SHIFTING THE LIMITS

Fronius Solar API V1

EN

Operating Instructions

System monitoring

42,0410,2012,EN 008-15092016

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.

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.

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 (markes 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.

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.

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.
    number APIVersion;

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

Listing 2: Example: Complete response for GetAPIVersion request

```
{
    "APIVersion" : 1,
    "BaseURL" : "/solar_api/v1/"
}
```

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


```
{
    "Head": {
        "RequestArguments": {
             "Scope": "Device",
             "DeviceId": "0",
             "DataCollection": "CommonInverterData"
         },
        "Status": {
            "Code": 0,
            "Reason": "",
            "UserMessage": ""
        },
        "Timestamp": "2011-10-20T10:09:14+02:00"
    },
    "Body": {
        "Data" : {
            "DAY_ENERGY" : {
                    "Value" : 8000,
                    "Unit" : "Wh"
            },
            "FAC" : {
                    "Value" : 50,
                    "Unit" : "Hz"
            },
            "IAC" : {
                    "Value" : 14.54,
                    "Unit" : "A"
            },
            "IDC" : {
                    "Value" : 8.2,
                    "Unit" : "A"
            },
            "PAC" : {
                    "Value" : 3373,
                    "Unit" : "W"
            },
            "SAC" : {
                    "Value" : 3413,
                    "Unit" : "VA"
            },
            "TOTAL_ENERGY" : {
                    "Value" : 45000,
                    "Unit" : "Wh"
            },
            "UAC" : {
                    "Value" : 232,
                    "Unit" : "V"
            },
            "UDC" : {
                    "Value" : 426,
                    "Unit" : "V"
            },
            "YEAR_ENERGY" : {
                    "Value" : 44000,
                    "Unit" : "Wh"
            },
            "DeviceStatus" : {
                    "DeviceState" : 7,
                    "MgmtTimerRemainingTime" : -1,
                    "ErrorCode" : 0,
                    "LEDColor" : 2,
                    "LEDState" : 0,
                    "StateToReset" : false
            }
        }
    }
```

```
}
```

2.4.1 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;

};
```

2.4.2 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

3.1.1 Availability

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

3.1.2 Collection availability on Fronius Hybrid Systems

DataCollection	supported
CumulationInverterData	Yes
CommonInverterData	Yes
3PInverterData	Yes
MinMaxInverterData	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
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	JSON type	Description
PAC	unsigned integer	AC power
DAY_ENERGY	unsigned integer	Energy generated on current day
YEAR_ENERGY	unsigned integer	Energy generated in current year
TOTAL_ENERGY	unsigned integer	Energy generated overall
DeviceStatus	object	Status information about inverter

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

Value name	JSON type	Description
PAC	unsigned integer	AC power
SAC	unsigned integer	AC power *Currently not implemented because not handled correctly by all inverters.*
IAC	floating point	AC current
UAC	floating point	AC voltage
FAC	floating point	AC frequency
IDC	floating point	DC current
UDC	floating point	DC voltage
DAY_ENERGY	unsigned integer	Energy generated on current day
YEAR_ENERGY	unsigned integer	Energy generated in current year
TOTAL_ENERGY	unsigned integer	Energy generated overall
DeviceStatus	object	Status information about inverter

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

Value name	JSON type	Description
IAC_L1	floating point	AC current Phase 1
IAC_L2	floating point	AC current Phase 2
IAC_L3	floating point	AC current Phase 3
UAC_L1	floating point	AC voltage Phase 1
UAC_L2	floating point	AC voltage Phase 2
UAC_L3	floating point	AC voltage Phase 3
T_AMBIENT	signed 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	JSON type	Description
DAY_PMAX	unsigned integer	Maximum AC power of current day
DAY_UACMAX	floating point	Maximum AC voltage of current day
DAY_UACMIN	floating point	Minimum AC voltage of current day
DAY_UDCMAX	floating point	Maximum DC voltage of current day
YEAR_PMAX	unsigned integer	Maximum AC power of current year
YEAR_UACMAX	floating point	Maximum AC voltage of current year
YEAR_UACMIN	floating point	Minimum AC voltage of current year
YEAR_UDCMAX	floating point	Maximum DC voltage of current year
TOTAL_PMAX	unsigned integer	Maximum AC power of current year
TOTAL_UACMAX	floating point	Maximum AC voltage overall
TOTAL_UACMIN	floating point	Minimum AC voltage overall
TOTAL_UDCMAX	floating point	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.
      number 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: Example of request body for GetInverterRealtimeData request (Scope "Device")

```
// GetInverterRealtimeData.cgi?Scope=Device&DeviceId=0&DataCollection=CommonInverterData

{
```

```
    "Data" : {
        "DAY_ENERGY" : {
                "Value" : 8000,
                "Unit" : "Wh"
        },
        "FAC" : {
                "Value" : 50,
                "Unit" : "Hz"
        },
        "IAC" : {
                "Value" : 14.54,
                "Unit" : "A"
        },
        "IDC" : {
                "Value" : 8.2,
                "Unit" : "A"
        },
        "PAC" : {
                "Value" : 3373,
                "Unit" : "W"
        },
        "SAC" : {
                "Value" : 3413,
                "Unit" : "VA"
        },
        "TOTAL_ENERGY" : {
                "Value" : 45000,
                "Unit" : "Wh"
        },
        "UAC" : {
                "Value" : 232,
                "Unit" : "V"
        },
        "UDC" : {
                "Value" : 426,
                "Unit" : "V"
        },
        "YEAR_ENERGY" : {
                "Value" : 44000,
                "Unit" : "Wh"
        },
        "DeviceStatus" : {
                "StatusCode" : 7,
                "MgmtTimerRemainingTime" : -1,
                "ErrorCode" : 0,
                "LEDColor" : 2,
                "LEDState" : 0,
                "StateToReset" : false
        }
    }
}
```

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

Listing 8: 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 {

        number 1;  # value from device with index 1.
        number 2;  # value from device with index 2.
        # .. and so on.

      }* Values;

    } __VALUE_NAME__;

  }* Data;

};
```

3.1.9 Example of request body (Scope "System")

Listing 9: Example of request body for GetInverterRealtimeData request (Scope "System")

```
// GetInverterRealtimeData.cgi?Scope=System

{
    "Data" : {
        "PAC" : {
            "Unit" : "W",
            "Values" : {
                "0" : 4330,
                "1" : 3721
            }
        },
        "DAY_ENERGY" : {
            "Unit" : "Wh",
            "Values" : {
                "0" : 6366,
                "1" : 6000
            }
        },
        "YEAR_ENERGY" : {
            "Unit" : "Wh",
            "Values" : {
                "0" : 1041607,
                "1" : 42000
            }
        },
        "TOTAL_ENERGY" : {
            "Unit" : "Wh",
            "Values" : {
                "0" : 3551131,
                "1" : 43000
            }
        }
    }
}
```

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

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 10: 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 11: Example of request body for GetSensorRealtimeData request (DataCollection "NowSensorData")

```
// GetSensorRealtimeData.cgi?Scope=Device&DeviceId=1&DataCollection=NowSensorData

{
```

```
    "Data" : {
        "0" : {
            "Value" : -9,
            "Unit" : "\u00B0C"
        },
        "1" : {
            "Value" : 24,
            "Unit" : "\u00B0C"
        },
        "2" : {
            "Value" : 589,
            "Unit" : "W\/m\u00B2"
        },
        "4" : {
            "Value" : 0,
            "Unit" : "kWh\/m\u00B2"
        }
    }
}
```

