp" : 1560346263,
            "Voltage_DC" : 53.164000000000001,
            "Voltage_DC_Maximum_Cell" : 3.3290000000000002,
            "Voltage_DC_Minimum_Cell" : 3.319
         },
         {
            "Capacity_Maximum" : 1200,
            "Current_DC" : 1.1299999999999999,
            "CycleCount_BatteryCell" : 254,
            "DesignedCapacity" : 1200,
            "Details" : {
               "Manufacturer" : "Sony",
               "Model" : "unknown",
               "Serial" : "S012002838␣"
            },
            "Enable" : 1,
            "StateOfCharge_Relative" : 55,
            "Status_BatteryCell" : 53,
            "Temperature_Cell" : 26.150000000000034,
            "Temperature_Cell_Maximum" : 26.75,
            "Temperature_Cell_Minimum" : 25.75,
            "TimeStamp" : 1560346263,
            "Voltage_DC" : 53.158999999999999,
            "Voltage_DC_Maximum_Cell" : 3.3290000000000002,
            "Voltage_DC_Minimum_Cell" : 3.3180000000000001
         },
         {
            "Capacity_Maximum" : 1200,
            "Current_DC" : 1.1200000000000001,
            "CycleCount_BatteryCell" : 256,
            "DesignedCapacity" : 1200,
            "Details" : {
               "Manufacturer" : "Sony",
               "Model" : "unknown",
               "Serial" : "S012002884␣"
            },
            "Enable" : 1,
            "StateOfCharge_Relative" : 55,
            "Status_BatteryCell" : 53,
            "Temperature_Cell" : 25.550000000000011,
            "Temperature_Cell_Maximum" : 26.150000000000034,
            "Temperature_Cell_Minimum" : 25.350000000000023,
            "TimeStamp" : 1560346263,
```

```
                    "Voltage_DC" : 53.146000000000001,
                    "Voltage_DC_Maximum_Cell" : 3.3260000000000001,
                    "Voltage_DC_Minimum_Cell" : 3.3170000000000002
                },
                {
                    "Capacity_Maximum" : 1200,
                    "Current_DC" : 1.1200000000000001,
                    "CycleCount_BatteryCell" : 255,
                    "DesignedCapacity" : 1200,
                    "Details" : {
                        "Manufacturer" : "Sony",
                        "Model" : "unknown",
                        "Serial" : "S012002857␣"
                    },
                    "Enable" : 1,
                    "StateOfCharge_Relative" : 55,
                    "Status_BatteryCell" : 53,
                    "Temperature_Cell" : 25.25,
                    "Temperature_Cell_Maximum" : 25.75,
                    "Temperature_Cell_Minimum" : 24.950000000000045,
                    "TimeStamp" : 1560346263,
                    "Voltage_DC" : 53.156999999999996,
                    "Voltage_DC_Maximum_Cell" : 3.3260000000000001,
                    "Voltage_DC_Minimum_Cell" : 3.3199999999999998
                }
            ]
        }
    }
},
"Head" : {
    "RequestArguments" : {
        "DeviceClass" : "Storage",
        "Scope" : "System"
    },
    "Status" : {
        "Code" : 0,
        "Reason" : "",
        "UserMessage" : ""
    },
    "Timestamp" : "2019-06-12T15:31:12+02:00"
}
}
```


Listing 48: Reply body for GetStorageRealtimeData System request (BYD B-Box)


```
{
    "Body" : {
        "Data" : {
            "0" : {
                "Controller" : {
                    "Capacity_Maximum" : 11520,
                    "Current_DC" : 0,
                    "DesignedCapacity" : 11520,
                    "Details" : {
                        "Manufacturer" : "BYD",
                        "Model" : "BYD␣Battery-Box␣HV",
                        "Serial" : "400481708-00059"
                    },
                    "Enable" : 1,
                    "StateOfCharge_Relative" : 4.7000000000000002,
                    "Status_BatteryCell" : 3,
                    "Temperature_Cell" : 23.949999999999999,
                    "TimeStamp" : 1560430543,
                    "Voltage_DC" : 462.60000000000002
                },
                "Modules" : []
            }
```

```
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "Storage",
         "Scope" : "System"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-13T14:55:44+02:00"
   }
}
```

3.9.9 Device-Request

Listing 49: Object structure of response body for GetStorageRealtimeData request

```
# object with detailed informations about one Battery
object {

    object {
      object {
          #serial number of Fronius Solar Battery
          string Serial;

          string Model;

          string Manufacturer;
      } Details;

      #channels of device (textual name and value)
      number * __CHANNEL_NAME__;

    } Controller;

    array {

      object {
          string Serial;

          string Model;

          string Manufacturer;
      } Details;

      #channels of device (textual name and value)
      number * __CHANNEL_NAME__;

    } * Modules;

} Data ;
```

Listing 50: Reply body for GetStorageRealtimeData Device request (LG Chem Resu H)

```
{
   "Body" : {
      "Data" : {
         "Controller" : {
            "Capacity_Maximum" : 9800,
            "Current_DC" : 0.90000000000000002,
            "DesignedCapacity" : 9800,
            "Details" : {
               "Manufacturer" : "LG-Chem",
```

```
                "Model" : "Resu␣H",
                "Serial" : "1706179036"
            },
            "Enable" : 1,
            "StateOfCharge_Relative" : 56,
            "Status_BatteryCell" : 3,
            "Temperature_Cell" : 27.550000000000001,
            "TimeStamp" : 1560346267,
            "Voltage_DC" : 407.5
         },
         "Modules" : []
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "Storage",
         "DeviceId" : "0",
         "Scope" : "Device"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-12T15:31:08+02:00"
   }
}
```


3.10 *GetOhmPilotRealtimeData* request

This request provides detailed information about OhmPilot. Inactive channels are not included in the response and may vary depended on used hardware and software version. Take care about permanently or temporary missing channels when processing this response.

3.10.1 Availability

Platform	Since version
Fronius Hybrid	1.6.1-4
Fronius Non Hybrid	3.8.1-4
Fronius Generation 24	ALWAYS

☞ API is available but always return an error on Generation 24

3.10.2 URL for HTTP requests

`/solar_api/v1/GetOhmPilotRealtimeData.cgi`

3.10.3 Parameters

Parameter	Type	Range/Values/Pattern	Description
Scope	String	”System” ”Device”	Mandatory
DeviceId	String	0..65535	Mandatory on non system scope

3.10.4 Reference to manual

`https://www.fronius.com/~/downloads/Solar%20Energy/Operating%20Instructions/42%2C0410%2C2141%2CEN.pdf`

3.10.5 System-Request

Listing 51: Object structure of response body for GetOhmPilotRealtimeData System request

```
# object with detailed informations about all OhmPilots,
object {

    object {

      object {
          # serial number of device
          string Serial;

