FlexOffer specifications

Introduction

FlexOffer (FO) is a representation of energy flexibility, which has the characteristics of i) being device-independent, ii) modeling flexibility with high accuracy, and iii) being scalable with respect to long time horizons and aggregation of many devices. This work has the purpose of defining the specifications for FOs in a general context. FlexOffer (FO) is a representation of energy product, which beside the trading market products covers also the energy flexibility. It has the characteristics of the minimum and maximum available amount of energy for consumption and production. FOs consider time as discrete, and divided in regular intervals called time slices. The duration of a time slice is usually 15 minutes, and is defined in the parameter numSecondsPerInterval. The FO concept was first proposed in the MIRABEL project, further developed in the TOTALFLEX project and demonstrated at large scale on the GOFLEX project. It is moreover demonstrated in FEVER, GIFT, edgeFlex, domos, LeapRe projects. A single FO typically includes: * Energy profile, having a number of discrete slices, specifies electricity consumption and production options over a device's active period of operation; * Time flexibility interval specifies a time period in which device's operation (profile) can be advanced or retarded. * Default profile specifies a preferred / locally optimal consumption profile (a baseload) * Price data specifies (discomfort) prices, e.g., associated to deviations from the default profile. Additionally the FO protocol supports the energy flow description of energy reservoirs like batteries, EVs and others with the advanced parameters describing total, dependency and uncertain constraints. This document is organized as follows. Chapter 1 describes how the FO protocol works, the actors involved in creation and management of FOs, the processes FOs go through, and the life cycle of an FO. Chapter 2 describes different types of FOs that can be generated and the energy constraints defining them. It is important to note that FlexOffers represent energy and not power: it is a convention decided because all slices have a duration. It can be converted to average power per slice if needed. Moreover, the metering is done on the energy.

Chapter 1: FlexOffer protocol

1.1 Flexibility and main actors

This work describes FOs, which are a representation for energy flexibility. We describe energy flexibility as the capability to change the time and amount of energy consumption from a grid actor. Flexibility represented by FOs can go through multiple processes: it can be optimized, aggregated and traded. We now describe all the actors that can be involved in these processes. * Prosumer: owns the flexible resources that deliver flexibility. FOs are generated at prosumer level by an automatic agent, and the same agent will execute FO schedules once they are received back by the prosumer. * Aggregator: collects FOs from prosumers. An aggregator is capable of aggregating, optimizing and

disaggregating FOs, and can also sell FOs to the flexibility market. * TSO: short for Transmission System Operator, it is responsible for operating, maintening and developing the transmission system in a given area. It is a buyer on the flexibility market. * DSO: short for Distribution System Operator, it is the grid operator with the responsbility for medium to low voltage power distribution in a geographical area. It is a buyer on the flexibility market. * BRP: short for Balance Responsible Party, it has the responsibility to secure balance between energy generation and consumption in a geographical area. It is a buyer on the flexibility market.

1.2 Flexibility operations

As mentioned in Section 1.1, flexibility represented by FOs can undergo several processes. This section will describe them in further detail.

1.2.1 Optimization Flexibility can be used for optimization towards several objectives, such as minimizing energy costs or CO2 emissions, maximizing renewable energy consumption, peak shaving, matching demand for power with supply (demand response), ancillary services and avoiding local grid congestions. In some cases, those objectives are related, e.g. energy prices may vary in order to encourage demand-response. Flexibility also enables prosumers to participate to spot (day-ahead, intraday) and balancing energy markets, either on their own or joining their flexibility with other prosumers. An FO is described by a set of constraints over energy variables within a given time horizon, that is, the number of time slices that the FO is considering. Optimizing an FO means creating an objective function over the energy and time variables, and finding the minimum of that function by optimization, which constraints over the energy variables are given by the FO.

1.2.2 Aggregation Prosumers who can only offer small amounts of flexibility may be unable to participate in the energy market alone, as they would not meet the threshold for the minimum bid size, which for example is 5MW in Switzerland. They can, however, do so by combining together their flexibility: this process is called aggregation. An important property of FOs is that they can be aggregated. This means that given a number N of FOs, it is possible to generate M ? N FOs which together represent the combined flexibility of the original N FOs, with some losses. Different types of FOs have different aggregation mechanisms, which however are all backwards-compatible: in other words, FOs of different types can always be aggregated, provided that aggregation makes sense in the first place (i.e. they refer to the same time span).

1.2.3 Disaggregation The importance of the aggregation process has been described in the previous subsection. Aggregated flexibility can be traded as if it belongs to one single entity. After that, its assignment can be dispatched to each user: this process is called disaggregation. Disaggregation can be seen as the

inverse process of aggregation; however, there is a specific conceptual difference. Aggregation collects many representations of flexibility and converts them into a small number of representations of flexibility; disaggregation converts a schedule for energy consumption into many schedules of energy consumption.

1.2.4 Flexibility trading It is also possible to trade flexibility in the respective market.

After Flex-Offer generation at the Prosumer-side, the Flex-Offer is typically sent to a receiving party, potentially, some utility company, BRP, or Aggregator, where is takes part in flexibility negotiation, planning, control, and billing processes, shown in Figure 1.1.