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

Listing 12: 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 Max {
                    number Value;
                    string Unit;
                };

                # 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 Min {
                    number Value;
                    string Unit;
                };
            } Day;

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

```
            } Month;

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

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

        } __CHANNEL_INDEX__;

    }* Data;

};
```

3.2.8 Example of request body (DataCollection "MinMaxSensorData")

Listing 13: Example of request body for GetSensorRealtimeData request (DataCollection "MinMaxSensorData")

```
// GetSensorRealtimeData.cgi?Scope=Device&DeviceId=1&DataCollection=MinMaxSensorData

{
    "Data" : {
        "0" : {
            "SensorActive" : true,
            "Day" : {
                "Max" : {
                    "Value" : 7,
                    "Unit" : "\u00B0C"
                },
                "Min" : {
                    "Value" : -1,
                    "Unit" : "\u00B0C"
                }
            },
            "Month" : {
                "Max" : {
                    "Value" : 15,
                    "Unit" : "\u00B0C"
                },
                "Min" : {
                    "Value" : -9,
                    "Unit" : "\u00B0C"
                }
            },
            "Year" : {
                "Max" : {
                    "Value" : 35,
                    "Unit" : "\u00B0C"
```

```
            },
            "Min" : {
                "Value" : -12,
                "Unit" : "\u00B0C"
            }
        },
        "Total" : {
            "Max" : {
                "Value" : 37,
                "Unit" : "\u00B0C"
            },
            "Min" : {
                "Value" : -15,
                "Unit" : "\u00B0C"
            }
        }
    },
    "1" : {
        "SensorActive" : true,
        "Day" : {
            "Max" : {
                "Value" : 7,
                "Unit" : "\u00B0C"
            },
            "Min" : {
                "Value" : -2,
                "Unit" : "\u00B0C"
            }
        },
        "Month" : {
            "Max" : {
                "Value" : 9,
                "Unit" : "\u00B0C"
            },
            "Min" : {
                "Value" : -8,
                "Unit" : "\u00B0C"
            }
        },
        "Year" : {
            "Max" : {
                "Value" : 27,
                "Unit" : "\u00B0C"
            },
            "Min" : {
                "Value" : -11,
                "Unit" : "\u00B0C"
            }
        },
        "Total" : {
            "Max" : {
                "Value" : 31,
                "Unit" : "\u00B0C"
            },
            "Min" : {
                "Value" : -14,
                "Unit" : "\u00B0C"
            }
        }
    },
    "2" : {
        "SensorActive" : false,
        "Day" : {
            "Max" : {
                "Value" : 593,
                "Unit" : "W\/m\u00B2"
            }
```

```
            },
            "Month" : {
                "Max" : {
                    "Value" : 785,
                    "Unit" : "W\/m\u00B2"
                }
            },
            "Year" : {
                "Max" : {
                    "Value" : 923,
                    "Unit" : "W\/m\u00B2"
                }
            },
            "Total" : {
                "Max" : {
                    "Value" : 932,
                    "Unit" : "W\/m\u00B2"
                }
            }
        },
        "3" : {
            "SensorActive" : true,
            "Day" : {
                "Max" : {
                    "Value" : 5,
                    "Unit" : "km\/h"
                }
            },
            "Month" : {
                "Max" : {
                    "Value" : 22,
                    "Unit" : "km\/h"
                }
            },
            "Year" : {
                "Max" : {
                    "Value" : 54,
                    "Unit" : "km\/h"
                }
            },
            "Total" : {
                "Max" : {
                    "Value" : 56,
                    "Unit" : "km\/h"
                }
            }
        },
        "4" : {
            "SensorActive" : false,
            "Day" : {
                "Max" : {
                    "Value" : 0,
                    "Unit" : "Wh"
                }
            },
            "Month" : {
                "Max" : {
                    "Value" : 0,
                    "Unit" : "Wh"
                }
            },
            "Year" : {
                "Max" : {
                    "Value" : 0,
                    "Unit" : "Wh"
                }
            },
```

```
            "Total" : {
                "Max" : {
                    "Value" : 0,
                    "Unit" : "Wh"
                }
            }
        },
        "5" : {
            "SensorActive" : false,
            "Day" : {
                "Max" : {
                    "Value" : 0,
                    "Unit" : "mbar"
                }
            },
            "Month" : {
                "Max" : {
                    "Value" : 0,
                    "Unit" : "mbar"
                }
            },
            "Year" : {
                "Max" : {
                    "Value" : 0,
                    "Unit" : "mbar"
                }
            },
            "Total" : {
                "Max" : {
                    "Value" : 0,
                    "Unit" : "mbar"
                }
            }
        }
    }
}
```


3.3 *GetStringRealtimeData* request

3.3.1 Availability

Platform	Since version
Fronius Hybrid	ALWAYS
Fronius Non Hybrid	ALWAYS

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 3.3.4 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 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).

3.3.5 Object structure of request body (DataCollection "NowStringControlData" and "CurrentSumStringControlData")

Listing 14: 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;

};
```

3.3.6 Example of request body (DataCollection "CurrentSumStringControlData")

Listing 15: Example of request body for GetStringRealtimeData request (DataCollection "CurrentSumStringControlData")

```
// GetStringRealtimeData.cgi?Scope=Device&DeviceId=1&DataCollection=
    CurrentSumStringControlData&TimePeriod=Day

{
    "1" : {
        "Value" : 59.57,
        "Unit" : "Ah"
    },
    "2" : {
        "Value" : 47.98,
        "Unit" : "Ah"
    },
    "3" : {
        "Value" : 47.6,
        "Unit" : "Ah"
    },
    "4" : {
        "Value" : 36.17,
        "Unit" : "Ah"
    },
    "5" : {
        "Value" : 47.91,
        "Unit" : "Ah"
```

```
    }
}
```

3.3.7 Object structure of request body (DataCollection "LastErrorStringControlData")

Listing 16: Object structure of request body for GetStringRealtimeData request (DataCollection "LastErrorStringControlData")

```
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;
}
```

3.3.8 Example of request body (DataCollection "LastErrorStringControlData")

Listing 17: Example of request body for GetStringRealtimeData request (DataCollection "LastErrorStringControlData")

```
// GetStringRealtimeData.cgi?Scope=Device&DeviceId=1&DataCollection=
    LastErrorStringControlData
{
    "TimeOfError" : "2012-03-02T23:50:00+01:00",
    "StringAverage" : {
        "Value" : 46.22,
        "Unit" : "Ah"
    },
    "Channels" : {
        "1" : {
            "Deviation" : {
                "Value" : 24.7,
                "Unit" : "%"
            },
            "Sum" : {
                "Value" : 57.66,
                "Unit" : "Ah"
            }
        },
        "2" : {
            "Deviation" : {
                "Value" : 0.3,
                "Unit" : "%"
            },
            "Sum" : {
                "Value" : 46.4,
                "Unit" : "Ah"
            }
        },
        "3" : {
            "Deviation" : {
                "Value" : -0.7,
                "Unit" : "%"
            },
            "Sum" : {
                "Value" : 45.9,
                "Unit" : "Ah"
            }
        },
        "4" : {
            "Deviation" : {
                "Value" : -24.5,
                "Unit" : "%"
            },
            "Sum" : {
                "Value" : 34.87,
                "Unit" : "Ah"
            }
        },
        "5" : {
            "Deviation" : {
                "Value" : 0.1,
                "Unit" : "%"
            },
            "Sum" : {
                "Value" : 46.29,
                "Unit" : "Ah"
            }
        }
    }
}
```

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

3.4.2 URL for HTTP requests

`/solar_api/v1/GetLoggerInfo.cgi`

3.4.3 Object structure of request body

Listing 18: Object structure of request body for GetLoggerInfo request