          # e.g. "Ohmpilot"
          string Model;

          # e.g. "Fronius"
          string Manufacturer;

          # software version
          string Software;

          # hardware version
          string Hardware;
      } Details;

      # total consumed energy [Wh]
      number EnergyReal_WAC_Sum_Consumed;

      # CodeOfState Values:
      # 0 ... up and running
      # 1 ... keep minimum temperature
      # 2 ... legionella protection
      # 3 ... critical fault
      # 4 ... fault
      # 5 ... boost mode
      number CodeOfState;

      # refer to OhmPilot manual
      # optional field
      number CodeOfError;

      # actual power consumption [W]
      number PowerReal_PAC_Sum;

      # temperature from sensor [°C]
      number Temperature_Channel_1;

    } * OhmPilot;

} Data ;
```

Listing 52: Reply body for GetOhmPilotRealtimeData System request

```
{
   "Body" : {
      "Data" : {
         "0" : {
            "CodeOfError" : 926,
            "CodeOfState" : 0,
            "Details" : {
               "Hardware" : "3",
               "Manufacturer" : "Fronius",
               "Model" : "Ohmpilot",
               "Serial" : "28136344",
               "Software" : "1.0.19-1"
            },
```

```
            "EnergyReal_WAC_Sum_Consumed" : 2964307,
            "PowerReal_PAC_Sum" : 0,
            "Temperature_Channel_1" : 23.899999999999999
         }
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "OhmPilot",
         "Scope" : "System"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-24T10:10:44+02:00"
   }
}
```

3.10.6 Device-Request

Listing 53: Object structure of response body for GetOhmPilotRealtimeData Device request

```
# object with detailed informations about one OhmPilot,
object {

   object {
          # serial number of device
          string Serial;

          # e.g. "Ohmpilot"
          string Model;

          # e.g. "Fronius"
          string Manufacturer;

          # software version
          string Software;

          # hardware version
          string Hardware;
   } Details;

   # total consumed energy [Wh]
   number EnergyReal_WAC_Sum_Consumed;

   # CodeOfState Values:
   # 0 ... up and running
   # 1 ... keep minimum temperature
   # 2 ... legionella protection
   # 3 ... critical fault
   # 4 ... fault
   # 5 ... boost mode
   number CodeOfState;

   # refer to OhmPilot manual
   # optional field
   number CodeOfError;

   # actual power consumption [W]
   number PowerReal_PAC_Sum;

   # temperature from sensor [°C]
   number Temperature_Channel_1;
```

```
} Data ;
```

Listing 54: Reply body for GetOhmPilotRealtimeData Device request


```
{
   "Body" : {
      "Data" : {
         "CodeOfError" : 926,
         "CodeOfState" : 0,
         "Details" : {
            "Hardware" : "3",
            "Manufacturer" : "Fronius",
            "Model" : "Ohmpilot",
            "Serial" : "28136344",
            "Software" : "1.0.19-1"
         },
         "EnergyReal_WAC_Sum_Consumed" : 2964307,
         "PowerReal_PAC_Sum" : 0,
         "Temperature_Channel_1" : 23.899999999999999
      }
   },
   "Head" : {
      "RequestArguments" : {
         "DeviceClass" : "OhmPilot",
         "DeviceId" : "0",
         "Scope" : "Device"
      },
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-24T10:10:41+02:00"
   }
}
```


3.11 *GetPowerFlowRealtimeData* request

This request provides detailed information about the local energy grid. The values replied represent the current state. Because of data has multiple asynchrone origins it is a matter of facts that the sum of all powers (grid, load and generate) will differ from zero.
This request does not care about the configured visibility of single inverters. All inverters are reported. Same for batteries.

3.11.1 Availability

Platform	Since version
Fronius Hybrid	1.2.1-X
Fronius Non Hybrid	3.3.9-X
Fronius Generation 24	ALWAYS

3.11.2 Version

This request is only a gateway to internal generated data containers. Please take care about the ”Version” field in the response.

Version	Changes
10	added smartloads/ohmpilot added Version field
11	added secondary meters for subloads or extra production
12	inverter nodes now provide component id

3.11.3 URL for HTTP requests

`/solar_api/v1/GetPowerFlowRealtimeData.fcgi`

Please note, for performance reasons the URL extension is different to other solar api requests.

3.11.4 Parameters

There are no parameters. Only one type of query exists.

3.11.5 Request

Listing 55: Object structure of response body for GetPowerFlowRealtimeData request

```
object {

    # mandatory field
    # implemented since Fronius Non Hybrid version 3.8.1-1
    #                   Fronius Hybrid version     1.6.1-1
    # describes the available fields for this request (PowerFlowVersion)
    unsigned integer Version;

    object {

        # mandatory field
        # Mode:                                 Contains:
        #   "produce-only",                     inverter only
        #   "meter", "vague-meter",             inverter and meter
        #   "bidirectional" or "ac-coupled"     inverter, meter and battery
        string Mode;

        # optional field, supported since Fronius Hybrid version 1.4.1-6
        #                 not available on Fronius Non Hybrid
        # true when battery is in standby
        boolean BatteryStandby;

        # optional field, supported since Fronius Hybrid version 1.3.1-0
        #                 not available on Fronius Non Hybrid
        # field is available if configured (false) or active (true)
        # if not available, mandatory config is not set
        boolean BackupMode;

        # mandatory field
        #this value is null if no meter is enabled ( + from grid, - to grid )
        number P_Grid;

        # mandatory field
        #this value is null if no meter is enabled ( + generator, - consumer )
        number P_Load;

        # mandatory field
        #this value is null if no battery is active ( - charge, + discharge )
        number P_Akku;

        # mandatory field
        #this value is null if inverter is not running  ( + production ( default ) )
        number P_PV;

        # mandatory field
        # available since Fronius Hybrid version 1.3.1-1
        # available since Fronius Non Hybrid version 3.7.1-2
        # current relative self consumption in %, null if no smart meter is connected
        number rel_SelfConsumption;

        # mandatory field
        # available since Fronius Hybrid version 1.3.1-1
```

```
        # available since Fronius Non Hybrid version 3.7.1-2
        # current relative autonomy in %, null if no smart meter is connected
        number rel_Autonomy;

        # optional field
        # "load", "grid" or "unknown" (during backup power)
        string Meter_Location;

        # optional field
        # implemented since Fronius Non Hybrid version 3.4.1-7
        # AC Energy [Wh] this day, null if no inverter is connected
        number E_Day;

        # optional field
        # implemented since Fronius Non Hybrid version 3.4.1-7
        # AC Energy [Wh] this year, null if no inverter is connected
        number E_Year;

        # optional field
        # implemented since Fronius Non Hybrid version 3.4.1-7
        # AC Energy [Wh] ever since, null if no inverter is connected
        number E_Total;

    } Site;

    object {

        object {

            # mandatory field
            # device type of inverter
            integer DT;

            # mandatory field
            # current power in Watt, null if not running (+ produce/export, - consume/
                import)
            integer P;