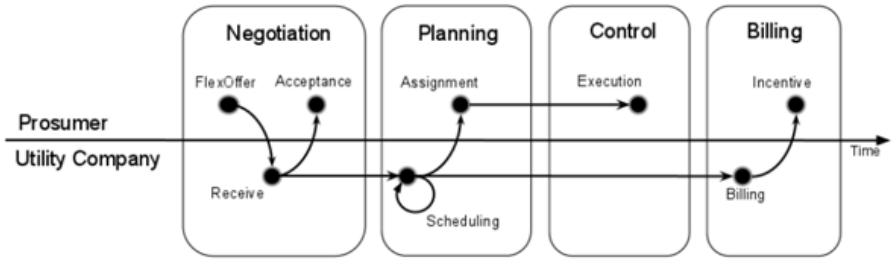


Figure 1.1: A schematic description of the FO life-cycle.

- Negotiation process The Flex-Offer can be accepted, e.g., if all of its attributes are valid and offered flexibility is valuable for the receiving party. On the other hand, the Flex-Offer can be rejected, e.g., due to some validation errors, or due to unacceptable price or energy, which then requires updating and resending the Flex-Offer or operating Prosumer processes under the default profile (baseload), e.g. not using the Flex-Offer.
- Planning process As mentioned earlier, the Flex-Offer can be decomposed into a number of decision variables and constraints, an used in actor-specific optimization and planning process. This results into one or more Flex-Offer schedules, i.e., assignments, which respect all Flex-Offer constraints and can be executed by the Prosumer.
- Control process Each Flex-Offer schedule (assignment) sent to a Prosumer is executed, starting at the given starting time, such that prescribed energy amounts are consumed or produced at subsequent time slices.
- Billing process Prosumer is rewarded by the flex-offer receiving party for its offered flexibility (Flex-Offers). Typical message exchange between the Prosumer and Flex-Offer receiving party, covering the negotiation and planning processes, is presented below.

1.3 FlexOffer life cycle

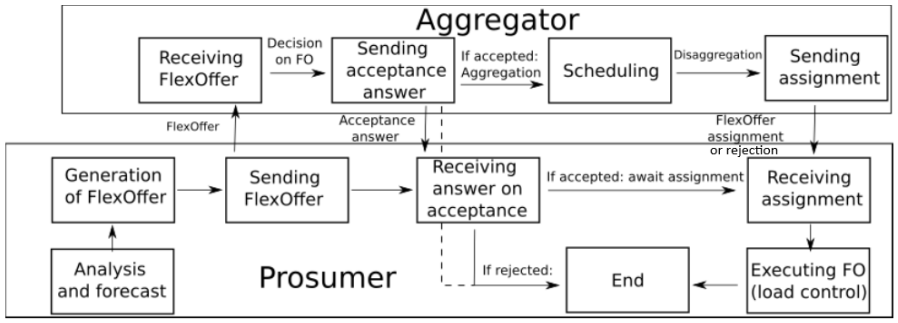


Figure 1.2: A schematic description of the FO life-cycle.

Figure 1.2 shows the life-cycle of an FO. Two main parties are involved: the prosumer, who generates and executes the FO, and the aggregator, who processes and issues schedules for the FO. The tasks on the prosumer's side are performed automatically by an agent, which operates according to the prosumer's requirements. First, the prosumer agent forecasts flexibility for the devices, and generates FOs according to that. Each FO is then sent to the aggregator, which will determine if the FO is useful for its needs, decides whether to accept the FO or to reject it (e.g., due to some validation errors, or due to unacceptable price or energy) and informs the prosumer of the response. If the FO is not accepted, it is not executed and the cycle ends here. Otherwise, the aggregator processes it (e.g. aggregating it with other FOs, performing optimization), and establishes a schedule for each FO. FO schedules are then sent back to the prosumer agent, which will execute them by controlling the devices.

The scheduling consists of the assignment deadline control (provided by flexibility manager - FMAN - component) and flex-offer matching (provided by flexibility market - FMAR - component). The FMAR component provides price based optimisation at combining the production with consumption flex-offers. If the matching could not be found till the assignment deadline then the flex-offer is rejected and no schedule is generated.

1.4 FlexOffer message

This section has the purpose to show how a JSON message for an FO is made, and which attributes are considered for it. This message is the core of the FO protocol. It is exchanged between the FO issuer and the FO receiving party. A range of optional attributes can be used to give indications on constraints and to be used in the different steps of the flexibility trading process. Depending on the attributes used, it can therefore be used to * Offer a flexibility bid * Accept or refuse a flexibility offer * Assign a flexibility It offers a common representation of all flexibilities, based on time slices and optional constraints. FOs define energy flexibility by specifying lower and upper bounds for energy, which identify the

minimum and maximum amount of energy that can be produced/consumed at the considered time slice. Positive values indicate energy production, negative values indicate energy consumption.