```
object {

  object {

    # Unique ID of the logging device.
    string UniqueID;

    # String identifying the exact product type.
    string ProductID;

    # String identifying the exact hardware platform.
    string PlatformID;

    # Hardware version of the logging device.
    string HWVersion;

    # Software version of the logging device.
    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").
    string DefaultLanguage;

    # 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;
```

```
};
```

3.4.4 Example of request body

Listing 19: Example of request body for GetLoggerInfo request

```
// GetLoggerInfo.cgi
{
    "LoggerInfo": {
        "UniqueID": "240.2",
        "ProductID": "fronius-datamanager-card",
        "PlatformID" : "wilma",
        "HWVersion": "1.4A",
        "SWVersion": "2.0.2-7",
        "TimezoneLocation": "Amsterdam",
        "TimezoneName": "CEST",
        "UTCOffset": 7200,
        "DefaultLanguage" : "de",
        "CashFactor" : 0.47,
        "CashCurrency" : "EUR",
        "CO2Factor" : 0.53,
        "CO2Unit" : "kg"
    }
}
```

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

3.5.2 URL for HTTP requests

`/solar_api/v1/GetLoggerLEDInfo.cgi`

3.5.3 Object structure of request body

Listing 20: 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 21: Example of request body for GetLoggerLEDInfo request

```
// GetLoggerLEDInfo.cgi
{
        "PowerLED" : {
                "Color" : "red",
                "State" : "blinking"
        },
        "SolarNetLED" : {
                "Color" : "green",
                "State" : "on"
        },
        "SolarWebLED" : {
                "Color" : "none",
                "State" : "off"
        },
        "WLANLED" : {
                "Color" : "green",
                "State" : "on"
        }
}
```

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

3.6.2 URL for HTTP requests

`/solar_api/v1/GetInverterInfo.cgi`

3.6.3 Object structure of request body

Listing 22: 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.
      # ATTENTION: This property will not be present or empty if device does not support
          custom names.
      string CustomName?;

      # Whether the device shall be displayed in visualizations according
      # to customer settings.
      # ATTENTION: This property may not be present if device does not support
      # visualization settings.
      number 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.
      number ErrorCode;

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

    } __INVERTER_INDEX__;

  }* Data;

};
```

3.6.4 Example of request body

Listing 23: Example of request body for GetInverterInfo request

```
// GetInverterInfo.cgi

{
  "Data" : {
      "1" : {
          "DT" : 192,
          "PVPower": 5000,
          "CustomName": "Southwest",
          "Show": 1,
          "UniqueID": "123456",
          "ErrorCode" : 0,
          "StatusCode" : 7
      },
      "2" : {
          "DT" : 192,
          "PVPower": 5000,
          "Show": 0,
          "CustomName": "Southeast",
          "UniqueID": "234567",
          "ErrorCode" : 0,
          "StatusCode" : 7
      }
  }

}
```

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
0 - 6	Startup
7	Running
8	Standby
9	Bootloading
10	Error

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

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" "StringControl" "Meter"[1] "System"[1]	Which kind of device class to search for active devices. Uses different response format

[1] [2]

3.7.4 DeviceClass not System

Listing 24: 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 {

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

      # Optional attribute: serialnumber
      string  Serial;

    } __DEVICE_INDEX__;

  }* Data;

};
```

[1] Supported since version 3.3.4-5
[2] Supported since version 3.6.1-3

Listing 25: Example of request body for GetActiveDeviceInfo Inverter request

```
// GetActiveDeviceInfo.cgi?DeviceClass=Inverter
{
  "Data" : {
      "1" : {
          "DT" : 192
      },
      "2" : {
          "DT" : 192
      }
  }
}
```

Listing 26: Example of request body for GetActiveDeviceInfo request on non hybrid inverter systems

```
// GetActiveDeviceInfo.cgi?DeviceClass=System
{
        "Body" : {
                "Data" : {
                        "Inverter" : {
                                "1" : {
                                        "DT" : 158
                                },
                                "2" : {
                                        "DT" : 120
                                }
                        },
                        "SensorCard" : {
                                "1" : {
                                        "DT" : 254,
                                        "ChannelNames" : [
                                                "Temperature␣1",
                                                "Temperature␣2",
                                                "Irradiation",
                                                "Digital␣1",
                                                "Digital␣2",
                                                "Current"
                                        ]
                                }
                        },
                        "StringControl" : {
                                "17" : {
                                        "DT" : 252
                                }
                        },
                        "Meter" : {
                                "0" : {
                                        "DT" : -1,
                                        "Serial" : "14300120"
                                }
                        },
                        "Ohmpilot" : {
                            "0" : {
                                "DT" : -1,
                                "Serial" : "112233445566"
                            }
                        }
                }
        }
}
```

Listing 27: Example of request body for GetActiveDeviceInfo request on hybrid inverter systems

```
// GetActiveDeviceInfo.cgi?DeviceClass=System
{
        "Body" : {
                "Data" : {
```

```
                        "Inverter" : {
                                "1" : {
                                        "DT" : 99
                                }
                        },
                        "SensorCard" : {},
                        "StringControl" : {},
                        "Meter" : {
                                "0" : {
                                        "DT" : -1,
                                        "Serial" : "15060037"
                                }
                        },
                        "Storage" : {
                                "0" : {
                                        "DT" : -1,
                                        "Serial" : "25310943"
                                }
                        }
                }
        }
}
```

3.7.5 DeviceClass is System

Listing 28: 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 of the device. (only for Inverter, SensorCard or StringControl;
            others have -1)
        integer DT;

        # Optional attribute: serialnumber
        string  Serial;

      } __DEVICE_INDEX__;

    } __DEVICE_CLASS__;

  }* Data;

};
```

Listing 29: Example of request body for GetActiveDeviceInfo request

```
{
  "Body" : {
        "Data" : {
                "Inverter" : {
                        "2" : {
                                "DT" : 126
                        }
                },
                "SensorCard" : {},
```

```
                "StringControl" : {},
                "Meter" : {
                        "1" : {
                                "DT" : -1,
                                "Serial", "14300002"
                        },
                        "2" : {
                                "DT" : -1,
                                "Serial", "14300001"
                        }
                }
        }
  }
}
```

3.8 *GetMeterRealtimeData* request

This request provides detailed information about Meter devices. Inactive channels are not included in the response and may vary depended 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

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 System-Request

Listing 30: 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)
    array { double __CHANNEL_NAME__ } ;
```

```
  } * DeviceId;

} Data ;
```

Listing 31: Example of response body for GetMeterRealtimeData System request

