Home Energy Market Simulator: The Scenario Definition Format

Andrea Monacchi [*], Sergii Zhevzhyk [†]
Institute of Networked Embedded Systems,
Alpen-Adria-Universität Klagenfurt, Austria

October 17, 2014[‡]

Abstract

A scenario in the HEMS simulator models the environment in which energy prosumers trade their energy. This document describes the format used to describe scenarios in the HEMS simulator. We show how to model generators, household appliances and weather conditions, as well as how to import time series resulting from measurements.

1 Writing a scenario

A scenario in HEMS consists in a weather model, as well as the specification of connections with the power grid and a model of the traders.

Code 1: Scenario format

```
{
 "weather":{},
 "grid_connections": [],
 "producers":[],
 "loads":[]
}
```

2 Modeling weather conditions

The first way of modelling weather conditions is to define time intervals. In the code reported in 2, the start and end time define the bounds of the intervals, modelled as $[start, end)$ using the format $[Month, Day, Hour, Minutes, Seconds]$. Accepted values are respectively, 1 to 12 for the month, 1 to 31 for the day, 0 to 23 for the hour, 0 to 59

[*]Supported by Lakeside Labs, Klagenfurt, Austria and funded by the European Regional Development Fund (ERDF) and the Carinthian Economic Promotion Fund (KWF) under grant 20214-23743-35469 and 35470 (project MONERGY http://monergy-project.eu)

[†]Supported by Lakeside Labs, Klagenfurt, Austria and funded by the ERDF and the KWF under grant 20214-22935-34445 (project Smart Microgrid).

[‡]First drafted on September 1st 2014

for minutes and seconds. Mistaken interval definitions, which do not overlap and leave out time portions will return a cloud factor of 1. The user can nevertheless change this behaviour to return an exception informing the user of the error.

Code 2: Weather as time intervals

```
{
  "type":"time-intervals",
  "intervals": [{"start":[11,21,0,0,0],"end":[3,21,0,0,0], "cloud-factor":0.6},
                {"start":[3,21,0,0,0],"end":[6,21,0,0,0], "cloud-factor":0.4},
                {"start":[6,21,0,0,0],"end":[9,21,0,0,0], "cloud-factor":0.2},
                {"start":[9,21,0,0,0],"end":[11,21,0,0,0], "cloud-factor":0.9}
  ]
}
```

The alternative is to refer to an external timeserie, using the format in 3.

Code 3: Weather as external timeserie

```
{
  "type":"external-timeserie",
  "position": "absolute/path/to/timeserie"
}
```

2.1 Modeling timeseries

A time series is a sequence of data points, measured typically at successive points in time spaced at uniform time intervals[1]. As we wanted to minimise the storage space needed to store time points, we opted for an event-based or edge-based time serie representation, which means, we only model changes in the represented value. The listing 4 shows how to model a timeserie. The first field is used to specify the beginning of the serie, expressed as seconds from the unix epoch beginning. The HEMS parser also accepts a date defined in the string format "yyyy MM dd HH:mm:ss". Consequently, the presented example is the same as writing "beginning": "2013 01 01 07:51:00". The designer should be careful that the date coincides or is in the past with respect to the beginning of the simulation time, as otherwise the value returned for time instants earlier than the beginning will be the one of the first interval. The user might however decide to change this behaviour so as to return 0 or an exception.

Code 4: Modeling a timeserie

```
{
"beginning": 1357023060,
"data": [
        [1,10],[250,50],[3,3600]
    ]
}
```

In the data field each element of the list represents a time interval, with the first element as the value and the second element as the interval duration in seconds. The example models the case of a 10 seconds interval in which the value is held to 1, a second interval in which the value 250 is held for 50 seconds and a third interval in which the value 3 is held for 3600 seconds.

[1] http://en.wikipedia.org/wiki/Time_series

3 Specifing grid connections

A connection to the power grid represents a smart meter or any other device metering and providing energy on demand at a specific price.

Code 5: Format to define grid connections

```
[
  {"name":"Grid_connection_1", "credit":10.0, "price-model": { }, "capability-model": { } },
  ...
  {"name":"Grid_connection_n", "credit":10.0, "price-model": { }, "capability-model": { } },
]
```

A grid connection is defined by a price model, describing the price sensitivity and the reservation price of the trader. The format of a price model is described in Sect. 3.1.

Code 6: Format to define the price model of a grid connection

```
{
  "tariff" : {},
  "feed-in" : {}
}
```

A capability model describes the available power at a given time, as well as the power that can be fed into the connection at a specific time.

Code 7: Format to define the capability model of a grid connection

```
{
  "power-availability" : {},
  "power-capability" : {}
}
```

The format of a price capability model is defined in Sect. 3.2.