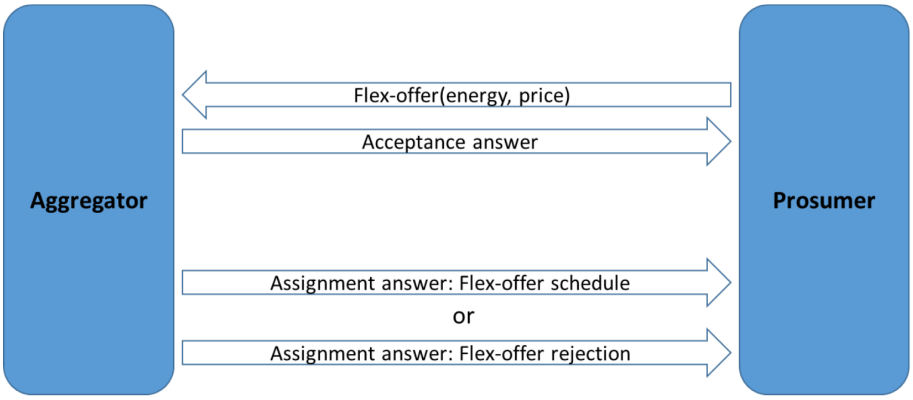


Figure 1.4: Messages exchange process

This is an example of a FO message (request):


```
{
  "flexOffer": [
    {
      "id": "17",
      "state": "offered",
      "stateReason": "initial offer",
      "creationTime": "2017-01-22T05:00:00Z",
      "offeredById": "1023",
      "acceptBeforeTime": "2017-01-22T07:45:00Z",
      "assignmentBeforeTime": "2017-01-22T22:45:00Z",
      "startAfterTime": "2017-01-22T09:00:00Z",
      "startBeforeTime": "2017-02-11T22:00:00Z",
      "numSecondsPerInterval": 900,
      "flexOfferProfileConstraints": [
        {
          "minDuration": 1,
          "maxDuration": 1,
          "energyConstraintList": [
            {
              "lowerBound": -5.1,
              "upperBound": -16.89
            }
          ],
          "tariffConstraint": {
            "minTariff": 0.03,
            "maxTariff": 0.03
```

```
    }
  },
  {
    "minDuration": 1,
    "maxDuration": 1,
    "energyConstraintList": [
      {
        "lowerBound": -5.1,
        "upperBound": 6.89
      }
    ],
    "tariffConstraint": {
      "minTariff": 0.03,
      "maxTariff": 0.15
    }
  },
  {
    "minDuration": 1,
    "maxDuration": 1,
    "energyConstraintList": [
      {
        "lowerBound": 3.14,
        "upperBound": 3.14
      }
    ],
    "tariffConstraint": {
      "minTariff": 0.03,
      "maxTariff": 0.03
    }
  },
  {
    "minDuration": 1,
    "maxDuration": 1,
    "energyConstraintList": [
      {
        "lowerBound": 11.89,
        "upperBound": 11.89
      },
      {
        "lowerBound": 2.1,
        "upperBound": 6.89
      },
      {
        "lowerBound": 2.1,
        "upperBound": 6.89
      }
```

```
          ],
          "tariffConstraint": {
            "minTariff": 0.15,
            "maxTariff": 0.15
          }
        }
      ],
      "defaultSchedule": {
        "startTime": "2017-01-22T22:45:00Z",
        "scheduleSlices": [
          {
            "duration": 1,
            "energyAmount": 0,
            "tariff": 1
          },
          {
            "duration": 1,
            "energyAmount": 0,
            "tariff": 1
          }
        ]
      }
    }
  ]
}
```


This is an example of a FO message (response):


```
{
  "flexOffer": [
    {
      "id": "123",
      "state": "assigned",
      "stateReason": "assigned",
      "creationTime": "2023-03-28T10:36:52Z",
      "offeredById": "201",
      "internalId": "14561741",
      "flexOfferSchedule": {
        "startTime": "2023-03-30T18:00:00Z",
        "numSecondsPerInterval": 900,
        "scheduleSlices": [
          {
            "duration": 1,
            "energyAmount": "-13342.610307504",
            "tariff": 0.158
```

```
          },
          {
            "duration": 1,
            "energyAmount": "-14330.47291966",
            "tariff": 0.0945
          },
          {
            "duration": 1,
            "energyAmount": "-15634.3049937015",
            "tariff": 0.111
          },
          {
            "duration": 1,
            "energyAmount": "-16754.1480817855",
            "tariff": 0.111
          }
        ]
      }
    }
  ]
}
```


In the following table, the attributes included in an FO message are listed.

Attribute	Mandatory	Type	Description
id	Yes	String	The ID that identifies the FO
state	Yes	String	State of the FO (initial (only until the offer is offered)/offered/accepted/rejected/assigned/executed
stateReason	No	String	Reason for FO state
numSecondsPerInterval	No	Int	Duration in seconds of a time slice (default value is 900)
creationTime	Yes	Datetime	Absolute time at which the FO has been created
creationInterval	No	Integer	FO creation Interval calculated as epoch value for creationTime/numSecondsPerInterval

Attribute	Mandatory	Type	Description
offeredById	Yes	String	ID of the FO owner
locationID	No	String	ID for representing the location of the FO in the grid system. If the parameter is not present, the location can be deducted through the offeredById parameter.

Attribute	Mandatory	Type	Description
acceptBeforeTime	No	Datetime	Absolute time before which FO with valid data must be accepted. Sets the deadline on when a flex-offer receiving party (e.g., BRP) should acknowledge successful acceptance or rejection of the flex-offer. A flex-offer rejection may occur if, e. g., flex-offer constraints or other metadata (e.g., prices) are invalid or inappropriate (e.g., quantities are too small, prices are too high). If the parameter is missing no acceptance response is generated (unless in the case of malformed message, when the response is returned immediately). An offer acceptance confirms the logically correctness of the flex-offer, respecting all constraints, and indicated that the offer may be assigned at a latter point in time. Depending on the use-case, there will be some business meaning to the acceptance state, defined in the contract.