            # optional field
            # current state of charge in % as decimal ( 5.3% ) or integer (0 - 100%)
            unsigned number SOC;

            # mandatory field
            # implemented since Fronius Non Hybrid version 3.13.1-1
            #                   Fronius Hybrid version     1.11.1-1
            #                   PowerFlowVersion 12
            # component identification (8bit group, 16 bit enum)
            unsigned integer CID;


            # optional field
            #  "disabled", "normal", "service", "charge boost",
            #  "nearly depleted", "suspended", "calibrate",
            #  "grid support", "deplete recovery", "non operable (voltage)",
            #  "non operable (temperature)", "preheating", "startup",
            #  "stopped (temperature)", "battery full"
            string Battery_Mode;

            # optional field
            # implemented since Fronius Non Hybrid version 3.7.1-1
            #                   Fronius Hybrid version     1.3.1-1
            # AC Energy [Wh] this day, null if no inverter is connected
            number E_Day;

            # optional field
            # implemented since Fronius Non Hybrid version 3.7.1-1
            #                   Fronius Hybrid version     1.3.1-1
```

```
            # AC Energy [Wh] this year, null if no inverter is connected
            number E_Year;

            # optional field
            # implemented since Fronius Non Hybrid version 3.7.1-1
            #                   Fronius Hybrid version     1.3.1-1
            # AC Energy [Wh] ever since, null if no inverter is connected
            number E_Total;

        } * DeviceId; #SolarNet ring address ("1" on hybrid systems)

    } Inverters;

    # optional field
    # implemented since Fronius Non Hybrid version 3.8.1-1
    #                   Fronius Hybrid version     1.6.1-1
    #                   PowerFlowVersion 10
    object {

        # optional field
        # implemented since PowerFlowVersion 10
        object {

            # optional field
            # implemented since PowerFlowVersion 10
            object {

                # mandatory field
                # implemented since PowerFlowVersion 10
                # current power consumption in Watt
                number P_AC_Total;

                # mandatory field
                # implemented since PowerFlowVersion 10
                # operating state "normal", "min-temperature", "legionella-protection",
                #  "fault", "warning" or "boost"
                string State;

                # mandatory field
                # implemented since PowerFlowVersion 10
                # temperature of storage / tank in degree Celsius
                number Temperature;

            } ComponentId;

        } Ohmpilots;

    } Smartloads;

    # optional field
    # implemented since Fronius Non Hybrid version 3.12.1-1
    #                   Fronius Hybrid version     1.10.1-1
    #                   PowerFlowVersion 11
    object {

        # implemented since PowerFlowVersion 11
        object {

        # mandatory field
        # implemented since PowerFlowVersion 11
        # current power consumption/production in Watt (direction is based on meter
            location)
        #   consumption is negative for meter location >= 256
        #   production is positive for meter location 3
        number P;

        # mandatory field
```

```
        # implemented since PowerFlowVersion 11
        # meter location of the device (see end of listing for more details)
        number MLoc;

        # mandatory field
        # implemented since PowerFlowVersion 11
        # user defined name of secondary meter or
        # "<primary>" for primary meters
        string Label;

        # mandatory field
        # implemented since PowerFlowVersion 11
        # category token
        #  "METER_CAT_WR"            ... Photovoltaic inverter
        #  "METER_CAT_BAT"           ... AC storage unit
        #  "METER_CAT_PV_BAT"        ... Photovoltaic inverter + storage unit
        #  "METER_CAT_WINDMILL"      ... Wind turbine
        #  "METER_CAT_BHKW"          ... Combined heat and power station (CHP)
        #  "METER_CAT_ECAR"          ... Electric vehicle
        #  "METER_CAT_HEATPUMP"      ... Heatpump
        #  "METER_CAT_OTHERHEATING"  ... Other heating system
        #  "METER_CAT_PUMP"          ... Pump
        #  "METER_CAT_WHITEGOODS"    ... White goods
        #  "METER_CAT_CLIMATE"       ... Climate control / cooling systems
        #  "METER_CAT_BUILDING"      ... Building services
        #  "METER_CAT_OTHER"         ... Other
        string Category;

        } ComponentId;

    } SecondaryMeters;

} Data ;
```

Reference to device type table in section 5.2.
Reference to meter location table in section 5.5.

Listing 56: Reply body for GetPowerFlowRealtimeData on Fronius Hybrid System


```
{
   "Body" : {
      "Data" : {
         "Inverters" : {
            "1" : {
               "Battery_Mode" : "normal",
               "DT" : 99,
               "E_Day" : 6758,
               "E_Total" : 7604385.5,
               "E_Year" : 1342638.25,
               "P" : 506,
               "SOC" : 55
            }
         },
         "Site" : {
            "BatteryStandby" : false,
            "E_Day" : 6758,
            "E_Total" : 7604385.5,
            "E_Year" : 1342638.2000000002,
            "Meter_Location" : "grid",
            "Mode" : "bidirectional",
            "P_Akku" : -384.70000000000005,
            "P_Grid" : -511.99000000000001,
            "P_Load" : 5.9900000000000091,
            "P_PV" : 941.60000000000002,
            "rel_Autonomy" : 100,
            "rel_SelfConsumption" : 0
         },
```

```
         "Smartloads" : {
            "Ohmpilots" : {
               "720897" : {
                  "P_AC_Total" : 2635,
                  "State" : "normal",
                  "Temperature" : 30.7
               }
            }
         },
         "Version" : "12"
      }
   },
   "Head" : {
      "RequestArguments" : {},
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-12T15:31:07+02:00"
   }
}
```


Listing 57: Reply body for GetPowerFlowRealtimeData on Fronius Non Hybrid System


```
{
   "Body" : {
      "Data" : {
         "Inverters" : {
            "1" : {
               "DT" : 102,
               "E_Day" : 1393.2000732421875,
               "E_Total" : 1734796.125,
               "E_Year" : 322593.5,
               "P" : 88
            },
            "2" : {
               "DT" : 86,
               "E_Day" : 1618.5,
               "E_Total" : 3026782,
               "E_Year" : 385172.09375,
               "P" : 104
            },
            "3" : {
               "DT" : 106,
               "E_Day" : 1695.800048828125,
               "E_Total" : 3160499.75,
               "E_Year" : 399904.09375,
               "P" : 109
            },
            "55" : {
               "DT" : 224,
               "E_Day" : 1699,
               "E_Total" : 3275219.75,
               "E_Year" : 403993.21875,
               "P" : 109
            }
         },
         "Site" : {
            "E_Day" : 6406.5001220703125,
            "E_Total" : 11197297.625,
            "E_Year" : 1511662.90625,
            "Meter_Location" : "unknown",
            "Mode" : "produce-only",
            "P_Akku" : null,
            "P_Grid" : null,
            "P_Load" : null,
```