3.1 Defining a price model

A price model can be used to model the price sensitivity and the reservation price of traders. It is important to remark that prices can be specified in any value, although they are considered per Kwh, as it usually is to define tariff plans. Traders handle hourly energy prices, while they get charged for allocated power as transaction $t \in \theta$ at each time instant, under the simple equation: $cost = t_{price} * t_{amount}/(1000 * 3600)$.

The simplest model is the constant price model as shown in 8.

Code 8: Format to define a constant price model

```
double
```

A more realistic solution is to set time intervals like $[start, end)$, such as tariff plans (see 9).

Code 9: Format to define a tariff-based price model

```
{
  "type": "time-intervals",
  "intervals" : [
      {"name":"day", "start":[6,0], "end":[20,0], "cost":0.50},
      {"name":"night", "start":[20,0], "end":[6,0], "cost":0.30}
   ]
}
```

Nevertheless the future Smart Grid will provide continuous interaction with energy producers and consumers to foster their reaction to the availability of energy. The possibility to load external time series modelling continuously changing price preferences is therefore necessary (See 10).

Code 10: Format to define a timeserie-based price model

```
{
  "type":"external-timeserie",
  "position": "path-to-timeserie"
}
```

3.2 Defining a capability model

Code 11: Format to define an interval-based capability model

```
{
  "type": "time-intervals",
  "intervals" : [
      {"start":[0,0], "end":[23,59], "amount":3000}
  ]
}
```

Code 12: Format to define a timeserie-based capability model

```
{
  "type":"external-timeserie",
  "position": "path-to-timeserie"
}
```

4 Modeling energy generators

An energy generator is a trading agent with an empty operation model and a generation model, which means, the device is only allowed to sell energy based on the generation model. The listing 13 shows the structure of a generator. Required fields for the model are: a name, a price model and a generator model type. Beside required parameters, additional optional parameters can be specified, such as the idle power to run the smart controller, an initial credit available to the agent (default 0) and the type of service provided (flexible or inflexible).

Code 13: Format to define an energy generator

```
[
  {"name":"Photovoltaic_house_1",
  "idle": int, "credit": double, "deferrable":boolean,
  "price-model":{},
  "type": String,
  "payload":{} }
  ...
  {"name":"Photovoltaic_house_1", "price-model":{}, "type": , "payload":{} }
]
```

The price model describes the reservation price model for the agent, and should respect the format described in Sect.3.1. In the payload, additional information are required, depending on the selected model type. At the time of writing the parser supports the following types of model: a *MODELED-PV*, a *MEASURED-PV*, a *MODELED-WINDGEN*, a *MEASURED-WINDGEN*. The measured models rely on measurement

data, which can be described using either intervals or external time series, as in Sec. 3.2. The modelled photovoltaic plant uses an internal component to calculate the produced power based on the size in sqm and the efficiency of the plant, as well as its position (See listing 14). The model was taken from the RAPSim simulator described in [PKE14, PSE13].

Code 14: Example model of a 3.270 KWp photovoltaic plant

```
{
  "peakPower":3270.0,
  "efficiency":0.15,
  "latitude":46.6,
  "longitude":14.4,
  "height":0.446,
  "size":50.0
}
```

The modelled wind generator will be supported soon.

5 Modeling household appliances

An household appliance is an electrical load, which operates based on a usage model and has no generation model. An appliance is described by a name, an initial credit used to buy energy to operate, the power needed to supply the smart controller used for trading, an hard deadline in seconds after which a first offer is considered expired (needed to maintain rationality), a type distinguishing different usage models, as well as a boolean selector determining the flexibility of the service with respect to time.

Code 15: Example definition for household appliances

```
[
  {"name":"Dishwasher",
   "credit":10.0,
   "idle":0,
   "offer_expiration":3600,
   "type":"RANDOM",
   "deferrable":true,
   "payload":{"power":[[800,5,30],[200,5,30],[400,600,2],[600,120,10],[200,5,10]],
              "willingness":0.6, "willingness_decay":0.6, "sensitivity":0.5}},
  ...
  {"name":"Light_bedroom",
   "credit":5.0,
   "idle":0,
   "offer_expiration":3600,
   "type":"RANDOM",
   "deferrable":false,
   "payload":{"power":[[60,120]],
              "willingness":0.6, "willingness_decay":0.5}}
]
```

In the example code listing in 15 it is shown a flexible and an inflexible device. The dishwasher trades based on a sensitivity price model (constant in the example) and a usage model. The bedroom light is an inflexible device, whose price sensitivity is automatically set to the market limit price and the tolerated start delay is 0. Another aspect of flexible devices its the discomfort they deliver to users upon shifting their operation over time, as well as the criticality resulting from delaying the operation of intermediate states. This value is modelled in the power profile (i.e., $[P_p, d, t_{ds}]$), where we distinguish for each state the peak power needed to operate, the duration

in seconds, and the tolerance to a delayed start. The latter value affect user's comfort for the first state, while it is a matter of the correct execution of the device for the next states. Therefore, an inflexible device implicitly intolerant to a delayed start (i.e., $t_{ds} = 0$).