Attribute	Mandatory	Type	Description
acceptBeforeInterval	No	Integer	Interval before which FO must be accepted.
assignmentBeforeTime	No	Datetime	Absolute time before which FO must be scheduled. Sets the deadlines on when flex-offer schedule update (assignment) is allowed to be sent by the flex-offer receiving party (BRP) to a flex-offer issuing party (flexible resource). In case the parameter is not present, the default value is startAfterTime.

Attribute	Mandatory	Type	Description
assignmentBeforeStart	No	Integer	May be used instead of assignmentBeforeTime. Sets the deadlines on when flex-offer schedule update (assignment) is allowed to be sent by the flex-offer receiving party (BRP) to a flex-offer issuing party (flexible resource). It is expressed relatively regarding the 'startAt' time in flex offer schedule. The value 'assignmentBeforeStart':300 means that the assignment message is sent latest 5 minutes before the schedule starts the execution. In case the parameter is not present its value is '0'.
assignmentBeforeInterval	No	Integer	Interval before which FO must be scheduled.

Attribute	Mandatory	Type	Description
startAfterTime	No	Datetime	Absolute time after which FO must be started.The range [startAfterTime, startBeforeTime] defines the time range within which the offer can be activated. In case it is not present, it is assumed to be equal to 'creationTime'
startAfterInterval	No	Integer	Interval after which FO must be started.
startBeforeTime	Yes	Datetime	Absolute time before which FO must be started.
startBeforeInterval	No	Integer	Interval before which FO must be started.
endAfterInterval	No	Integer	Interval after which FO execution must end. The parameters are used when time flexibility is being described (startAfterTime is not equal to startBeforeTime) and minDuration is not equal to maxDuration in FlexOfferSlice element

Attribute	Mandatory	Type	Description
endBeforeInterval	No	Integer	Interval before which FO execution must end. The parameters are used when time flexibility is being described (startAfterTime is not equal to startBeforeTime) and minDuration is not equal to maxDuration in FlexOfferSlice element
flexOfferProfileConstraints	Yes	array of flexOffer-Slice	Constraints for FO profile. A null value or an empty list means the flexibility removal.
endAfterTime	No	Datetime	Absolute time after which FO execution must end. The parameters are used when time flexibility is being described (startAfterTime is not equal to startBeforeTime) and minDuration is not equal to maxDuration in FlexOfferSlice element

Attribute	Mandatory	Type	Description
endBeforeTime	No	Datetime	Absolute time before which FO execution must end. The parameters are used when time flexibility is being described (startAfterTime is not equal to startBeforeTime) and minDuration is not equal to maxDuration in FlexOfferSlice element
flexOfferPriceConstraints	No	array of priceSlice	Constraints for FO price.
defaultSchedule	No	ScheduleSlice	Default energy consumption and time schedule of an FO.
powerFactorConstraint	No	List of parameters	Has two sub-elements: lower, and upper.The definition of the cos phi range of energy flexibility in the adapationPotential. It is defined as pair min, max and default. If not present it is assumed min = max = 1.0.
totalCostConstraint	No	List of parameters	Has two sub-elements: lower, and upper.
flexOfferProfilType	No	String	‘activeEnergy’(default)/‘reactiveEnergy’/‘voltage’
unit	No	String	‘Wh’(default)/‘VAh’/‘V’ – units of the energy constraint list

Attribute	Mandatory	Type	Description
multiplier	No	String	‘k’(default), ‘1’, ‘M’

Table 1.1: FO attributes and their descriptions.

Here are the descriptions of the different elements mentionned in the table above.

Attribute	Mandatory	Type	Description
lowerBound	Yes	Float	If positive: Minimal produced energy; If negative: Maximal consumed energy
upperBound	Yes	Float	If positive: Maximal produced energy; If negative: Minimal consumed energy

Table 1.2: energyConstraintsList element attributes and their descriptions.

Attribute	Mandatory	Type	Description
minprice	Yes	Float	Maximal price to be paid at consumption increase
maxprice	Yes	Float	Minimal price to be received at production increase

Table 1.3: priceConstraint element attributes and their descriptions.

Attribute	Mandatory	Type	Description
startTime	Yes	Date time	Start time of the the price slices time series
priceSlices	Yes	Array of priceSlice	See detailed description in a separate table

Table 1.4: priceConstraintsList element attributes and their descriptions.

Attribute	Mandatory	Type	Description
duration	No	Integer	Duration of the slice in nomber of intervals- If absent it is equal to 1
priceConstraint	Yes	Object	See detailed description above

Table 1.5: priceSlice element attributes and their descriptions.

Moreover, several constraints, that can be used to detail the offer, can be added to this message. They are described in the following chapter.

Chapter 2: FlexOffer constraints

Additionnal parameters can be added, as contraints to the flex offers: 'totalEnergyConstraint', 'subTotalEnergyConstraint' and 'DependencyEnergyConstraint'. This creates a few different types of flex offers.