```
            "P_PV" : 410,
            "rel_Autonomy" : null,
            "rel_SelfConsumption" : null
         },
         "Version" : "12"
      }
   },
   "Head" : {
      "RequestArguments" : {},
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-06-12T15:31:08+02:00"
   }
}
```


Listing 58: Reply body for GetPowerFlowRealtimeData on Generation 24 Primo


```
{
   "Body" : {
      "Data" : {
         "Inverters" : {
            "1" : {
               "Battery_Mode" : "normal",
               "DT" : 1,
               "P" : 501,
               "SOC" : 30.600000381469727
            }
         },
         "Site" : {
            "BatteryStandby" : false,
            "E_Day" : null,
            "E_Total" : null,
            "E_Year" : null,
            "Meter_Location" : "grid",
            "Mode" : "bidirectional",
            "P_Akku" : -1006.1868286132812,
            "P_Grid" : -497.62,
            "P_Load" : -3.3799999999999955,
            "P_PV" : 1547.739990234375,
            "rel_Autonomy" : 100.0,
            "rel_SelfConsumption" : 0.67465069860279347
         },
         "Version" : "12"
      }
   },
   "Head" : {
      "RequestArguments" : {},
      "Status" : {
         "Code" : 0,
         "Reason" : "",
         "UserMessage" : ""
      },
      "Timestamp" : "2019-08-28T09:43:29+00:00"
   }
}
```


☞ Energies are not provided and device types DT are invalid on Generation 24

4 Archive Requests

4.1 Common

Archive requests shall be provided whenever access to historic device-data is possible and it makes sense to provide such a request.
Of course, the Datalogger Web can only provide what is stored in its internal memory and has not been overwritten by newer data yet. It can loose data, due to capacity reason. The number of days stored dependence on the number of connected units to log. This limitation of is not present for Solar.web, provided that the Datalogger has reliably uploaded the data.

Different from what is specified within the previously released drafts, there is only one CGI to access all historic data. This CGI contains detailed, summed, error and events queries.

Call is *http://<insert hostname or IP here>/solar_api/v1/GetArchiveData.cgi?<your query parameters>*

The number of parallel queries is system wide restricted to 4 clients.

4.1.1 Availability

Platform	Since version
Fronius Hybrid	1.1.2-16
Fronius Non Hybrid	3.3.4-5
Fronius Generation 24	NEVER

4.1.2 ChannelId

Each channel is handled and requested by name. Most of the channels are recorded in constant cyclic intervals which can be set between 5 and 30 minutes. Only $Digital\_PowerManagementRelay\_Out\_*$, $InverterErrors$, $InverterEvents$ and $Hybrid\_Operating\_State$ are event triggered and may occure every time.
[4]

[4] introduced in SolarAPI CompatibilityRange Version 1.5-10 (Datamanager 3.11.1 or Hybridmanager 1.9.1)

Table 5: Available channels

Name	Unit
TimeSpanInSec	sec
EnergyReal_WAC_Sum_Produced	Wh
EnergyReal_WAC_Sum_Consumed [4]	Wh
InverterEvents	struct
InverterErrors	struct
Current_DC_String_1	1A
Current_DC_String_2	1A
Voltage_DC_String_1	1V
Voltage_DC_String_2	1V
Temperature_Powerstage	deg C
Voltage_AC_Phase_1	1V
Voltage_AC_Phase_2	1V
Voltage_AC_Phase_3	1V
Current_AC_Phase_1	1A
Current_AC_Phase_2	1A
Current_AC_Phase_3	1A
PowerReal_PAC_Sum	1W
EnergyReal_WAC_Minus_Absolute	1Wh
EnergyReal_WAC_Plus_Absolute	1Wh
Meter_Location_Current	1
Temperature_Channel_1	1
Temperature_Channel_2	1
Digital_Channel_1	1
Digital_Channel_2	1
Radiation	1
Digital_PowerManagementRelay_Out_1	1
Digital_PowerManagementRelay_Out_2	1
Digital_PowerManagementRelay_Out_3	1
Digital_PowerManagementRelay_Out_4	1
Hybrid_Operating_State	1

4.1.3 Parameters

Scope	String	”Device” ”System”	Query specific device(s) or whole system. *Mandatory*
SeriesType	String	”DailySum” ”Detail” (default)	Resolution of the data-series. *Optional*
HumanReadable	BoolString	”True” (default) ”False”	Unset/Set readable output. *Optional*
StartDate	DateString	”21.5.[20]14” ”5/21/[20]14” ”[20]14-5-21” ”2011-10-20T10:09:14+02:00”	*Mandatory* supplying only the date will be interpreted as local time
EndDate	DateString	”21.5.[20]14” ”5/21/[20]14” ”[20]14-5-21” ”2011-10-20T10:09:14Z”	*Mandatory*
Channel	String	available channels from table 5. *Mandatory, multiple times*	
DeviceClass	String	”Inverter” ”SensorCard” ”StringControl”	Which kind of device will be queried. *Mandatory and accepted only if Scope is not ”System”*
DeviceClass		”Meter” ”Storage” ”OhmPilot”	since DM 3.7.4-6 HM 1.3.1-1 since DM 3.7.4-6 HM 1.3.1-1 since DM 3.8.1-4 HM 1.6.1-4
DeviceId	String	*Solar Net:* 0 …199	*Only needed for Scope ”Device”* Which device to query. This parameter can be given more than once, thus specifying a list of devices to query.

4.1.4 Object Structure of response body

Listing 59: Object structure of request body

```
object {

  # Object with dataseries for each requested device.
  # Property names correspond to the DeviceId the series belongs to.
  object {

    # Object representing data-series of one device (may contain more than one channel).
    object {

      # Optional Nodetype if localnet node
      integer NodeType;

      # Optional Devicetype if localnet node
      integer DeviceType;

      # Starting date of the series (i.e. date of the first value in Values)
      # yyyy-MM-ddThh-mm-ss%z   "2017-05-20T00:00:00+02:00"
      string Start;

      # Starting date of the series (i.e. date of the first value in Values)
      # yyyy-MM-ddThh-mm-ss%z   "2017-05-20T23:59:59+02:00"
      string End;


      # Collection of objects representing one channel, each object containing values and
          metadata.
      # Objects are named after the Channel they represent (e.g. "Power").
      object {

        # Object representing one channel.
```

```
        object {
          # Baseunit of the channel, never contains any prefixes
          string Unit;

          # Unscaled values, offset between datapoints can be deduced through "SeriesType"
          # ATTENTION: Unavailable datapoints are included but have value null
          # NOTE: the data records are listed in alphabetical order
          # example: "3600" : 10.11 .... offset is 3600 sec and value is 10.11
          number * __OFFSET_IN_SECONDS__;

          # reference to internal used unique channel identifier
          string _comment;

        } __CHANNEL_NAME__;

      }* Data;

