Version: 1.0

EM1Data

The EM1Data component stores data from an energy meter. It uses `EM1Data` as RPC namespace and provides the methods:

- `EM1Data.SetConfig` to update the component's configuration
- `EM1Data.GetConfig` to obtain the component's configuration
- `EM1Data.GetStatus` to obtain the component's status
- `EM1Data.GetRecords` to get saved emeter data time intervals
- `EM1Data.GetData` to get saved emeter data values
- `EM1Data.DeleteAllData` to delete all saved data
- `EM1Data.ResetCounters` to zero the total counters
- `EM1Data.GetNetEnergies` to get net energies

EM1Data components are identified with `em1data:<id>` in objects containing multiple component payloads.

Methods:

EM1Data.SetConfig

Properties:

Property	Type	Description
`id`	*number*	Id of the EM1Data component instance
`config`	*object*	Configuration that the method takes

Find more about the config properties in config section

EM1Data.GetConfig

Properties:

Property	Type	Description
`id`	*number*	Id of the EM1Data component instance

Find the EM1Data.GetConfig response properties in config section

EM1Data.GetStatus

Parameters:

Property	Type	Description
`id`	*number*	Id of the EM1Data component instance

Find more about the status response properties in status section

EM1Data.GetRecords

Parameters:

Property	Type	Description
`id`	*number*	Id of the EM1Data component instance
`ts`	*number*	UNIX timestamp of the first interval. Used for selecting the next data chunk when the response is too large to fit in one call. Default is 0.

EM1Data.GetData

Parameters:

Property	Type	Description
`id`	*number*	Id of the EM1Data component instance
`ts`	*number*	UNIX timestamp of the first record. Any record with data having a timestamp between `ts` and `end_ts` will be retrieved.
`end_ts`	*number*	UNIX timestamp of the last record to get (if available). If the response is too big, it will be chunked. The default is to get all available records without limit.
`add_keys`	*boolean*	If false will not print the keys array in the response. The default is true.

INFO

The type of data structure returned is:

```
namespace EM1Data {

   interface DataKey {
      ts: number;
      period: string;
      values: number[][];
   }

   export interface GetDataResult {
      keys?: string[];
      data: DataKey[];
   }
}
```

Usually `EM1Data.data` will be an array containing a single object. There will be situations when the records will be interrupted (power loss) and then the array will contain more than one item.

EM1Data.DeleteAllData

Parameters:

Property	Type	Description
`id`	*number*	Id of the EM1Data component instance

- Deletes all the data stored on the device, and nullifies the perpetual counters for the specified component.

EM1Data.ResetCounters

Parameters:

Property	Type	Description
`id`	*number*	Id of the EM1Data component instance

- Reset the total counters for the specified component.

EM1Data.GetNetEnergies

Parameters:

Property	Type	Description
`id`	*number*	Id of the EM1Data component instance **Required**
`ts`	*number*	UNIX timestamp of the first record. It must align with the first second of the selected time granularity *. **Required**

Property	Type	Description
`end_ts`	*number*	UNIX timestamp of the last record to get (if available). Default is to get all available records, if the response is too big - it will be chunked. **Optional**
`period`	*number*	Period over which to accumulate energies, possible values are 300, 900, 1800, or 3600 **seconds**. **Required**
`add_keys`	*boolean*	If false will not print the keys array in the response. Default is true. **Optional**

INFO

Usually `EM1Data.GetNetEnergies.data` will be an array containing a single object. There will be situations when the records will be interrupted (power loss) and then the array will contain more than one item.

NOTE

* `ts` - must align with the first second of the selected time granularity.

The first record starts on: `December 4, 2024 8:02:00 AM GMT` (UNIX ts - 1733299320).

- The selected period is `period=1800` seconds (30 min). The closest round and multiple number is `December 4, 2024 8:00:00 AM GMT` (UNIX ts - 1733299200) however, because this record is not available, the closest available ts is `December 4, 2024 8:30:00 AM GMT` (UNIX ts - 1733301000). This means the request must state `ts=1733301000`.
- Time stamps not multiple of the selected period are not supported: The selected period is `period=300` seconds (5 min). `ts=1733299320`, `ts=1733299322`, `ts=1733299560`, `ts=1733299559`, `ts=1733299561` are all invalid time stamps.

Configuration

The `EM1Data` component doesn't have configuration options.

Status

The status of the EM1Data component contains information about the perpetual counters and possible errors.

Parameters:

Property	Type	Description
`id`	*number*	Id of the EM1Data component instance
`total_act_energy`	*number*	Total active energy, Wh
`total_act_ret_energy`	*number*	Total active returned energy, Wh
`errors`	*array of type string*	Error condition occurred. May contain `database_error` or `ct_type_not_set`, (shown if the error is present).

Notifications

Data

This notification is triggered whenever data is saved to the device flash. The format of the data is the same as EM1Data.GetData response. See the example below.

- `method`: "NotifyEvent"
- `params`:
  - `ts`: number, UNIX timestamp
  - `event`: `"data"`
  - `component`: id of the EM1Data component

- `data`: array of objects:
  - `ts`: UNIX timestamp
  - `period`: Period of aggregated data
  - `values`: array of type number, data values corresponding to `keys` array of GetData method

StatusChange

A `StatusChange` event of EM1Data is emitted when perpetual total active energy counters are saved to flash.