2.1 Running example

In order to show how different types of FOs work, we will define a running example that will be used through the document. It is important to note that, through this document, we will use the following conventions: * FOs will represent energy. * Positive amounts of energy refer to energy consumed by the prosumer, negative amounts refer to energy obtained by the prosumer. For our running example, we will consider a Tesla Powerwall battery. Its capacity is 14 kWh, its maximum charging and discharging power are both 5 kW, and its round-trip efficiency is 90%. We use one hour time units, i.e. the battery can either be charged or discharged by an amount up to 5 kWh at each time unit. For describing the functioning of the battery, we use Coulomb counting [1]. At each time unit t, we write the state of charge (SoC) of the battery as

$$SoC(t) = SoC(t-1) + K \cdot u^{+}(t) + K^{-1} \cdot u^{-}(t)$$

$$SoC_{min} \leq SoC(t) \leq SoC_{max}; \frac{E_{min}}{K} \leq u(t) \leq E_{max}.$$

Figure 1: equation 2.1

Here, SoC(t) is the amount of energy in the battery at time t, expressed in kWh. u(t) is the amount of energy that the prosumer gives to/receives from the battery at time t, in kWh: u(t) is positive if the battery is being charged, negative otherwise. u + (t) is max{u(t),0}, u − (t) is min{u(t),0}. K is a real

number that measures how much energy is kept while charging/discharging the battery: it goes from 0 (all the energy is lost) to 1 (no energy is lost). SoC min and SoC max are the minimum and maximum state of charge that the battery can have in kWh, respectively. Lastly, E min and E max are the minimum and maximum amount of energy (in kWh) that can be taken from/given to the battery in one time unit.

We will consider two cases. In the first one (charging example), SoC(0) = 0 kWh and the battery can only be charged. In the second one (switching example), SoC(0) = 7 kWh and the battery can switch between charge and discharge at any time unit.

2.2 Energy slice constraint FO (SFO)

There are many types of constraints that have been used to define FOs. The most simple ones are start time constraints and slice (energy) constraints. A start time constraint determines the earliest and latest time unit at which the load can start. An energy constraint establishes, for each time unit at which the load is operating, the minimum and maximum amount of energy that can be consumed from that load. This means that for every time unit t, the energy constraint specifies a lower and an upper bound emin t and emax t such that emin t e t emax t . A standard FO (SFO) is an FO whose constraints are all slice constraints. Figure 2.1 shows an example of a slice FO (SFO) for the charging example. There are many possible ways of generating an SFO: in this case, we want to use all the flexibility available. At each time unit, a minimum of 0 kWh and a maximum of 5 kWh of energy can be used to charge the battery; this allows to employ as much flexibility as possible. However, from this representation, we may generate unfeasible configurations: for example, if 5 kWh are used at each time unit, 30 kWh will be given in total to the battery, but this is impossible since the maximum charge amount is 14 kWh.

Figure 2.1: Example of SFO

2.2.1 FlexOffer message Section 1.4 describes in detail how the FO message looks like. However, the attribute FlexOfferProfileConstraints has several sub-attributes, which depend on the type of FO that has been issued. In this section, we will describe them for an SFO. This is how the FlexOfferProfileConstraints attribute looks like for an SFO:


```
"flexOfferProfileConstraints": [ {
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
```

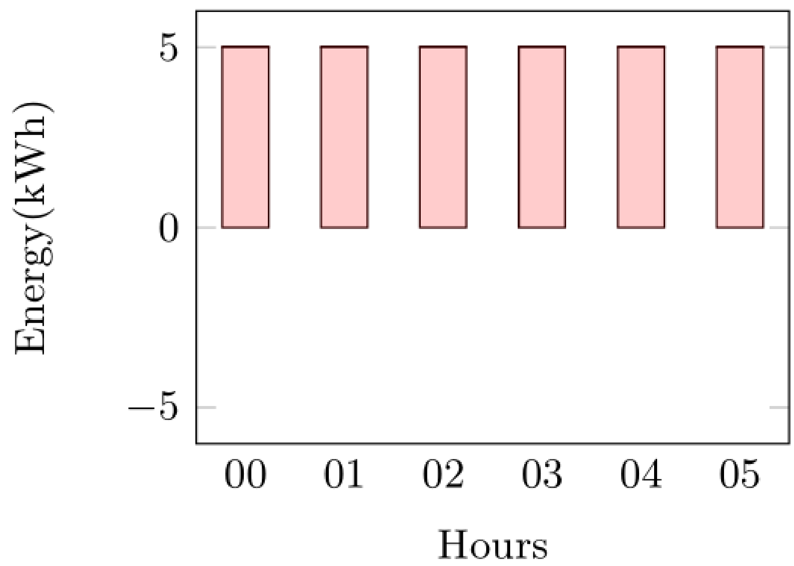


Figure 2: Example of SFO


```
 "minDuration": 1,
 "maxDuration": 1
},{
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
```

}]

Table 2.1 describes the sub-attributes of the flexOfferProfileConstraints more in detail. |Attribute |Mandatory| Type |Description| |———-|———-|—-|———-| |energyConstraintsList |Yes |array of 'energyConstraints' |Contains the list(s) of energy constraints for one time unit. Has two sub-elements: lower, and upper.| |priceConstraint| No |Object |List of price constraints. Has two elements: minPrice, and maxPrice.| |minDuration |No |Integer | Minimal slice duration in number of intervals| |maxDuration |No |Integer | Maximal slice duration in number of intervals|

Table 2.1: Sub-attributes of flexOfferSlice for SFOs.

