Version: 1.0

EMData

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

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

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

Methods:

EMData.SetConfig

Properties:

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

Find more about the config properties in config section

EMData.GetConfig

Properties:

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

Find the EMData.GetConfig response properties in config section

EMdata.GetStatus

Parameters:

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

Find more about the status response properties in status section

EMdata.GetRecords

Parameters:

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

EMdata.GetData

Parameters:

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

INFO

Type of data structure returned is:

```
namespace EMData {

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

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

Usually `EMData.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.

EMdata.DeleteAllData

Parameters:

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

- Deletes all the data stored on the device, nullifies the perpetual counters.

EMData.ResetCounters

Parameters:

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

- Reset the total counters for the specified component.

EMData.GetNetEnergies

Parameters:

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

Property	Type	Description
`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 `EMData.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). Invalid time stamps - `ts=1733299320`, `ts=1733299322`, `ts=1733299560`, `ts=1733299559`, `ts=1733299561`.

Configuration

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

Status

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

Parameters:

Property	Type	Description
`id`	*number*	Id of the EMData component instance
`a_total_act_energy`	*number*	Total active energy on phase A, Wh
`a_total_act_ret_energy`	*number*	Total active returned energy on phase A, Wh
`b_total_act_energy`	*number*	Total active energy on phase B, Wh
`b_total_act_ret_energy`	*number*	Total active returned energy on phase B, Wh
`c_total_act_energy`	*number*	Total active energy on phase C, Wh
`c_total_act_ret_energy`	*number*	Total active returned energy on phase C, Wh
`total_act`	*number*	Total active energy on all phases, Wh
`total_act_ret`	*number*	Total active returned energy on all phases, 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 device flash. Format of the data is the same as EMData.GetData response. See the example below.

- `method`: "NotifyEvent"
- `params`:
  - `ts`: number, UNIX timestamp
  - `event`: `"data"`
  - `component`: id of the EMData 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 EMData is emitted when perpetual total active energy counters are saved to flash.

Modbus registers

Address	Type	Description
31160	uint32	Timestamp of the last update
31162	float	Total active energy accumulated for all phases - perpetual count, Wh
31164	float	Total active returned energy accumulated for all phases - perpetual count, Wh
31166		4 registers reserved
31170	float	Phase A total active energy, Wh
31172	float	Phase A fundamental active energy, Wh

Address	Type	Description
31174	float	Phase A total active returned energy, Wh
31176	float	Phase A fundamental active returned energy, Wh
31178	float	Phase A lagging reactive energy, VARh
31180	float	Phase A leading reactive energy, VARh
31182	float	Phase A total active energy - perpetual count, Wh
31184	float	Phase A total active returned energy - perpetual count, Wh
31186		4 registers reserved
31190	float	Phase B total active energy, Wh
31192	float	Phase B fundamental active energy, Wh
31194	float	Phase B total active returned energy, Wh
31196	float	Phase B fundamental active returned energy, Wh
31198	float	Phase B lagging reactive energy, VARh
31200	float	Phase B leading reactive energy, VARh
31202	float	Phase B total active energy - perpetual count, Wh
31204	float	Phase B total active returned energy - perpetual count, Wh
31206		4 registers reserved
31210	float	Phase C total active energy, Wh
31212	float	Phase C fundamental active energy, Wh
31214	float	Phase C total active returned energy, Wh

Address	Type	Description
31216	float	Phase C fundamental active returned energy, Wh
31218	float	Phase C lagging reactive energy, VARh
31220	float	Phase C leading reactive energy, VARh
31222	float	Phase C total active energy - perpetual count, Wh
31224	float	Phase C total active returned energy - perpetual count, Wh
31226		4 registers reserved

If there is a second EMData component on the device, its corresponding registers addresses are calculated by adding 70 to the address in the table above.

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>`*/emdata/*`<id>`*/data.csv*

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

Examples

EMData.SetConfig example

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

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

Response

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

EMData.GetConfig example

EMData.GetConfig HTTP GET Request | EMData.GetConfig Curl Request | EMData.GetConfig Mos Request

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

Response

```
{}
```

EMData.GetStatus example

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

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

Response

```
{
  "id": 0,
  "a_total_act_energy": 0,
  "a_total_act_ret_energy": 0,
  "b_total_act_energy": 0,
  "b_total_act_ret_energy": 0,
  "c_total_act_energy": 0,
  "c_total_act_ret_energy": 0,
  "total_act": 0,
```

```
  "total_act_ret": 0
}
```

EMData.GetRecords example

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

```
http://192.168.33.1/rpc/EMData.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
    }
  ]
}
```

EMData.GetData example

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

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

Response


```
{
  "keys": [
    "a_total_act_energy",
    "a_fund_act_energy",
    "a_total_act_ret_energy",
    "a_fund_act_ret_energy",
    "a_lag_react_energy",
    "a_lead_react_energy",
    "a_max_act_power",
    "a_min_act_power",
    "a_max_aprt_power",
    "a_min_aprt_power",
    "a_max_voltage",
    "a_min_voltage",
    "a_avg_voltage",
    "a_max_current",
    "a_min_current",
    "a_avg_current",
    "b_total_act_energy",
    "b_fund_act_energy",
    "b_total_act_ret_energy",
    "b_fund_act_ret_energy",
    "b_lag_react_energy",
    "b_lead_react_energy",
    "b_max_act_power",
    "b_min_act_power",
    "b_max_aprt_power",
    "b_min_aprt_power",
    "b_max_voltage",
    "b_min_voltage",
    "b_avg_voltage",
    "b_max_current",
    "b_min_current",
    "b_avg_current",
    "c_total_act_energy",
    "c_fund_act_energy",
    "c_total_act_ret_energy",
    "c_fund_act_ret_energy",
    "c_lag_react_energy",
    "c_lead_react_energy",
    "c_max_act_power",
    "c_min_act_power",
    "c_max_aprt_power",
    "c_min_aprt_power",
    "c_max_voltage",
```

```
      "c_min_voltage",
      "c_avg_voltage",
      "c_max_current",
      "c_min_current",
      "c_avg_current",
      "n_max_current",
      "n_min_current",
      "n_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,
            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,
          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,
          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; For n_isum_mismatch: 0/1

EMData.DeleteAllData example

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

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

Response

```
null
```

EMData.ResetCounters example

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

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

Response

```
null
```

EMData.GetNetEnergies example

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

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

Response

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

```
          -150.5615,
          -239.8302
        ],
        [
          -150.5172,
          -151.1119,
          -238.9687
        ]
      ]
    }
  ],
  "next_record_ts": 1725276720
}
```


INFO

Data units are: Wh

Notifications example

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

Example:

Notify that new data is saved.


```
{
   "src": "shellypro3em-f008d1d8b8b8",
   "dst": "user_1",
   "method": "NotifyEvent",
   "params": {
      "ts": 1631266595.43,
      "events": [
        {
          "component":"emdata:0",
          "id":0,
          "event":"data",
          "ts":1631266595.43,
          "data":[
              {
                "ts":0,
                "period":60,
                "values":[
```

```
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                ]
              }
          ],
        }
      ]
    }
  }
```


CSV file download example

Example 1:

Using curl GET to download CSV data with header row.

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