Modbus registers

Address	Type	Description
32300	uint32	Timestamp of the last update
32302	float	Total active energy, Wh
32304	float	Total active returned energy, Wh
32306	float	Lagging reactive energy, VARh
32308	float	Leading reactive energy, VARh
32310	float	Total active energy - perpetual count, Wh
32312	float	Total active returned energy - perpetual count, Wh
32314		6 registers reserved

If there are more than one EM1Data component on the device, every next component's register block start address is calculated by adding 20 to the start address of the previous component. For example, EM1Data:1 will start at 32320.

CSV file download

Alternatively to the RPC method GetData, the same data can be downloaded in CSV file by accessing a URL:

http://`<device ip>`/em1data/`<id>`/data.csv

Optional HTTP parameters `ts`, `end_ts`, and `add_keys` may be used the same way as in `EM1Data.GetData`. See the `example` below.

Examples

EM1Data.SetConfig example

EM1Data.SetConfig HTTP GET Request | **EM1Data.SetConfig Curl Request** | **EM1Data.SetConfig Mos Request**

```
http://192.168.33.1/rpc/EM1Data.SetConfig?id=0&config={}
```

Response

```
{
  "restart_required": false
}
```

EM1Data.GetConfig example

EM1Data.GetConfig HTTP GET Request | **EM1Data.GetConfig Curl Request** | **EM1Data.GetConfig Mos Request**

```
http://192.168.33.1/rpc/EM1Data.GetConfig?id=0
```

Response

```
{}
```

EM1Data.GetStatus example

EM1Data.GetStatus HTTP GET Request | EM1Data.GetStatus Curl Request | EM1Data.GetStatus Mos Request

```
http://192.168.33.1/rpc/EM1Data.GetStatus?id=0
```

Response

```
{
  "id": 0,
  "total_act_energy": 0,
  "total_act_ret_energy": 0
}
```

EM1Data.GetRecords example

EM1Data.GetRecords HTTP GET Request | EM1Data.GetRecords Curl Request | EM1Data.GetRecords Mos Request

```
http://192.168.33.1/rpc/EM1Data.GetRecords?id=0&ts=0
```

Response

```
{
  "data_blocks": [
    {
      "ts": 1657739460,
      "period": 60,
      "records": 1
    },
    {
      "ts": 180,
      "period": 60,
      "records": 1
    },
    {
      "ts": 1657739580,
      "period": 60,
```

```
      "records": 2
    }
  ]
}
```

EM1Data.GetData example

EM1Data.GetData HTTP GET Request | EM1Data.GetData Curl Request | EM1Data.GetData Mos Request

```
http://192.168.33.1/rpc/EM1Data.GetData?id=0&ts=1656356400&end_ts=1656356800
```

Response

```
{
  "keys": [
    "total_act_energy",
    "total_act_ret_energy",
    "lag_react_energy",
    "lead_react_energy",
    "max_act_power",
    "min_act_power",
    "max_aprt_power",
    "min_aprt_power",
    "max_voltage",
    "min_voltage",
    "avg_voltage",
    "max_current",
    "min_current",
    "avg_current"
  ],
  "data": [
    {
      "ts": 0,
      "period": 60,
      "values": [
        [
          0,
          0,
          0,
          0,
          0,
```

```
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0
        ],
        [
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0
        ]
      ]
    }
  ],
  "next_record_ts": 1656357000
}
```


INFO

Data units are: Wh, VAh, VARh, W, V, A

EM1Data.DeleteAllData example

EM1Data.DeleteAllData HTTP GET Request | EM1Data.DeleteAllData Curl Request | EM1Data.DeleteAllData Mos Request

```
http://192.168.33.1/rpc/EM1Data.DeleteAllData?id=0
```

Response

```
null
```

EM1Data.ResetCounters example

EM1Data.ResetCounters HTTP GET Request | EM1Data.ResetCounters Curl Request | EM1Data.ResetCounters Mos Request

```
http://192.168.33.1/rpc/EM1Data.ResetCounters?id=0
```

Response

```
null
```

EM1Data.GetNetEnergies example

EM1Data.GetNetEnergies HTTP GET Request | EM1Data.GetNetEnergies Curl Request | EM1Data.GetNetEnergies Mos Request

```
http://192.168.33.1/rpc/EM1Data.GetNetEnergies?id=0&ts=1720256400&period=300
```

Response

```
{
  "keys": [
    "net_act_energy"
  ],
  "data": [
    {
      "ts": 1720256400,
      "period": 300,
      "values": [
        [
          -150.5384
        ],
        [
```

```
          -151.1524
        ]
      ]
    }
  ],
  "next_record_ts": 1725276720
}
```

INFO

Data units are: Wh

Notifications example

- When data is saved to the database in the device flash memory:

Example:

Notify that new data is saved.

```
{
   "src": "shellyproem-f008d1d8b8b8",
   "dst": "user_1",
   "method": "NotifyEvent",
   "params": {
      "ts": 1631266595.43,
      "events": [
        {
          "component":"em1data:0",
          "id":0,
          "event":"data",
          "ts":1631266595.43,
          "data":[
              {
                "ts":0,
                "period":60,
                "values":[
                  [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
                ]
              }
          ],
        }
      ]
```

```
    }
}
```

CSV file download example

> Example 1:

Using curl GET to download CSV data with a header row.

```
curl -OJ http://192.168.33.1/em1data/0/data.csv?add_keys=true
```