```
//GetMeterRealtimeData.cgi?Scope=System
{
  "Body" : {
    "Data" : {
      "0" : {
        /* older devices do not support any detailed information */
        "TimeStamp" : 1406816001,
        "Enable" : 1,
        "Visible" : 1,
        "PowerReal_P_Sum" : -834.13,
        "Meter_Location_Current" : 0,
        "PowerReal_P_Phase_1" : -261.86,
        "PowerReal_P_Phase_2" : -296.26,
        "PowerReal_P_Phase_3" : -276.01,
        "PowerReactive_Q_Sum" : 489.34,
        "PowerReactive_Q_Phase_1" : 169.63,
        "PowerReactive_Q_Phase_2" : 158.39,
        "PowerReactive_Q_Phase_3" : 161.32,
        "Current_AC_Phase_1" : 1.43,
        "Current_AC_Phase_2" : 1.522,
        "Current_AC_Phase_3" : 1.44,
        "Voltage_AC_Phase_1" : 233.1,
        "Voltage_AC_Phase_2" : 234.4,
        "Voltage_AC_Phase_3" : 233.8,
        "Voltage_AC_PhaseToPhase_12" : 404.9,
        "Voltage_AC_PhaseToPhase_23" : 405.5,
        "Voltage_AC_PhaseToPhase_31" : 404.3,
        "Frequency_Phase_Average" : 50,
        "PowerApparent_S_Sum" : 967,
        "PowerFactor_Sum" : 0.86,
        "PowerFactor_Phase_1" : 0.83,
        "PowerFactor_Phase_2" : 0.88,
        "PowerFactor_Phase_3" : 0.86,
        "EnergyReal_WAC_Sum_Produced" : 33989,
        "EnergyReal_WAC_Sum_Consumed" : 1365,
        "EnergyReactive_VArAC_Sum_Produced" : 4020,
        "EnergyReactive_VArAC_Sum_Consumed" : 204310,
        "EnergyReal_WAC_Plus_Absolute" : 1365,
        "EnergyReal_WAC_Minus_Absolute" : 33989
      },
      "1" : {
        /* only supported with newer versions:
        #    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
        */
        "Details" : {
           "Serial" : "15160164",
           "Model" : "Fronius␣SmartMeter",
           "Manufacturer" : "Fronius"
        },
        "TimeStamp" : 1406816000,
        "Enable" : 1,
        "Visible" : 1,
        "PowerReal_P_Sum" : -235.27,
        "Meter_Location_Current" : 1,
        "PowerReal_P_Phase_1" : -235.27,
        "PowerReal_P_Phase_2" : 0,
        "PowerReal_P_Phase_3" : 0,
        "PowerReactive_Q_Sum" : -18.58,
        "PowerReactive_Q_Phase_1" : -18.58,
```

```
        "PowerReactive_Q_Phase_2" : 0,
        "PowerReactive_Q_Phase_3" : 0,
        "Current_AC_Phase_1" : 1.019,
        "Current_AC_Phase_2" : 0.012,
        "Current_AC_Phase_3" : 0.022,
        "Voltage_AC_Phase_1" : 232.7,
        "Voltage_AC_Phase_2" : 107.9,
        "Voltage_AC_Phase_3" : 107.9,
        "Voltage_AC_PhaseToPhase_12" : 301.5,
        "Voltage_AC_PhaseToPhase_23" : 186.9,
        "Voltage_AC_PhaseToPhase_31" : 301.5,
        "Frequency_Phase_Average" : 50,
        "PowerApparent_S_Sum" : 236,
        "PowerFactor_Sum" : 0.99,
        "PowerFactor_Phase_1" : 0.99,
        "PowerFactor_Phase_2" : 1,
        "PowerFactor_Phase_3" : 1,
        "EnergyReal_WAC_Sum_Produced" : 0,
        "EnergyReal_WAC_Sum_Consumed" : 10439,
        "EnergyReactive_VArAC_Sum_Produced" : 7490,
        "EnergyReactive_VArAC_Sum_Consumed" : 6110,
        "EnergyReal_WAC_Plus_Absolute" : 0,
        "EnergyReal_WAC_Minus_Absolute" : 10439
      }
    }
  }
}
```

3.8.5 Device-Request

Listing 32: 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)
    array { double __CHANNEL_NAME__ } ;

} Data ;
```

Listing 33: Example of response body for GetMeterRealtimeData Device request

```
//GetMeterRealtimeData.cgi?Scope=Device&DeviceId=1
{
  "Body" : {
    "Data" : {
      "Details" : {
         "Serial" : "15160164",
         "Model" : "Fronius SmartMeter",
         "Manufacturer" : "Fronius"
      },
      "TimeStamp" : 1406818247,
      "Enable" : 1,
      "Visible" : 1,
      "PowerReal_P_Sum" : -835.93,
      "Meter_Location_Current" : 0,
      "PowerReal_P_Phase_1" : -262.8,
      "PowerReal_P_Phase_2" : -293.37,
```

```
      "PowerReal_P_Phase_3" : -279.76,
      "PowerReactive_Q_Sum" : 496.37,
      "PowerReactive_Q_Phase_1" : 170.06,
      "PowerReactive_Q_Phase_2" : 163.32,
      "PowerReactive_Q_Phase_3" : 162.99,
      "Current_AC_Phase_1" : 1.418,
      "Current_AC_Phase_2" : 1.502,
      "Current_AC_Phase_3" : 1.438,
      "Voltage_AC_Phase_1" : 233.8,
      "Voltage_AC_Phase_2" : 235.5,
      "Voltage_AC_Phase_3" : 235,
      "Voltage_AC_PhaseToPhase_12" : 406.4,
      "Voltage_AC_PhaseToPhase_23" : 407.5,
      "Voltage_AC_PhaseToPhase_31" : 406,
      "Frequency_Phase_Average" : 50,
      "PowerApparent_S_Sum" : 971,
      "PowerFactor_Sum" : 0.86,
      "PowerFactor_Phase_1" : 0.83,
      "PowerFactor_Phase_2" : 0.87,
      "PowerFactor_Phase_3" : 0.86,
      "EnergyReal_WAC_Sum_Produced" : 34474,
      "EnergyReal_WAC_Sum_Consumed" : 1365,
      "EnergyReactive_VArAC_Sum_Produced" : 4020,
      "EnergyReactive_VArAC_Sum_Consumed" : 207380,
      "EnergyReal_WAC_Plus_Absolute" : 1365,
      "EnergyReal_WAC_Minus_Absolute" : 34474
    }
  }
}
```


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

3.9.2 URL for HTTP requests

`/solar_api/v1/GetStorageRealtimeData.cgi`

3.9.3 Parameters

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

3.9.4 Details about Channels

Table 1: Channel and value description

<table>
<tr><th>Name of channel</th><th>Description</th></tr>
<tr><td>Status_BatteryCell</td><td>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>
</table>

3.9.5 System-Request

Listing 34: Object structure of response body for GetStorageRealtimeData request

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

  object {

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

          string Model;

          string Manufacturer;
      } Details;

      #channels of device (textual name and value)
      array { double __CHANNEL_NAME__ } ;

    } Controller;

    object {

      object {
        string Serial;

        string Model;

        string Manufacturer;
      } Details;
```

```
      #channels of device (textual name and value)
      array { double __CHANNEL_NAME__ } ;

    } * Modules;

  } * DeviceId;

} Data ;
```

Listing 35: Example of response body for GetStorageRealtimeData System request