2.2.2 Response schedule Data in ScheduleSlice format has the following sub-elements:

Attribute	Mandatory	Type	Description
Duration	No	Integer	Indicates the duration of the considered slices, in time units.
EnergyAmount	Yes	Float	Indicates the energy consumption for that slice. For every discrete interval of an active device operation, energy amount flexibility is characterized by a range
price	No	Float	Indicates the price amount for that slice.

Table 2.2: ScheduleSlice data.

An example for this data:

```
"schedule": {
 "scheduleId": 0,
 "updateId": 0,
 "scheduleSlices": [{"duration": 1, "energyAmount": 2, "price": 0.5},
  {"duration": 1, "energyAmount": 3, "price": 0.5}
```

```
  {"duration": 1, "energyAmount": 3, "price": 0.5}
  {"duration": 1, "energyAmount": 0, "price": 0}
  {"duration": 1, "energyAmount": 0, "price": 0}
  {"duration": 1, "energyAmount": 0, "price": 0}],
 "startTime": "2019-04-02T00:00:00.000+0000"
}
```


2.3 Total energy constraints FlexOffers

Another type of constraint is the total energy constraint (TEC), which specifies the lower (TE min) and upper (TE max) bounds for the energy that can be consumed over the considered time horizon. With the notation used before, this

$$TE_{min} \leq \sum_{t=1}^{T} e_t \leq TE_{max}.$$

means

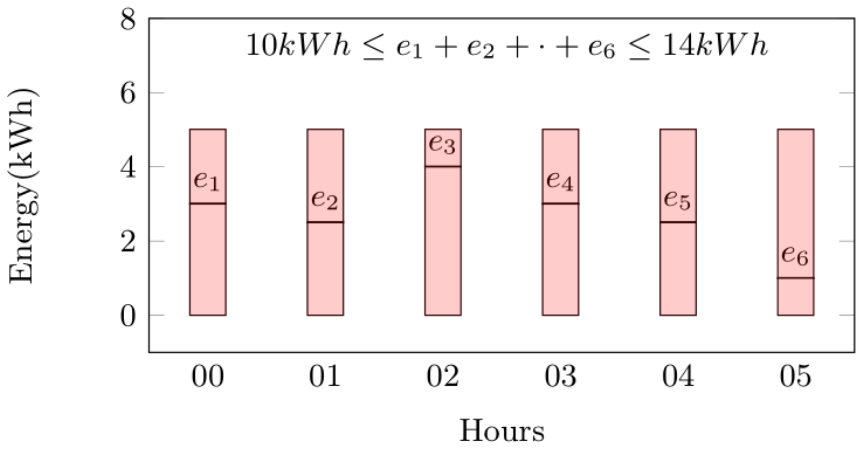


Figure 2.2: Example of TEC FO

A total energy constraint standard FO (TEC-SFO) is an FO with slice and total energy constraints. In the charging example, we can define a TEC-SFO by defining an SFO with all the slices between 0 kWh and 5 kWh, and adding a TEC defined by TE max = 14 kWh, since the maximum possible charge of the battery is 14 kWh. The prosumer would also usually define a minimum amount of charge to be obtained during the process, which may be for example 10 kWh

: this would be represented by a TEC defined by TE min = 10 kWh. These two TECs are shown in Figure 2.2.

2.3.1 FlexOffer Message Like in the previous subsection, we describe the sub-attributes for flexOfferProfileConstraints in the case of a TEC-SFO. The only change in comparison to SFOs is that inside FlexOfferProfileConstraint, we have the sub-attribute totalEnergyConstraint it is an object with two sub-attributes, lower and upper. They indicate the lowest and highest amount of total consumption for energy respectively.

The part in the JSON message relative to it would be as described below:

```
"flexOfferProfileConstraints": [ {
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "energyConstraintList": [{"lower": 0, "upper": 5}],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},
 "totalEnergyConstraint": [{"lower": [10],"upper": [14]}]
]
```

We can see that it is represented like an SFO, with the addition of the total

energy constraint.

Attribute	Mandatory	Type	Description
totalEnergyConstraint	Yes	Object	Contains the total energy constraints. Bounds the total energy amount requested or offered within the full active operation of a flexible resource. It declares the change of the SoC at the end of adaptation. Has two sub-elements: lower, and upper.
subTotalEnergyConstraint	Yes	Object	Describes the available capacity of energy reservoir for charging (lower) and discharging (upper) regarding the SoC at the adaptation start. Has two sub-elements: 'lower', and 'upper'.

Table 2.3: Additional sub-attribute for TEC-SFOs.

2.4 Dependency FlexOffers

A further type of constraint is the dependent energy constraint. This constraint specifies at each time unit t a lower and an upper bound on the amount of energy that can be consumed, depending on the total amount of energy that has been consumed before time unit t. In more formal terms, this means that there are three real numbers a, b, c such that

A dependency FO (DFO) is an FO with dependency energy constraints. Figure 2.2 shows a DFO created from the charging example, for the first four time units : for each slice, the x axis represents the amount of energy used up until that time unit, while the y axis represents the amount of usable energy at the considered time unit. In this figure, at time t = 1, 2 and 3, the amount of energy that has been consumed up to that time is 0 kWh, between 0 kWh and 5 kWh, between

$$a \cdot (e_1 + \ldots + e_{t-1}) + b \cdot e_t \leq c$$

Figure 3: equation

0 kWh and 10 kWh respectively, as shown in the x axis. The amount of energy that the prosumer may consume is always between 0 and 5 kWh, no matter the amount of energy consumed before, as shown in the y axis. At time t = 4, the amount of energy consumed up to that time is between 0 kWh and 14 kWh, and the amount that can be consumed depends on the amount consumed up to that time, as shown in the rightmost part of Figure 2.3. This constraint should be used for the most advanced forms of flexible resources (e.g., heat-pumps), where the flexibility changes over time and is dependent on an internal system state (e.g., temperature).