    }* __DEVICE_ID__;

  } Data;

};
```

4.2 Example of response body

4.2.1 Meter data

Listing 60: detailed response body for meter data

```
// /solar_api/v1/GetArchiveData.cgi?Scope=System&StartDate=1.3.2018&EndDate=1.3.2018&
    Channel=TimeSpanInSec&Channel=EnergyReal_WAC_Plus_Absolute&Channel=
    EnergyReal_WAC_Minus_Absolute&Channel=Meter_Location_Current
{
  "Body" :
  {
    "Data" :
    {
      "inverter/1" :
      {
        "Data" :
        {
          "TimeSpanInSec" :
          {
            "Unit" : "sec",
            "Values" :
            {
              "0" : 298,                /* 0 seconds after "Start" the value was 298 */
              "10200" : 299,
              /* shorten list for readability */
              "1200" : 300,
              /* shorten list for readability */
              "12000" : 300,
              /* shorten list for readability */
              "2100" : 300,
              /* shorten list for readability */
              "900" : 299,
              "9000" : 300,
              "9300" : 299,
              "9600" : 299,
              "9900" : 300
            },
            "_comment" : "channelId=65549"
          }
        },
        "DeviceType" : 99,
        "End" : "2018-03-01T23:59:59+01:00",
```

```
      "NodeType" : 97,
      "Start" : "2018-03-01T00:00:00+01:00"
    },
    "meter:15480258" :
    {
      "Data" :
      {
        "EnergyReal_WAC_Minus_Absolute" :
        {
          "Unit" : "Wh",
          "Values" :
          {
            "0" : 744657,
            "10200" : 744657,
            "10500" : 744657,
            /* shorten list for readability */
            "9300" : 744657,
            "9600" : 744657,
            "9900" : 744657
          },
          "_comment" : "channelId=167837960"
        },
        "EnergyReal_WAC_Plus_Absolute" :
        {
          "Unit" : "Wh",
          "Values" :
          {
            "0" : 605047,
            "10200" : 605194,
            "10500" : 605198,
            "10800" : 605202,
            /* shorten list for readability */
            "9000" : 605177,
            "9300" : 605181,
            "9600" : 605185,
            "9900" : 605190
          },
          "_comment" : "channelId=167772424"
        },
        "Meter_Location_Current" :
        {
          "Unit" : "1",
          "Values" :
          {
            "0" : 0,
            "10200" : 0,
            "10500" : 0,
            "10800" : 0,
            /* shorten list for readability */
            "9600" : 0,
            "9900" : 0
          },
          "_comment" : "channelId=117050390"
        }
      },
      "End" : "2018-03-01T23:59:59+01:00",
      "Start" : "2018-03-01T00:00:00+01:00"
    }
  }
},
"Head" :
{
  "RequestArguments" :
  {
    "Channel" :
    [
      "TimeSpanInSec",
```

```
        "EnergyReal_WAC_Plus_Absolute",
        "EnergyReal_WAC_Minus_Absolute",
        "Meter_Location_Current"
      ],
      "EndDate" : "2018-03-01T23:59:59+01:00",
      "HumanReadable" : "True",
      "Scope" : "System",
      "SeriesType" : "Detail",
      "StartDate" : "2018-03-01T00:00:00+01:00"
    },
    "Status" :
    {
      "Code" : 0,
      "ErrorDetail" :
      {
        "Nodes" : []
      },
      "Reason" : "",
      "UserMessage" : ""
    },
    "Timestamp" : "2018-03-02T07:57:54+01:00"
  }
}
```

4.2.2 Inverter data

Listing 61: detailed response body with multiple inverters

```
// /solar_api/v1/GetArchiveData.cgi?Scope=System&StartDate=1.3.2018&EndDate=1.3.2018&
    Channel=EnergyReal_WAC_Sum_Produced&Channel=EnergyReal_WAC_Sum_Consumed

{
  "Body" :
  {
    "Data" :
    {
      "inverter/24" :
      {
        "Data" :
        {
          "EnergyReal_WAC_Sum_Produced" :
          {
            "Unit" : "Wh",
            "Values" :
            {
              "39900" : 457831.95472222223,
              "40200" : 316.86027777777775,
              "40500" : 350.18166666666667,
              "40800" : 357.11305555555555,
              "41100" : 330.60611111111109,
              /* shorten list for readability */
              "85800" : 0,
              "86100" : 0
            },
            "_comment" : "channelId=67830024"
          }
        },
        "DeviceType" : 192,
        "End" : "2018-03-01T23:59:59+01:00",
        "NodeType" : 120,
        "Start" : "2018-03-01T00:00:00+01:00"
      },
      "inverter/25" :
      {
        "Data" :
```

```
        {
          "EnergyReal_WAC_Sum_Produced" :
          {
            "Unit" : "Wh",
            "Values" :
            {
              "39900" : 319.23555555555555,
              /* shorten list for readability */
              "85200" : 0,
              "85500" : 0,
              "85800" : 0,
              "86100" : 0
            },
            "_comment" : "channelId=67830024"
          }
        },
        "DeviceType" : 192,
        "End" : "2018-03-01T23:59:59+01:00",
        "NodeType" : 121,
        "Start" : "2018-03-01T00:00:00+01:00"
      }
    }
  },
  "Head" :
  {
    "RequestArguments" :
    {
      "Channel" :
      [
        "EnergyReal_WAC_Sum_Produced",
        "EnergyReal_WAC_Sum_Consumed"
      ],
      "EndDate" : "2018-03-01T23:59:59+01:00",
      "HumanReadable" : "True",
      "Scope" : "System",
      "SeriesType" : "Detail",
      "StartDate" : "2018-03-01T00:00:00+01:00"
    },
    "Status" :
    {
      "Code" : 0,
      "ErrorDetail" :
      {
        "Nodes" : []
      },
      "Reason" : "",
      "UserMessage" : ""
    },
    "Timestamp" : "2018-03-02T09:49:51+01:00"
  }
}
```

4.2.3 Errors - Structure

Listing 62: Example of response body for inverter errors

```
// /solar_api/v1/GetArchiveData.cgi?Scope=System&StartDate=1.3.2018&EndDate=2.3.2018&
    Channel=InverterErrors

{
  "Body": {
    "Data": {
      "inverter/1": {
        "Data": {
          "InverterErrors": {
            "Unit": "Object",
```