```
//GetStorageRealtimeData.cgi?Scope=System
{
  "Head" : {
    "RequestArguments" : {
      "DeviceClass" : "Storage",
      "Scope" : "System"
    },
    "Status" : {
      "Code" : 0,
      "Reason" : "",
      "UserMessage" : ""
    },
    "Timestamp" : "2015-07-10T07:12:40+02:00"
  },
  "Body" : {
    "Data" : {
      "0" : {
        "Controller" : {
          "Details" : {
            "Serial" : "26175063",
            "Model" : "C5",
            "Manufacturer" : "Sony"
          },
          "StateOfCharge_Relative" : 4,
          "Voltage_DC" : 370.4,
          "Current_DC" : 0,
          "Voltage_DC_Maximum_Cell" : 2.959,
          "Voltage_DC_Minimum_Cell" : 2.822,
          "Temperature_Cell" : 25.6,
          "DesignedCapacity" : 9600
        },
        "Modules" : [
          {
            "Details" : {
              "Serial" : "S0120028626",
              "Model" : "P49929080N",
              "Manufacturer" : "Sony"
            },
            "TimeStamp" : 1436505153,
            "Enable" : 1,
            "StateOfCharge_Relative" : 5,
            "Current_DC" : -0.02,
            "Voltage_DC" : 46.621,
            "Temperature_Cell" : 26.3,
            "Temperature_Cell_Maximum" : 26.9,
            "Temperature_Cell_Minimum" : 26,
            "Voltage_DC_Maximum_Cell" : 2.951,
            "Voltage_DC_Minimum_Cell" : 2.866,
            "CycleCount_BatteryCell" : 8,
            "Status_BatteryCell" : 50,
            "DesignedCapacity" : 1200
          },
          {
            "Details" : {
              "Serial" : "S0120028389",
              "Model" : "P49929080N",
```

```
      "Manufacturer" : "Sony"
    },
    "TimeStamp" : 1436505153,
    "Enable" : 1,
    "StateOfCharge_Relative" : 4,
    "Current_DC" : -0.02,
    "Voltage_DC" : 45.866,
    "Temperature_Cell" : 26.1,
    "Temperature_Cell_Maximum" : 26.7,
    "Temperature_Cell_Minimum" : 25.9,
    "Voltage_DC_Maximum_Cell" : 2.894,
    "Voltage_DC_Minimum_Cell" : 2.822,
    "CycleCount_BatteryCell" : 8,
    "Status_BatteryCell" : 50,
    "DesignedCapacity" : 1200
  },
  {
    "Details" : {
      "Serial" : "S0120028446",
      "Model" : "P49929080N",
      "Manufacturer" : "Sony"
    },
    "TimeStamp" : 1436505153,
    "Enable" : 1,
    "StateOfCharge_Relative" : 5,
    "Current_DC" : -0.02,
    "Voltage_DC" : 46.478,
    "Temperature_Cell" : 26.1,
    "Temperature_Cell_Maximum" : 26.6,
    "Temperature_Cell_Minimum" : 25.8,
    "Voltage_DC_Maximum_Cell" : 2.937,
    "Voltage_DC_Minimum_Cell" : 2.883,
    "CycleCount_BatteryCell" : 8,
    "Status_BatteryCell" : 50,
    "DesignedCapacity" : 1200
  },
  {
    "Details" : {
      "Serial" : "S0120028435",
      "Model" : "P49929080N",
      "Manufacturer" : "Sony"
    },
    "TimeStamp" : 1436505153,
    "Enable" : 1,
    "StateOfCharge_Relative" : 5,
    "Current_DC" : -0.02,
    "Voltage_DC" : 46.47,
    "Temperature_Cell" : 25.8,
    "Temperature_Cell_Maximum" : 26.5,
    "Temperature_Cell_Minimum" : 25.5,
    "Voltage_DC_Maximum_Cell" : 2.938,
    "Voltage_DC_Minimum_Cell" : 2.88,
    "CycleCount_BatteryCell" : 8,
    "Status_BatteryCell" : 50,
    "DesignedCapacity" : 1200
  },
  {
    "Details" : {
      "Serial" : "S0120028479",
      "Model" : "P49929080N",
      "Manufacturer" : "Sony"
    },
    "TimeStamp" : 1436505153,
    "Enable" : 1,
    "StateOfCharge_Relative" : 5,
    "Current_DC" : -0.02,
    "Voltage_DC" : 46.49,
```

```
      "Temperature_Cell" : 25.6,
      "Temperature_Cell_Maximum" : 26.3,
      "Temperature_Cell_Minimum" : 25.2,
      "Voltage_DC_Maximum_Cell" : 2.94,
      "Voltage_DC_Minimum_Cell" : 2.877,
      "CycleCount_BatteryCell" : 8,
      "Status_BatteryCell" : 50,
      "DesignedCapacity" : 1200
    },
    {
      "Details" : {
        "Serial" : "S012002885B",
        "Model" : "P49929080N",
        "Manufacturer" : "Sony"
      },
      "TimeStamp" : 1436505153,
      "Enable" : 1,
      "StateOfCharge_Relative" : 5,
      "Current_DC" : -0.02,
      "Voltage_DC" : 46.252,
      "Temperature_Cell" : 25.3,
      "Temperature_Cell_Maximum" : 26,
      "Temperature_Cell_Minimum" : 25,
      "Voltage_DC_Maximum_Cell" : 2.959,
      "Voltage_DC_Minimum_Cell" : 2.847,
      "CycleCount_BatteryCell" : 8,
      "Status_BatteryCell" : 50,
      "DesignedCapacity" : 1200
    },
    {
      "Details" : {
        "Serial" : "S012002884A",
        "Model" : "P49929080N",
        "Manufacturer" : "Sony"
      },
      "TimeStamp" : 1436505153,
      "Enable" : 1,
      "StateOfCharge_Relative" : 5,
      "Current_DC" : -0.02,
      "Voltage_DC" : 46.116,
      "Temperature_Cell" : 25.1,
      "Temperature_Cell_Maximum" : 25.8,
      "Temperature_Cell_Minimum" : 24.9,
      "Voltage_DC_Maximum_Cell" : 2.899,
      "Voltage_DC_Minimum_Cell" : 2.867,
      "CycleCount_BatteryCell" : 8,
      "Status_BatteryCell" : 50,
      "DesignedCapacity" : 1200
    },
    {
      "Details" : {
        "Serial" : "S012002857A",
        "Model" : "P49929080N",
        "Manufacturer" : "Sony"
      },
      "TimeStamp" : 1436505153,
      "Enable" : 1,
      "StateOfCharge_Relative" : 5,
      "Current_DC" : -0.02,
      "Voltage_DC" : 46.211,
      "Temperature_Cell" : 24.6,
      "Temperature_Cell_Maximum" : 25.1,
      "Temperature_Cell_Minimum" : 24.2,
      "Voltage_DC_Maximum_Cell" : 2.912,
      "Voltage_DC_Minimum_Cell" : 2.865,
      "CycleCount_BatteryCell" : 8,
      "Status_BatteryCell" : 50,
```