5.1 Usage models

A usage model models a timeofuse probability, which is the probability of a user to operate an appliance at a specific time of the day. This can be expressed through a willingness value $\omega^* \in [0, 1]$, which can be defined statically either for the whole simulation time or for time intervals (e.g., hourly, quarterly). This probability can be extracted from collected consumption data at appliance level as in [SGB12]. The probability P_{use} to have a transition from OFF to ON within a time interval of length N is given by $P_{use} = 1 - (P_{hold})^N$, which translates into $P_{use} = 1 - (1 - \omega^*)^N$ and consequently $(1 - \omega^*)^N = 1 - P_{use}$ and $\omega^* = 1 - \sqrt[N]{1 - P_{use}}$. A willingness decay $\lambda \in \mathbb{R}$ can be used to update the probability to run for the current time interval (e.g., the same hour) as $P_{use} = P_{use}(1 - \lambda)^n$, with $n \in \mathbb{N}$ number of operations for the current interval [MZE14].

In the HEMS, for each device we distinguish devices for their usage model, we have accordingly *RANDOM* devices and *MODELED* devices. A *RANDOM* device has a static usage model, in which a willingness value is used to model its willingness to start (and therefore make a first offer). A decay factor is then used to update the willingness value after each operation, so that a next operation can be less or more likely to occur than the previous one. The device willingness is then reset at each hour. To overcome this simple model, HEMS also supports *MODELED* devices. Such a device type can either define an array of willingness and decay factors for different times of the day. We speak in this case of a *LOOKUP* usage model. Alternatively, we speak of *MODEL* when an external usage model learned from actual usage data is referred. As a matter of fact, the simple decay shown can not model situations such as "two washing machine operations in a row might be more common than just one, although three consecutive ones are very unlikely to occur". Therefore, we suggest the use of the latter to model device demand. While we support both model types, we plan to deprecate the *LOOKUP* model type in future. Luckily, more sophisticated usage models based on machine learning and data mining techniques can be learned from consumption dataset. For instance, we showed in [MEE+14] how a Bayesian network can be used to mine the usage of appliances in Austrian and Italian households.

In the scenario description, the selection of the usage model type is done in the payload (See the code listing 16).

Code 16: Payload format for household appliances

```
{
  "profile-type" : "LOOKUP" | "MODEL",
  "profile-position" : "absolute/path/to/profile",
}
```

As for the *LOOKUP* usage model, the imported model specifies the power profiles, as well as the usage model and a willingness decay over different times of the day or week. The price sensitivity model is required for flexible devices and unnecessary for inflexible ones.

Code 17: Format of a LOOKUP usage model

```
{
 "power": [],
 "usage": [],
 "willingness_decay" : [],
 "sensitivity" : {}
}
```

A clear requirement is that the usage array and its associated willingness decay array have same size. Accepted array dimensions are: 24 (hourly-daily), 96 (quarterly-daily), 168 (hourly-daily), 672 (quarterly-daily) and 8760 (hourly-yearly).

Concerning the importation of an external usage model based on data mining and machine learning techniques, the payload used to describe the device specifies also the format of the model (e.g., bayesian network, neural network). The format is firstly checked in a central model manager so that a specific plug-in can be called to handle the given model.

Code 18: Format of a MODEL usage model

```
{
  "power": [],
  "sensitivity":{},
  "profile_type" : "MODEL",
  "profile_position" : "absolute/path/to/profile",
  "format" :  String
}
```