```
            "Values": {                         /* alphabetic sorted list of time offsets */
              "123180" : {                              /* 123180 seconds after "Start" */
                "flags" : [ "fatal","official" ],
                "#" : 731                               /* Error Code 731 */
              },
              "123240" : {
                "flags" : [ "fatal", "official"],
                "#" : 766
              },
              "23240" : {
                "flags" : [ "fatal", "official"],
                "#" : 482
              }
            },
            "_comment": "channelId=16646144"
          }
        },
        "DeviceType": 99,
        "End": "2018-03-02T23:59:59+01:00",
        "NodeType": 97,
        "Start": "2018-03-01T00:00:00+01:00"
      }
    }
  },
  "Head": {
    "RequestArguments": {
      "Channel": [
        "InverterErrors"
      ],
      "EndDate": "2018-03-02T23:59:59+01:00",
      "HumanReadable": "True",
      "Scope": "System",
      "SeriesType": "Detail",
      "StartDate": "2018-03-01T00:00:00+01:00"
    },
    "Status": {
      "Code": 0,
      "ErrorDetail": {
        "Nodes": []
      },
      "Reason": "",
      "UserMessage": ""
    },
    "Timestamp": "2018-03-02T11:32:22+01:00"
  }
}
```

4.2.4 Events - Structure

Listing 63: Example of response body for inverter events

```
// /solar_api/v1/GetArchiveData.cgi?Scope=System&StartDate=2.3.2018&EndDate=2.3.2018&
    Channel=InverterEvents

{
  "Body" :
  {
    "Data" :
    {
      "broadcast/" :
      {
        "Data" :
        {
          "InverterEvents" :
          {
            "Unit" : "Object",
```

```
            "Values" :
            {
              "42060" :                                 /* seconds after "Start" */
              {
                "#" : 3,                                /* Event Code 3 */
                "attr" :                                /* Event Specific Data */
                {
                  "Power" : "20␣[%]",
                  "Radient" : "255␣[1]",
                  "affect" : "P"
                },
                "desc" : "Power␣limitation␣20%",        /* Event Description */
                "flags" :
                [
                  "send"
                ]
              }
            },
            "_comment" : "channelId=16711680"
          }
        },
        "End" : "2018-03-02T23:59:59+01:00",
        "Start" : "2018-03-02T00:00:00+01:00"
      }
    }
  },
  "Head" :
  {
    "RequestArguments" :
    {
      "Channel" :
      [
        "InverterEvents"
      ],
      "EndDate" : "2018-03-02T23:59:59+01:00",
      "HumanReadable" : "True",
      "Scope" : "System",
      "SeriesType" : "Detail",
      "StartDate" : "2018-03-02T00:00:00+01:00"
    },
    "Status" :
    {
      "Code" : 0,
      "ErrorDetail" :
      {
        "Nodes" : []
      },
      "Reason" : "",
      "UserMessage" : ""
    },
    "Timestamp" : "2018-03-02T11:42:50+01:00"
  }
}
```

4.2.5 OhmPilot Energy

OhmPilot uses total energy counter!

Listing 64: detailed response body for one OhmPilot

```
// /solar_api/v1/GetArchiveData.cgi?Scope=Device&DeviceClass=OhmPilot&DeviceId=0&StartDate
    =6.3.2018&EndDate=6.3.2018&Channel=EnergyReal_WAC_Sum_Consumed

{
  "Body" :
  {
```

```
    "Data" :
    {
      "ohmpilot:28136344" :
      {
        "Data" :
        {
          "EnergyReal_WAC_Sum_Consumed" :
          {
            "Unit" : "Wh",
            "Values" :
            {
              "0" : 858547,
              "10200" : 858547,
              "10500" : 858547,
              "10800" : 858547,
              "11100" : 858547,
              "11400" : 858547,
              "11700" : 858547,
              "1200" : 858547,
              "12000" : 858547,
              /* shorten list for readability */
              "84000" : 867084,
              "84300" : 867085,
              "84600" : 867087,
              "84900" : 867089,
              "85200" : 867091,
              "85500" : 867093,
              "85800" : 867095,
              "86100" : 867097,
              "8700" : 858547,
              "900" : 858547,
              "9000" : 858547,
              "9300" : 858547,
              "9600" : 858547,
              "9900" : 858547
            },
            "_comment" : "channelId=67895560"
          }
        },
        "End" : "2018-03-06T23:59:59+01:00",
        "Start" : "2018-03-06T00:00:00+01:00"
      }
    }
  },
  "Head" :
  {
    "RequestArguments" :
    {
      "Channel" :
      [
        "EnergyReal_WAC_Sum_Consumed"
      ],
      "DeviceClass" : "OhmPilot",
      "DeviceId" : "0",
      "EndDate" : "2018-03-06T23:59:59+01:00",
      "HumanReadable" : "True",
      "Scope" : "Device",
      "SeriesType" : "Detail",
      "StartDate" : "2018-03-06T00:00:00+01:00"
    },
    "Status" :
    {
      "Code" : 0,
      "ErrorDetail" :
      {
        "Nodes" : []
      },
```