```
            "DesignedCapacity" : 1200
          }
        ]
      }
    }
  }
}
```

3.9.6 Device-Request

Listing 36: Object structure of response body for GetStorageRealtimeData request

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

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

                  string Model;

                  string Manufacturer;
          } Details;

      #channels of device (textual name and value)
      array { double __CHANNEL_NAME__ } ;

    } Controller;

    object {

      object {
                  string Serial;

                  string Model;

                  string Manufacturer;
          } Details;

      #channels of device (textual name and value)
      array { double __CHANNEL_NAME__ } ;

    } * Modules;

} Data ;
```

Listing 37: Example of response body for GetStorageRealtimeData Device request

```
//GetStorageRealtimeData.cgi?Scope=Device&DeviceId=0
{
  "Head" : {
    "RequestArguments" : {
      "DeviceClass" : "Storage",
      "DeviceId" : "0",
      "Scope" : "Device"
    },
    "Status" : {
      "Code" : 0,
      "Reason" : "",
      "UserMessage" : ""
    },
    "Timestamp" : "2015-07-10T07:49:09+02:00"
  },
  "Body" : {
    "Data" : {
```

```
    "Controller" : {
      "Details" : {
        "Serial" : "26175063",
        "Model" : "C5",
        "Manufacturer" : "Sony"
      },
      "TimeStamp" : 1436507345,
      "Enable" : 1,
      "StateOfCharge_Relative" : 4,
      "Voltage_DC" : 368.9,
      "Current_DC" : 0,
      "Temperature_Cell" : 25.5,
      "Voltage_DC_Maximum_Cell" : 2.95,
      "Voltage_DC_Minimum_Cell" : 2.806,
      "DesignedCapacity" : 9600
    },
    "Modules" : [
      {
        "Details" : {
          "Serial" : "S0120028626",
          "Model" : "P49929080N",
          "Manufacturer" : "Sony"
        },
        "TimeStamp" : 1436507345,
        "Enable" : 1,
        "StateOfCharge_Relative" : 5,
        "Voltage_DC" : 46.436,
        "Current_DC" : -0.02,
        "Temperature_Cell" : 26.3,
        "Voltage_DC_Maximum_Cell" : 2.941,
        "Voltage_DC_Minimum_Cell" : 2.853,
        "Temperature_Cell_Maximum" : 26.8,
        "Temperature_Cell_Minimum" : 26,
        "CycleCount_BatteryCell" : 8,
        "Status_BatteryCell" : 50,
        "DesignedCapacity" : 1200
      },
      {
        "Details" : {
          "Serial" : "S0120028389",
          "Model" : "P49929080N",
          "Manufacturer" : "Sony"
        },
        "TimeStamp" : 1436507345,
        "Enable" : 1,
        "StateOfCharge_Relative" : 4,
        "Voltage_DC" : 45.649,
        "Current_DC" : -0.02,
        "Temperature_Cell" : 26.1,
        "Voltage_DC_Maximum_Cell" : 2.882,
        "Voltage_DC_Minimum_Cell" : 2.806,
        "Temperature_Cell_Maximum" : 26.7,
        "Temperature_Cell_Minimum" : 25.9,
        "CycleCount_BatteryCell" : 8,
        "Status_BatteryCell" : 50,
        "DesignedCapacity" : 1200
      },
      {
        "Details" : {
          "Serial" : "S0120028446",
          "Model" : "P49929080N",
          "Manufacturer" : "Sony"
        },
        "TimeStamp" : 1436507345,
        "Enable" : 1,
        "StateOfCharge_Relative" : 5,
        "Voltage_DC" : 46.292,
```

```
      "Current_DC" : -0.02,
      "Temperature_Cell" : 26,
      "Voltage_DC_Maximum_Cell" : 2.928,
      "Voltage_DC_Minimum_Cell" : 2.87,
      "Temperature_Cell_Maximum" : 26.6,
      "Temperature_Cell_Minimum" : 25.8,
      "CycleCount_BatteryCell" : 8,
      "Status_BatteryCell" : 50,
      "DesignedCapacity" : 1200
    },
    {
      "Details" : {
        "Serial" : "S0120028435",
        "Model" : "P49929080N",
        "Manufacturer" : "Sony"
      },
      "TimeStamp" : 1436507345,
      "Enable" : 1,
      "StateOfCharge_Relative" : 5,
      "Voltage_DC" : 46.281,
      "Current_DC" : -0.02,
      "Temperature_Cell" : 25.8,
      "Voltage_DC_Maximum_Cell" : 2.927,
      "Voltage_DC_Minimum_Cell" : 2.866,
      "Temperature_Cell_Maximum" : 26.4,
      "Temperature_Cell_Minimum" : 25.5,
      "CycleCount_BatteryCell" : 8,
      "Status_BatteryCell" : 50,
      "DesignedCapacity" : 1200
    },
    {
      "Details" : {
        "Serial" : "S0120028479",
        "Model" : "P49929080N",
        "Manufacturer" : "Sony"
      },
      "TimeStamp" : 1436507345,
      "Enable" : 1,
      "StateOfCharge_Relative" : 5,
      "Voltage_DC" : 46.31,
      "Current_DC" : -0.02,
      "Temperature_Cell" : 25.5,
      "Voltage_DC_Maximum_Cell" : 2.931,
      "Voltage_DC_Minimum_Cell" : 2.864,
      "Temperature_Cell_Maximum" : 26.3,
      "Temperature_Cell_Minimum" : 25.2,
      "CycleCount_BatteryCell" : 8,
      "Status_BatteryCell" : 50,
      "DesignedCapacity" : 1200
    },
    {
      "Details" : {
        "Serial" : "S012002885B",
        "Model" : "P49929080N",
        "Manufacturer" : "Sony"
      },
      "TimeStamp" : 1436507345,
      "Enable" : 1,
      "StateOfCharge_Relative" : 5,
      "Voltage_DC" : 46.055,
      "Current_DC" : -0.03,
      "Temperature_Cell" : 25.2,
      "Voltage_DC_Maximum_Cell" : 2.95,
      "Voltage_DC_Minimum_Cell" : 2.832,
      "Temperature_Cell_Maximum" : 26,
      "Temperature_Cell_Minimum" : 25,
      "CycleCount_BatteryCell" : 8,
```

```
          "Status_BatteryCell" : 50,
          "DesignedCapacity" : 1200
        },
        {
          "Details" : {
            "Serial" : "S012002884A",
            "Model" : "P49929080N",
            "Manufacturer" : "Sony"
          },
          "TimeStamp" : 1436507345,
          "Enable" : 1,
          "StateOfCharge_Relative" : 5,
          "Voltage_DC" : 45.907,
          "Current_DC" : -0.02,
          "Temperature_Cell" : 25,
          "Voltage_DC_Maximum_Cell" : 2.887,
          "Voltage_DC_Minimum_Cell" : 2.853,
          "Temperature_Cell_Maximum" : 25.7,
          "Temperature_Cell_Minimum" : 24.9,
          "CycleCount_BatteryCell" : 8,
          "Status_BatteryCell" : 50,
          "DesignedCapacity" : 1200
        },
        {
          "Details" : {
            "Serial" : "S012002857A",
            "Model" : "P49929080N",
            "Manufacturer" : "Sony"
          },
          "TimeStamp" : 1436507345,
          "Enable" : 1,
          "StateOfCharge_Relative" : 5,
          "Voltage_DC" : 46.003,
          "Current_DC" : -0.02,
          "Temperature_Cell" : 24.5,
          "Voltage_DC_Maximum_Cell" : 2.9,
          "Voltage_DC_Minimum_Cell" : 2.851,
          "Temperature_Cell_Maximum" : 25.1,
          "Temperature_Cell_Minimum" : 24.2,
          "CycleCount_BatteryCell" : 8,
          "Status_BatteryCell" : 50,
          "DesignedCapacity" : 1200
        }
      ]
    }
  }
}
```


3.10 *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.