6 Complete example scenario

Code 19: Example scenario

```
{
  "weather":{"type":"time-intervals",
                  "intervals": [{"start":[11,21,0,0,0],"end":[3,21,0,0,0], "cloud-factor":0.6},
                                {"start":[3,21,0,0,0],"end":[3,21,0,0,0], "cloud-factor":0.4},
                                {"start":[6,21,0,0,0],"end":[9,21,0,0,0], "cloud-factor":0.2},
                                {"start":[9,21,0,0,0],"end":[11,21,0,0,0], "cloud-factor":0.9}]
  },
  "grid_connections":[
    {"name":"Grid_connection_1", "credit":10.0, "price-model": {
                                                        "tariff": {"type": "time-intervals",
                                                                   "intervals" : [{"name":"day", "start":[6,0], "end":[20,0], "cost":0.50},
                                                                                  {"name":"night", "start":[20,0], "end":[6,0], "cost":0.30}]
                                                                  },
                                                        "feed-in":{ "type": "time-intervals",
                                                                   "intervals" : [{"name":"day", "start":[6,0], "end":[20,0], "cost":0.20},
                                                                                  {"name":"night", "start":[20,0], "end":[6,0], "cost":0.05}]
                                                                  }
                                                        },
                                                "capability-model": {
                                                         "power-availability": {"type": "time-intervals",
                                                                                "intervals" : [{"start":[0,0], "end":[23,59], "amount":3000}]},
                                                         "power-capability": {"type" : "time-intervals",
                                                                                "intervals" : [{"start":[0,0], "end":[23,59], "amount":3000}]}
                                                                  }
                                                        }
  ],
  "producers":[
   {"name":"Photovoltaic_house_1", "idle":0, "price-model":0.1, "type":"MODELED-PV",
       "payload":{"peakPower":3270.0, "efficiency":0.15, "latitude":46.6, "longitude":14.4, "height":0.446, "size":50.0} },
  ],
  "loads":[
   {"name":"Dishwasher", "credit":10.0, "idle":0, "offer_expiration":3600, "type":"RANDOM", "deferrable":true,
       "payload":{"power":[[800,5,30],[200,5,30],[400,600,2],[600,120,10],[200,5,10]],
                  "willingness":0.6, "willingness_decay":0.6, "sensitivity":0.5}},
   {"name":"Fridge", "credit":10.0, "idle":0, "offer_expiration":3600, "type":"RANDOM", "deferrable":true,
       "payload":{"power":[[200,5,30]],
                  "willingness":0.8, "willingness_decay":0.1, "sensitivity":0.5}},
   {"name":"Boiler", "credit":10.0, "idle":0, "offer_expiration":3600, "type":"RANDOM", "deferrable":true,
       "payload":{"power":[[800,5,30],[800,5,30],[800,5,30],[800,5,30]],
                  "willingness":0.5, "willingness_decay":0.5, "sensitivity":0.3}},
   {"name":"Washingmachine_1", "credit":10.0, "idle":0, "offer_expiration":3600, "type":"RANDOM", "deferrable":true,
```

```
        "payload":{"power":[[800,120,30],[200,5,30],[400,10,2],[400,10,2],[400,10,2],[400,10,2],[400,10,2],[400,10,2],[400,10,2],[400,10,2],
                           [400,10,2],[400,10,2],[700,120,10],[200,2,10]],
                    "willingness":0.6, "willingness_decay":0.5, "sensitivity":0.5}},
    {"name":"Washingmachine_2", "credit":10.0, "idle":0, "offer_expiration":3600, "type":"RANDOM", "deferrable":true,
        "payload":{"power":[[800,120,30],[200,5,30],[400,10,2],[400,10,2],[400,10,2],[400,10,2],[400,10,2],[400,10,2],[400,10,2],[400,10,2],
                           [400,10,2],[400,10,2],[700,120,10],[200,2,10]],
                    "willingness":0.6, "willingness_decay":0.5, "sensitivity":0.5}},
    {"name":"Light_bedroom", "credit":5.0, "idle":0, "offer_expiration":3600, "type":"RANDOM", "deferrable":false,
        "payload":{"power":[[60,120]],
                    "sensitivity":0.5, "willingness":0.6, "willingness_decay":0.5}}
]
}
```

References

[MEE+14] Andrea Monacchi, Dominik Egarter, Wilfried Elmenreich, Salvatore D'Alessandro, and Andrea M. Tonello. GREEND: an energy consumption dataset of households in Italy and Austria. In *Proc. of IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Venice, Italy, Nov 2014.

[MZE14] Andrea Monacchi, Sergii Zhevzhyk, and Wilfried Elmenreich. HEMS: A home energy market simulator. In *Proc. of EnergieInformatik 2014 (to appear)*, Zurich, Switzerland, Nov 2014.

[PKE14] Manfred Pöchacker, Tamer Khatib, and Wilfried Elmenreich. The microgrid simulation tool RAPSim: description and case study. In *Proc. of the 2014 IEEE Innovative Smart Grid Technologies (ISGT) Conference - Asia*, Kuala Lumpur, Malaysia, May 2014.

[PSE13] M. Pöchacker, A. Sobe, and W. Elmenreich. Simulating the smart grid. In *IEEE PowerTech*, Grenoble, France, June 2013.

[SGB12] D.P. Seetharam, T. Ganu, and J. Basak. Sepia - a self-organizing electricity pricing system. In *IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*, pages 1–6, Tianjin, China, May 2012.