```
      "Reason" : "",
      "UserMessage" : ""
    },
    "Timestamp" : "2018-03-07T10:28:39+01:00"
  }
}
```


5 Definitions and Mappings

5.1 Sunspec State Mapping

Table 6: Shows mapping between Fronius device status and SunSpec Inverter-Model states

Fronius device state	SunSpec device state
not used	I_STATUS_OFF
not used	I_STATUS_SLEEPING
not used	I_STATUS_THROTTLED
not used	I_STATUS_SHUTTING_DOWN
10	I_STATUS_FAULT
8	I_STATUS_STANDBY
7	I_STATUS_MPPT
others	I_STATUS_STARTING

5.2 Inverter Device Type List

DeviceType	Model Name
54	Fronius Primo Hybrid 5.0-1 240
55	Fronius Primo Hybrid 6.0-1 240
56	Fronius Primo Hybrid 8.0-1 240
57	Fronius Primo Hybrid 10.0-1 240
58	Fronius Primo Hybrid 3.6-1
59	Fronius Primo Hybrid 4.0-1
60	Fronius Primo Hybrid 4.6-1
61	Fronius Primo Hybrid 5.0-1
62	Fronius Primo Hybrid 6.0-1
63	Fronius Primo Hybrid 8.0-1
64	Fronius Primo Hybrid 11.4-1
65	Fronius Primo Hybrid 10.0-1
66	Fronius Primo Hybrid 11.4-1 240
67	Fronius Primo 15.0-1 208-240
68	Fronius Primo 12.5-1 208-240
69	Fronius Primo 11.4-1 208-240
70	Fronius Primo 10.0-1 208-240
71	Fronius Symo 15.0-3 208
72	Fronius Eco 27.0-3-S
73	Fronius Eco 25.0-3-S
75	Fronius Primo 6.0-1
76	Fronius Primo 5.0-1
77	Fronius Primo 4.6-1
78	Fronius Primo 4.0-1
79	Fronius Primo 3.6-1
80	Fronius Primo 3.5-1
81	Fronius Primo 3.0-1
82	Fronius Symo Hybrid 4.0-3-S
83	Fronius Symo Hybrid 3.0-3-S

84	Fronius IG Plus 120 V-1
85	Fronius Primo 3.8-1 208-240
86	Fronius Primo 5.0-1 208-240
87	Fronius Primo 6.0-1 208-240
88	Fronius Primo 7.6-1 208-240
89	Fronius Symo 24.0-3 USA Dummy
90	Fronius Symo 24.0-3 480
91	Fronius Symo 22.7-3 480
92	Fronius Symo 20.0-3 480
93	Fronius Symo 17.5-3 480
94	Fronius Symo 15.0-3 480
95	Fronius Symo 12.5-3 480
96	Fronius Symo 10.0-3 480
97	Fronius Symo 12.0-3 208-240
98	Fronius Symo 10.0-3 208-240
99	Fronius Symo Hybrid 5.0-3-S
100	Fronius Primo 8.2-1 Dummy
101	Fronius Primo 8.2-1 208-240
102	Fronius Primo 8.2-1
103	Fronius Agilo TL 360.0-3
104	Fronius Agilo TL 460.0-3
105	Fronius Symo 7.0-3-M
106	Fronius Galvo 3.1-1 208-240
107	Fronius Galvo 2.5-1 208-240
108	Fronius Galvo 2.0-1 208-240
109	Fronius Galvo 1.5-1 208-240
110	Fronius Symo 6.0-3-M
111	Fronius Symo 4.5-3-M
112	Fronius Symo 3.7-3-M
113	Fronius Symo 3.0-3-M
114	Fronius Symo 17.5-3-M
115	Fronius Symo 15.0-3-M
116	Fronius Agilo 75.0-3 Outdoor
117	Fronius Agilo 100.0-3 Outdoor
118	Fronius IG Plus 55 V-1
119	Fronius IG Plus 55 V-2
120	Fronius Symo 20.0-3 Dummy
121	Fronius Symo 20.0-3-M
122	Fronius Symo 5.0-3-M
123	Fronius Symo 8.2-3-M
124	Fronius Symo 6.7-3-M
125	Fronius Symo 5.5-3-M
126	Fronius Symo 4.5-3-S
127	Fronius Symo 3.7-3-S
128	Fronius IG Plus 60 V-2
129	Fronius IG Plus 60 V-1
130	SPR 8001F-3 EU
131	Fronius IG Plus 25 V-1
132	Fronius IG Plus 100 V-3
133	Fronius Agilo 100.0-3
134	SPR 3001F-1 EU
135	Fronius IG Plus V/A 10.0-3 Delta
136	Fronius IG 50
137	Fronius IG Plus 30 V-1
138	SPR-11401f-1 UNI
139	SPR-12001f-3 WYE277
140	SPR-11401f-3 Delta
141	SPR-10001f-1 UNI
142	SPR-7501f-1 UNI

143	SPR-6501f-1 UNI
144	SPR-3801f-1 UNI
145	SPR-3301f-1 UNI
146	SPR 12001F-3 EU
147	SPR 10001F-3 EU
148	SPR 8001F-2 EU
149	SPR 6501F-2 EU
150	SPR 4001F-1 EU
151	SPR 3501F-1 EU
152	Fronius CL 60.0 WYE277 Dummy
153	Fronius CL 55.5 Delta Dummy
154	Fronius CL 60.0 Dummy
155	Fronius IG Plus V 12.0-3 Dummy
156	Fronius IG Plus V 7.5-1 Dummy
157	Fronius IG Plus V 3.8-1 Dummy
158	Fronius IG Plus 150 V-3 Dummy
159	Fronius IG Plus 100 V-2 Dummy
160	Fronius IG Plus 50 V-1 Dummy
161	Fronius IG Plus V/A 12.0-3 WYE
162	Fronius IG Plus V/A 11.4-3 Delta
163	Fronius IG Plus V/A 11.4-1 UNI
164	Fronius IG Plus V/A 10.0-1 UNI
165	Fronius IG Plus V/A 7.5-1 UNI
166	Fronius IG Plus V/A 6.0-1 UNI
167	Fronius IG Plus V/A 5.0-1 UNI
168	Fronius IG Plus V/A 3.8-1 UNI
169	Fronius IG Plus V/A 3.0-1 UNI
170	Fronius IG Plus 150 V-3
171	Fronius IG Plus 120 V-3
172	Fronius IG Plus 100 V-2
173	Fronius IG Plus 100 V-1
174	Fronius IG Plus 70 V-2
175	Fronius IG Plus 70 V-1
176	Fronius IG Plus 50 V-1
177	Fronius IG Plus 35 V-1
178	SPR 11400f-3 208/240
179	SPR 12000f-277
180	SPR 10000f
181	SPR 10000F EU
182	Fronius CL 33.3 Delta
183	Fronius CL 44.4 Delta
184	Fronius CL 55.5 Delta
185	Fronius CL 36.0 WYE277
186	Fronius CL 48.0 WYE277
187	Fronius CL 60.0 WYE277
188	Fronius CL 36.0
189	Fronius CL 48.0
190	Fronius IG TL 3.0
191	Fronius IG TL 4.0
192	Fronius IG TL 5.0
193	Fronius IG TL 3.6
194	Fronius IG TL Dummy
195	Fronius IG TL 4.6
196	SPR 12000F EU
197	SPR 8000F EU
198	SPR 6500F EU
199	SPR 4000F EU
200	SPR 3300F EU
201	Fronius CL 60.0

202	SPR 12000f
203	SPR 8000f
204	SPR 6500f
205	SPR 4000f
206	SPR 3300f
207	Fronius IG Plus 12.0-3 WYE277
208	Fronius IG Plus 50
209	Fronius IG Plus 100
210	Fronius IG Plus 100
211	Fronius IG Plus 150
212	Fronius IG Plus 35
213	Fronius IG Plus 70
214	Fronius IG Plus 70
215	Fronius IG Plus 120
216	Fronius IG Plus 3.0-1 UNI
217	Fronius IG Plus 3.8-1 UNI
218	Fronius IG Plus 5.0-1 UNI
219	Fronius IG Plus 6.0-1 UNI
220	Fronius IG Plus 7.5-1 UNI
221	Fronius IG Plus 10.0-1 UNI
222	Fronius IG Plus 11.4-1 UNI
223	Fronius IG Plus 11.4-3 Delta
224	Fronius Galvo 3.0-1
225	Fronius Galvo 2.5-1
226	Fronius Galvo 2.0-1
227	Fronius IG 4500-LV
228	Fronius Galvo 1.5-1
229	Fronius IG 2500-LV
230	Fronius Agilo 75.0-3
231	Fronius Agilo 100.0-3 Dummy
232	Fronius Symo 10.0-3-M
233	Fronius Symo 12.5-3-M
234	Fronius IG 5100
235	Fronius IG 4000
236	Fronius Symo 8.2-3-M Dummy
237	Fronius IG 3000
238	Fronius IG 2000
239	Fronius Galvo 3.1-1 Dummy
240	Fronius IG Plus 80 V-3
241	Fronius IG Plus 60 V-3
242	Fronius IG Plus 55 V-3
243	Fronius IG 60 ADV
244	Fronius IG 500
245	Fronius IG 400
246	Fronius IG 300
247	Fronius Symo 3.0-3-S
248	Fronius Galvo 3.1-1
249	Fronius IG 60 HV
250	Fronius IG 40
251	Fronius IG 30 Dummy
252	Fronius IG 30
253	Fronius IG 20
254	Fronius IG 15

5.3 Event Table for Fronius Devices

Event Code	Description
1	System offset
2	Calibrate factor
3	Power control commands
4	Gradual Voltage dependend Power Reduction
5	Frequency Limit Change
6	Enter Backup Power Mode
7	Leave Backup Power Mode
8	Critical SOC reached within backmode
9	Component Specific StateCode
10	Calibration Suspension enabled
11	Datamanager reboot due to malfunction

5.4 Hybrid_Operating_State

Hybrid_Operating_State	Description
0	disabled
1	normal
2	service
3	charge boost