3.10.1 Availability

Platform	Since version
Fronius Hybrid	1.2.1-X
Fronius Non Hybrid	3.3.9-X

3.10.2 URL for HTTP requests

`/solar_api/v1/GetPowerFlowRealtimeData.fcgi`

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

3.10.3 Parameters

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

3.10.4 Request

Listing 38: Object structure of response body for GetPowerFlowRealtimeData request

```
object {

    object {

        # mandatory field
        # "produce-only", "meter", "vague-meter", "bidirectional" or "ac-coupled"
        string Mode;

        # 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
        bool BackupMode;

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

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

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

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

        # mandatory field
        # available since Fronius Hybrid version 1.3.1-1
        # not available for Fronius Non Hybrid
        # current relative self consumption in %, null if no smart meter is connected
        float rel_SelfConsumption;

                # mandatory field
        # available since Fronius Hybrid version 1.3.1-1
        # not available for Fronius Non Hybrid
        # current relative autonomy in %, null if no smart meter is connected
        float rel_Autonomy;

        # optional field
        # "load" or "grid"
        string Meter_Location;

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

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

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

```
        float E_Total;

    } Site;

    object {

        object {

            # mandatory field
            # device type of inverter
            int DT;

            # mandatory field
            # current power in Watt, null if not running
            int P;

            # optional field
            # current state of charge in % ( 0 - 100% )
            int SOC;

            # optional field
            #  "disabled", "normal", "service", "charge boost",
            #  "nearly depleted", "suspended", "calibrate",
            #  "grid support", "deplete recovery" or "none operable"
            string Battery_Mode;

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

                    # optional field
            # implemented since Fronius Non Hybrid version 3.7.1-1
            #                   Fronius Hybrid version     1.3.1-1
            # Energy [Wh] this year, null if no inverter is connected
            float E_Year;

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

        } * DeviceId; #SolarNet ring address

    } Inverters;

} Data ;
```

Listing 39: Example of response body for GetPowerFlowRealtimeData on Fronius Hybrid System


```
{
        "Head" : {
                "RequestArguments" : {},
                "Status" : {
                        "Code" : 0,
                        "Reason" : "",
                        "UserMessage" : ""
                },
                "Timestamp" : "2015-08-12T07:15:28+02:00"
        },
        "Body" : {
                "Data" : {
                        "Site" : {
                                "Mode" : "bidirectional",
                                "BackupMode" : true,
```

```
                                "P_Grid" : -316.92,
                                "P_Load" : -18.08,
                                "P_Akku" : -61.78,
                                "P_PV" : 449.8,
                                "E_Day" : 17844,
                                "E_Year" : 1325177.9,
                                "E_Total" : 2874459.8,
                                "Meter_Location" : "grid"
                        },
                        "Inverters" : {
                                "1" : {
                                        "DT" : 99,
                                        "P" : 335,
                                        "SOC" : 99,
                                        "Battery_Mode" : "normal",
                                        "E_Day" : 17844,
                                        "E_Year" : 1325177.9,
                                        "E_Total" : 2874459.8
                                }
                        }
                }
        }
}
```


Listing 40: Example of response body for GetPowerFlowRealtimeData on Fronius Non Hybrid System


```
{
        "Head" : {
                "RequestArguments" : {},
                "Status" : {
                        "Code" : 0,
                        "Reason" : "",
                        "UserMessage" : ""
                },
                "Timestamp" : "2015-08-12T07:15:28+02:00"
        },
        "Body" : {
                "Data" : {
                        "Site" : {
                                "Mode" : "produce-only",
                                "P_Grid" : null,
                                "P_Load" : null,
                                "P_Akku" : null,
                                "P_PV" : 449.8
                        },
                        "Inverters" : {
                                "1" : {
                                        "DT" : 99,
                                        "P" : 250,
                                },
                                "80" : {
                                        "DT" : 99,
                                        "P" : 200,
                                }
                        }
                }
        }
}
```


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

4.2 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 $Digital\_PowerManagementRelay\_Out\_1$, $InverterErrors$ and $InverterEvents$ are event triggered and may occure every time.

Table 2: Available channels

Name	Unit
TimeSpanInSec	sec
Digital_PowerManagementRelay_Out_1	1
EnergyReal_WAC_Sum_Produced	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
Digital_PowerManagementRelay_Out_1	1

4.3 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*
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 2. *Mandatory, multiple times*	
DeviceClass	String	"Inverter" ("SensorCard") ("StringControl")	Which kind of device will be queried. *Mandatory and accepted only if Scope is not "System"*
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.

4.3.1 Object Structure of response body

Listing 41: 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
      [ int NodeType; ]

      # Optional Devicetype if localnet node
      [ int DeviceType; ]

      # Starting date of the series (i.e. date of the first value in Values)
      string Start/yyyy-MM-ddThh-mm-ss%z/;

      # Starting date of the series (i.e. date of the first value in Values)
      string End/yyyy-MM-ddThh-mm-ss%z/;


      # 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"
          # Unavailable datapoints are included but have value null
```

```
          array { union {number;null;}; } Values;
        } __CHANNEL_NAME__;

      }* Data;

    }* __DEVICE_ID__;

  } Data;

};
```

4.4 Example of response body

4.4.1 Meter data

Listing 42: detailed query parameters

```
/solar_api/v1/GetArchiveData.cgi?Scope=System&StartDate=1.6.2015&EndDate=1.6.2015&Channel=
    TimeSpanInSec&Channel=EnergyReal_WAC_Plus_Absolute&Channel=
    EnergyReal_WAC_Minus_Absolute&Channel=Meter_Location_Current
```