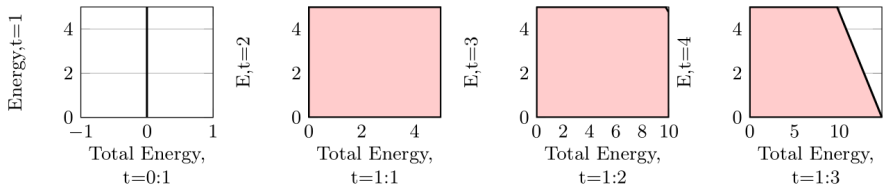


Figure 4: Example of DFO for the charging example.

Figure 2.3: Example of DFO for the charging example.

2.4.1 FlexOffer message Like in the previous subsection, we describe the sub-attributes for FlexOfferProfileConstraints in the case of a DFO. Inside FlexOfferProfileConstraints there is the sub-attribute DependencyEnergyConstraintList: it is a matrix, referred to the considered time unit. It represents a set of linear constraints which, in turn, represents the dependency energy constraints. The other attributes are the same as FlexOfferProfileConstraints. However, while EnergyConstraintsList has sub-attributes lower and higher like in the example before, DependencyEnergyConstraintsList has only the matrix. The matrix has as many rows as the number of sides of the polygons, and three columns: if the inequality representing the side is written as ax + by c, the row will be |abc|. This is called the H-representation of the slice [2].

Figure 2.4: Example of DFO for the switching example.

In the example of Figure 2.4, the set representing this DFO has four matrices. In Figure 4 we show a more complex example of DFO, which refers to the switching example from Section 1. At each time we can see the dependency

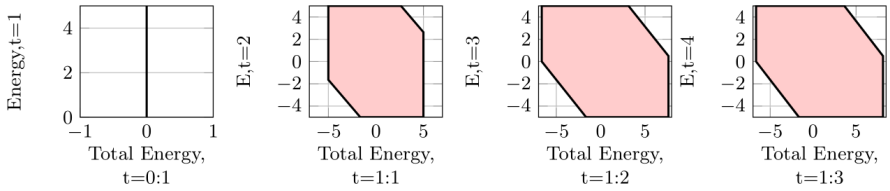


Figure 5: Example of DFO for the switching example.

between energy used up and energy available : for t = 1 the energy available is between -5 kWh and 5 kWh, for the following time units it depends on the amount used before. In all those time units the sum between energy used up to that moment and energy available is limited by a certain amount ; however, this amount changes depending on the time unit. In particular, the polygon representing the fourth time unit can be defined by the equations

Where x represents the amount of energy consumed up to time 3 included, and y the amount of energy that will be consumed at time 4. Those equations can be represented by the matrix

Attribute	Mandatory	Type	Description
DependencyEnergyConstraintList	Yes	Object	Contains the DFO constraints.

^*Table 2.4: Additional sub-attribute for DFOs.*

This is how this constraint is incorporated in the message :


```
"flexOfferProfileConstraints": [ {
 "DependencyEnergyConstraintList": [[0 1 5],[0 -1 -5]],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "DependencyEnergyConstraintList": [[0 -1 -5],[0 1 5],[-1 0 -5],[1 0 5],[-1 -1 -6.64],[1 1 7
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
 "DependencyEnergyConstraintList": [[0 -1 -5],[0 1 5],[-1 0 -6.64],[1 0 7.14],[-1-1 -6.64],
 "priceConstraint": {"minPrice": 0.03, "maxPrice": 0.15},
 "minDuration": 1,
 "maxDuration": 1
},{
```

$$
\begin{aligned}
y &\geq 5 \\
y &\leq -5 \\
x &\geq -6.64 \\
x &\leq 8.14 \\
x + y &\geq -6.64 \\
x + y &\leq 8.64
\end{aligned}
$$

Figure 6: equation (2.2)

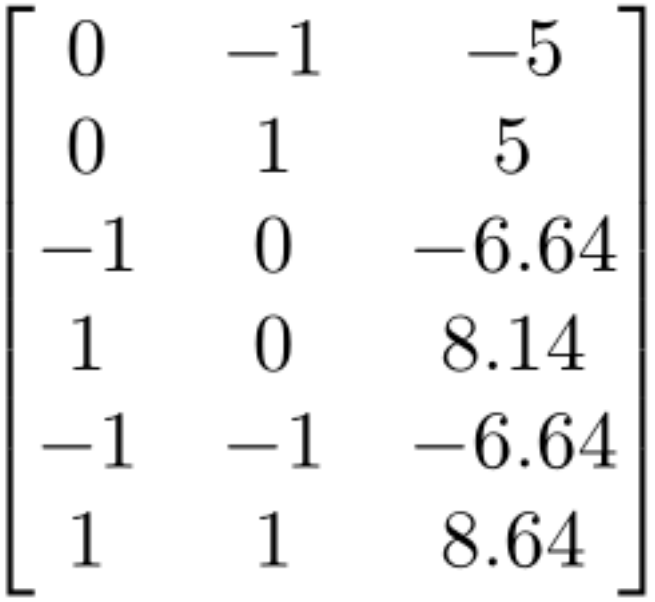


Figure 7: equation (2.3)

```
 "DependencyEnergyConstraintList": [[0 -1 -5],[0 1 5],[-1 0 -6.64],[1 0 7.64],[-1 -1 -6.64],
 "priceConstraint": {"minPrice": 0, "maxPrice": 0},
 "minDuration": 1,
 "maxDuration": 1
}
]
```