4	nearly depleted
5	suspended
6	calibrate
7	grid support
8	deplete recovery
9	non operable (temperature)
10	non operable (voltage)
11	preheating
12	startup
13	until Hybrid 1.13.1: awake but non operable (temperature) since Hybrid 1.13.1: stopped (temperature)
14	battery full

5.5 Meter Locations

Meter Location	Description
0	Load
1	Grid
2	RESERVED
3	additional A.C. generator (generation only)
4	additional A.C. generator providing a battery (consumption and generation)
5-255	RESERVED
256-511	Sub Load

6 Changelog

Document Version 14

- PowerFlowRealtimeData battery state 14 ”battery full” added

Document Version 13

- description for Fronius Generation 24 added but be aware that this is only temporary provided. API will be removed!
- updated and added missing json examples
- added inverter device type list in section 5.2
- PowerFlowRealtimeData provides data of secondary meters
- added GetStorageRealtime example for LG-Chem and BYD B-Box

- NOTE: manufacturer changed for Solar Battery at GetStorageRealtimeData.cgi
- PowerFlowRealtimeData battery state 13 ”stopped (temperature)” added
- Inverter energy is AC related
- PowerFlowRealtimeData battery soc changed from non-decimal to decimal on demand (support both)
- PowerFlowRealtimeData introduced component identifier field CID
- added meter location table in section 5.5

Document Version 12

- never been published. Changes listed at version 13

Document Version 11

- NOTE: DefaultLanguage at GetLoggerInfo will be removed soon

Document Version 10

- note that all inverters (even invisible configured) are reportet at PowerFlow and Inverter request
- fixed description about availability of rel_Autonomy and rel_SelfConsumption at PowerFlow request
- fixed missing description of BatteryStandby at PowerFlow request
- improved and fixed GetArchiveData descriptions and examples

Document Version 9

- $Battery\_Mode$ at PowerFlowRealtimeData got more states
- fixed GetLoggerLEDInfo.cgi example
- added meter location state ”unknown” while backup power is active
- placed notification to use http-get request (refer to section 2.3)
- added Smartloads/OhmPilot node at PowerFlowRealtimeData.fcgi
- added description about PowerFlowRealtimeData versioning
- described $Status\_BatteryCell$ for Controller of Tesla at GetStorageRealtimeData.cgi
- added $Status\_Battery$ description for Tesla at GetStorageRealtimeData.cgi
- GetInverterRealtimeData PAC type changed from unsigned to signed integer
- added channel names for Sensor Card (refer to table 5)
- added description of field DeliveryFactor at GetLoggerInfo and updated example
- fixed description of GetInverterInfo: properties ’Show’ and ’CustomName’ have been mandatory since Version 3.0.3
- added GetOhmPilotRealtimeData.cgi
- added description of all possible Error Codes
- intodruced SolarAPI ”Compatibility Range” at GetAPIVersion.cgi
- fixed description of datatypes

cgi	Field	old description	fixed description
GetAPIVersion.cgi	APIVersion	number	unsigned integer
GetInverterInfo.cgi	Body.Data.<>... ...ErrorCode ...StatusCode Body.Data.<>.Show	number number	integer unsigned integer
GetInverterRealtimeData.cgi	Body.Data... ...DAY_ENERGY ...YEAR_ENERGY ...TOTAL_ENERGY	unsigned int	unsigned number

15th Septemper 2016

- fixed availability notes of GetInverterRealtimeData

- OhmPilot is listed too
- added battery status description
- added description for energies at GetPowerFlowRealtimeData

11th February 2016

- fixed availability of request GetPowerFlowRealtimeData

13th August 2015

- Added realtime request GetPowerFlowRealtimeData to api

10th July 2015

- Added realtime request GetStorageRealtimeData to api

1st June 2015

- Minor documentation update.

  GetLoggerLedInfo.cgi added ”alternating” led state (timeout of access point)
  GetArchiveData.cgi revised data queries and responses

7 Frequently asked questions

1. **The application I wrote for the Fronius Datalogger Web does not work with the Fronius Datamanager. Why is that?**
   This is because we had to make some changes in the API to ensure compatibility with future devices. Specifically the *DeviceIndex* parameter is now named *DeviceId* and the request URLs have been changed to include an API version. For further details please refer to the latest version of the API specs.

2. **Which data can I get?**
   Currently only realtime data from inverters, Fronius Sensor Cards and Fronius String Controls. Also some information about the logging device itself is available.
   Please refer to the API specs for further details.

3. **Can multiple clients send requests to the API at the same time?**
   Yes, but the requests may take longer to complete.

4. **Can I use this API at the same time as other services of the Datalogger Web / DataManager?**
   Yes. The datalogging, Solar.access/Solar.web connection, Webinterface, this API or any other service can be used independently from the others.

5. **Can the API calls be password protected?**
   No. The API is always accessible without authentication, regardless of the user or admin password set on the Webinterface.

6. **The API reports more inverters than I have, why is that?**
   This may be the case when the inverter number of an inverter is changed while the Fronius Datalogger Web / Fronius Datamanager is running. The logger then detects a new device but keeps the same device with the previous inverter number in the system for 24 hours. This is due to the fact that the datalogger is caching the devices for a certain time even if they are not present on the bus (e.g. to be able to display energy values during the night when the inverters are offline).
   Those ghost devices will disappear 24 hours after the have been last seen by the datalogger. Alternatively, a reboot of the datalogger also clears the device cache and repopulates it with the currently present devices.

Fronius Worldwide - www.fronius.com/addresses

Fronius International GmbH
4600 Wels, Froniusplatz 1, Austria
E-Mail: pv-sales@fronius.com
http://www.fronius.com

Fronius USA LLC Solar Electronics Division
6797 Fronius Drive, Portage, IN 46368
E-Mail: pv-us@fronius.com
http://www.fronius-usa.com

Under http://www.fronius.com/addresses you will find all addresses of our sales branches and partner firms!