Listing 43: detailed response body

```
{
        "Head" : {
                "RequestArguments" : {
                        "Scope" : "System",
                        "StartDate" : "2015-06-01T00:00:00+02:00",
                        "EndDate" : "2015-06-01T23:59:59+02:00",
                        "Channel" : "TimeSpanInSec",
                        "Channel" : "EnergyReal_WAC_Plus_Absolute",
                        "Channel" : "EnergyReal_WAC_Minus_Absolute",
                        "SeriesType" : "Detail",
                        "HumanReadable" : "True"
                },
                "Status" : {
                        "Code" : 0,
                        "Reason" : "",
                        "UserMessage" : "",
                        "ErrorDetail" : {
                                "Nodes" : []
                        }
                },
                "Timestamp" : "2015-06-01T13:47:50+02:00"
        },
        "Body" : {
                "Data" : {
                        "meter:15160189" : {
                                "Start" : "2015-06-01T00:00:00+02:00",
                                "End" : "2015-06-01T23:59:59+02:00",
                                "Data" : {
                                        "EnergyReal_WAC_Plus_Absolute" : {
                                                "_comment" : "channelId=167772424",
                                                "Unit" : "Wh",
                                                "Values" : {
                                                        "1" : 111815,
                                                        "300" : 111815,
                                                        /* Logpoint missing here */
                                                        "48900" : 112008,
                                                        "49200" : 112068,
                                                        "49500" : 112072
                                                }
                                        },
                                        "EnergyReal_WAC_Minus_Absolute" : {
                                                "_comment" : "channelId=167837960",
                                                "Unit" : "Wh",
```

```
                                        "Values" : {
                                                "1" : 46446,
                                                "300" : 46446,
                                                /* Logpoint missing here */
                                                "48900" : 48084,
                                                "49200" : 48084,
                                                "49500" : 48087
                                        }
                                },
                                "Meter_Location_Current" : {
                                        "_comment" : "channelId=117050390",
                                        "Unit" : "1",
                                        "Values" : {
                                                "1" : 0,
                                                "300" : 0,
                                                /* Logpoint missing here */
                                                "49500" : 0,
                                                "49800" : 0,
                                                "50100" : 0
                                        }
                                }
                        },
                        "inverter\/1" : {
                                "NodeType" : 97,
                                "DeviceType" : 99,
                                "Start" : "2015-06-01T00:00:00+02:00",
                                "End" : "2015-06-01T23:59:59+02:00",
                                "Data" : {
                                        "TimeSpanInSec" : {
                                                "_comment" : "channelId=65549",
                                                "Unit" : "sec",
                                                "Values" : {
                                                        "0" : 299,
                                                        "300" : 300,
                                                        /* Logpoint missing here */
                                                        "48900" : 238,
                                                        "49200" : 298,
                                                        "49500" : 300
                                                }
                                        }
                                }
                        }
                }
        }
}
```

4.4.2 Inverter data

Listing 44: detailed query parameters

```
/solar_api/v1/GetArchiveData.cgi?Scope=System&StartDate=1.6.2015&EndDate=1.6.2015&Channel=
    EnergyReal_WAC_Sum_Produced
```

Listing 45: detailed response body with multiple inverters

```
{
        "Head" : {
                "RequestArguments" : {
                        "Scope" : "System",
                        "StartDate" : "2015-06-01T00:00:00+02:00",
                        "EndDate" : "2015-06-01T23:59:59+02:00",
                        "Channel" : "EnergyReal_WAC_Sum_Produced",
                        "SeriesType" : "Detail",
                        "HumanReadable" : "True"
                },
```

```
                "Status" : {
                        "Code" : 0,
                        "Reason" : "",
                        "UserMessage" : "",
                        "ErrorDetail" : {
                                "Nodes" : []
                        }
                },
                "Timestamp" : "2015-06-01T14:11:53+02:00"
        },
        "Body" : {
                "Data" : {
                        "inverter\/0" : {
                                "NodeType" : 97,
                                "DeviceType" : 113,
                                "Start" : "2015-06-01T00:00:00+02:00",
                                "End" : "2015-06-01T23:59:59+02:00",
                                "Data" : {
                                        "EnergyReal_WAC_Sum_Produced" : {
                                                "_comment" : "channelId=67830024",
                                                "Unit" : "Wh",
                                                "Values" : {
                                                        "0": 3244,
                                                        "300": 3423,
                                                        /* Logpoint missing here */
                                                        "900": 4324,
                                                        "1200": 5443,
                                                        "1500": 6567
                                                }
                                        }
                                }
                        },
                        "inverter\/3": {
                                "NodeType" : 157,
                                "DeviceType" : 113,
                                "Start" : "2015-06-01T00:00:00+02:00",
                                "End" : "2015-06-01T23:59:59+02:00",
                                "Data": {
                                        "EnergyReal_WAC_Sum_Produced": {
                        "Unit" : "Wh",
                        "Values": {
                                                        "0": 2324,
                                                        "300": 4365,
                                                        "600": 4376,
                                                        "900": 5466,
                                                        "1200": 5465,
                                                        "1500": 5446
                                                }
                                        }
                                }
                        }
                }
        }
}
```

4.4.3 Errors - Structure

Listing 46: detailed query parameters

```
/solar_api/v1/GetArchiveData.cgi?Scope=System&StartDate=24.5.2015&EndDate=28.5.2015&Channel
    =InverterErrors
```

Listing 47: Example of response body for inverter errors

```
{
        "Head" : {
```

```
                "RequestArguments" : {
                        "Scope" : "System",
                        "StartDate" : "2015-05-24T00:00:00+02:00",
                        "EndDate" : "2015-05-28T23:59:59+02:00",
                        "Channel" : "InverterErrors",
                        "SeriesType" : "Detail",
                        "HumanReadable" : "True"
                },
                "Status" : {
                        "Code" : 0,
                        "Reason" : "",
                        "UserMessage" : "",
                        "ErrorDetail" : {
                                "Nodes" : []
                        }
                },
                "Timestamp" : "2015-06-01T14:23:13+02:00"
        },
        "Body" : {
                "Data" : {
                        "inverter\/1" : {
                                "NodeType" : 97,
                                "DeviceType" : 99,
                                "Start" : "2015-05-24T00:00:00+02:00",
                                "End" : "2015-05-28T23:59:59+02:00",
                                "Data" : {
                                        "InverterErrors" : {
                                                "_comment" : "channelId=16646144",
                                                "Unit" : "Object",
                                                "Values" : {
                                                        "123180" :
                                                        {
                                                                "flags" : [ "fatal","
                                                                    official" ],
                                                                "#" : 731
                                                        },
                                                        "123240" :
                                                        {
                                                                "flags" : [ "fatal", "
                                                                    official"],
                                                                "#" : 766
                                                        },
                                                        "123240" :
                                                        {
                                                                "flags" : [ "fatal", "
                                                                    official"],
                                                                "#" : 482
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }
}
```


4.4.4 Events - Structure

Listing 48: detailed query parameters


```
/solar_api/v1/GetArchiveData.cgi?Scope=System&StartDate=24.5.2015&EndDate=28.5.2015&Channel
    =InverterEvents
```


Listing 49: Example of response body for inverter events


```
{
```

```
    "Head": {
        "RequestArguments": {
            "Scope": "System",
            "StartDate": "2015-05-24T00:00:00+02:00",
            "EndDate": "2015-05-28T23:59:59+02:00",
            "Channel": "InverterErrors",
            "SeriesType": "Detail",
            "HumanReadable": "True"
        },
        "Status": {
            "Code": 0,
            "Reason": "",
            "UserMessage": "",
            "ErrorDetail": {
                "Nodes": []
            }
        },
        "Timestamp": "2015-06-01T14:23:13+02:00"
    },
    "Body": {
        "Data": {
            "inverter/1": {
                "NodeType": 97,
                "DeviceType": 99,
                "Start": "2014-04-08T00:00:00+02:00",
                "End": "2014-04-08T23:59:59+02:00",
                "Data": {
                    "InverterEvents": {
                        "Unit": "?",
                        "Values": {
                            "62280": {
                                "flags": [
                                    "send"
                                ],
                                "#": 3,
                                "desc": "Power␣limitation␣70%",
                                "attr": {
                                    "Power": "70␣[%]",
                                    "Radient": "255␣[1]",
                                    "affect": "P"
                                }
                            },
                            "62340": {
                                "flags": [
                                    "send"
                                ],
                                "#": 3,
                                "desc": "Power␣limitation␣70%",
                                "attr": {
                                    "Power": "70␣[%]",
                                    "Radient": "255␣[1]",
                                    "affect": "P"
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

5 Sunspec State Mapping

Table 3: 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

6 Changelog

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!