2.5 Uncertain FlexOffers

Uncertain FOs are a type of FOs that take uncertainty from flexibility approximation into account. There are two main types of uncertainty that are considered: time and amount uncertainty. For example, suppose that a prosumer wants to recharge an electric vehicle (EV) overnight. At each time unit, time uncertainty refers to the probability for the EV to be plugged in for recharge at that time, and amount uncertainty refers to the amount of energy that can be given to/taken from the EV at that time. An UFOs is created in two steps. First, uncertainty related to the device status is modeled at each time t; second, we calculate the probability for each energy value at each time to be feasible, taking into account all three types of uncertainty. We will then obtain some functions {f 1 ,...,f T } describing those probabilities: those functions will define the UFO. UFOs can be visualized by choosing a probability threshold P 0 . At each time t, the energy values having probability at least P 0 of being feasible can be described by intervals. Figure 2.6 shows what happens in the switching case: with P 0 = 1, the feasible energy values are described by the pink bar; however, if we choose P 0 = 0.8, the available flexibility is represented by the combined pink and blue bars. The figure shows that the choice of a value for P 0 generates a SFO: optimization and aggregation of UFOs are performed by choosing a value for P 0 , and then optimizing and/or aggregating the resulting SFOs.

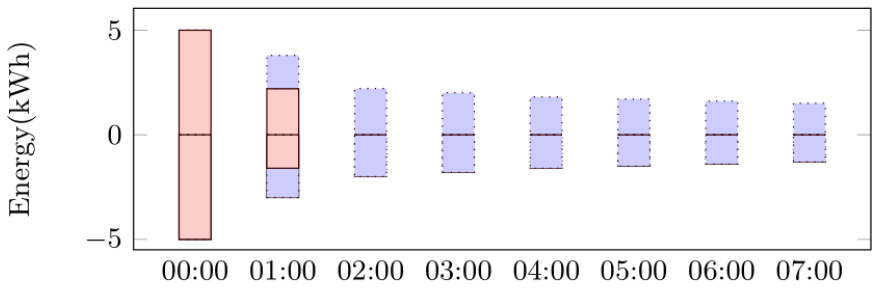
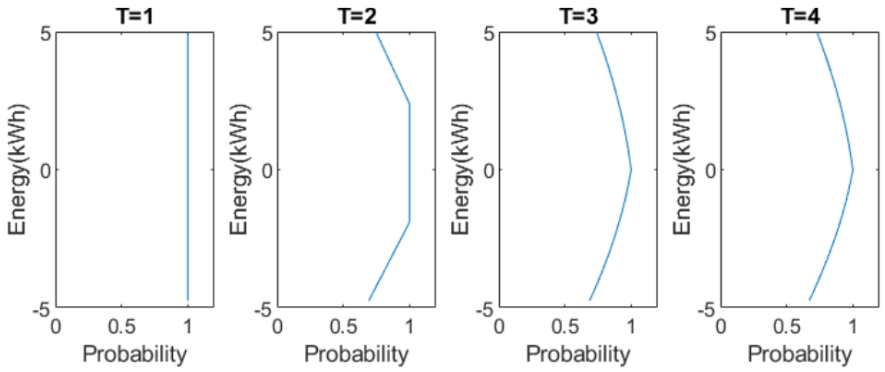


Figure 8: An uncertain FlexOffer (P 0 = 1 and P 0 = 0.8)

Figure 2.5: An uncertain FlexOffer (P 0 = 1 and P 0 = 0.8)

2.5.1 FlexOffer message We describe the sub-attributes for FlexOfferProfileConstraints in the case of an UFO. Similarly to DFOs there are new sub-attributes, shown in Table 2.5. It has to be noted that, up to today, UFOs have never been employed in a real FO message: we define now how this infomation



will be encoded.

Figure 2.6: An uncertain FlexOffer (P 0 = 1 and P 0 = 0.8)

The UncertainThreshold attribute is a real number P 0 which represent the probability threshold. UncertainFunctions are functions that are piecewise polynomials, and will be represented as such. As an example, for the switching case, the functions f 1 ,...,f 4 look like in Figure 6. For P 0 = 0.95, we represent the UFO by the following message (to be modified):

Attribute	Mandatory	Type	Description
UncertainFunctions	Yes ?	Define the UFO constraints.	
UncertainThreshold	Yes	Float	Defines the probability threshold.

Table 2.5: Additional sub-attributes for UFOs.

Example:

```
"UncertainflexOfferProfileConstraints": [
 {
  "UncertainFunctions": [{g1, g2, g3}],
  "UncertainThreshold": 0.95,
  "minDuration": 3,
  "maxDuration": 3
 }
]
```

xEMS-FOA exchange protocol

This protocol is not part of FlexOffer. However, an optional library of API/adapters will be made available open-source to facilitate the adoption of FO. Reference installations will also be described.