Community energy management with FlexMeasures, fully scriptable

Seita

LF ENERGY

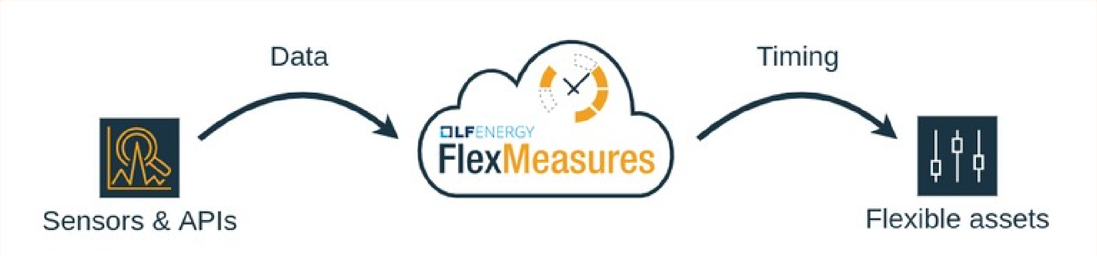
FlexMeasures

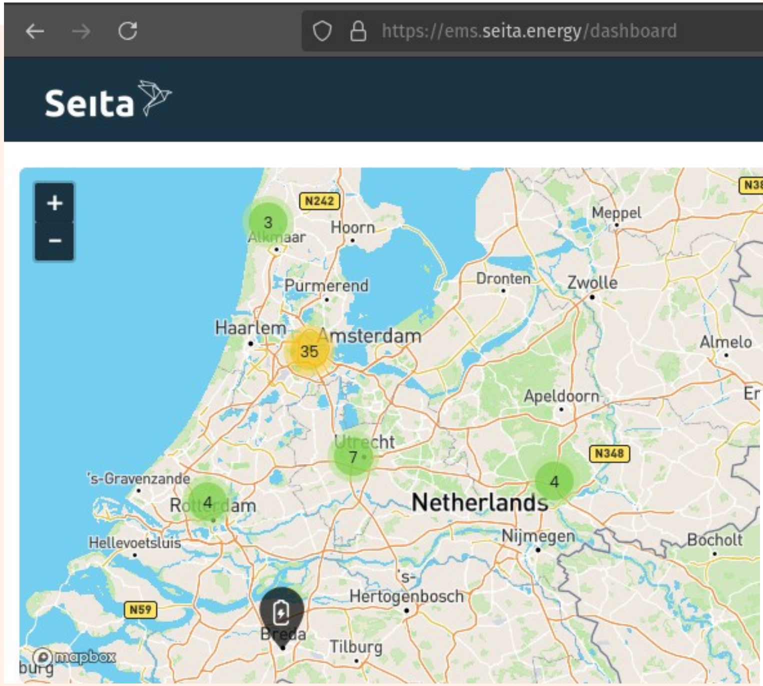
LF ENERGY

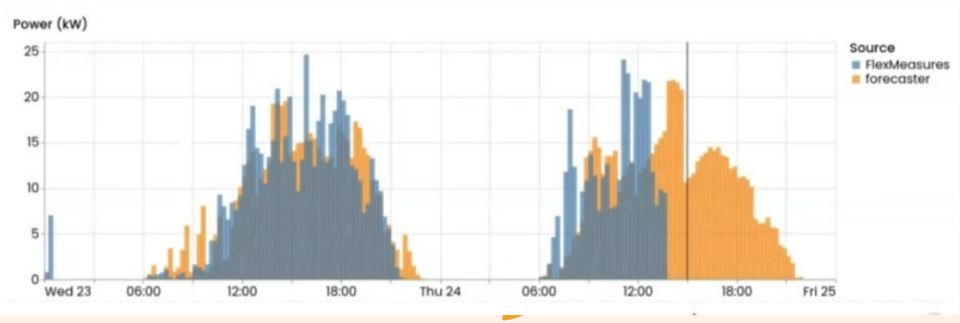
FlexMeasures

FlexMeasures

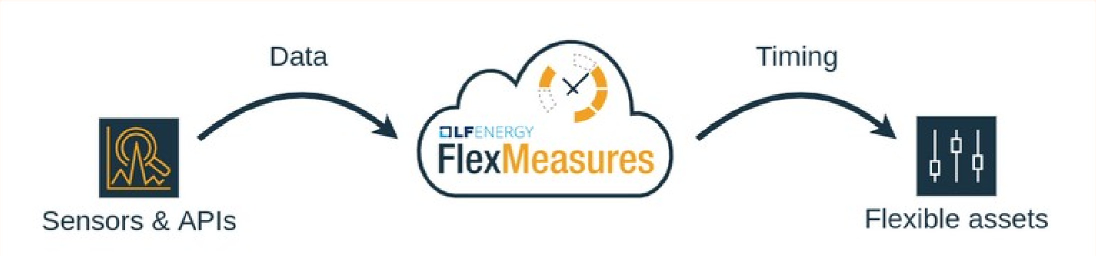
A smart planning layer — add smart orchestration to any site.

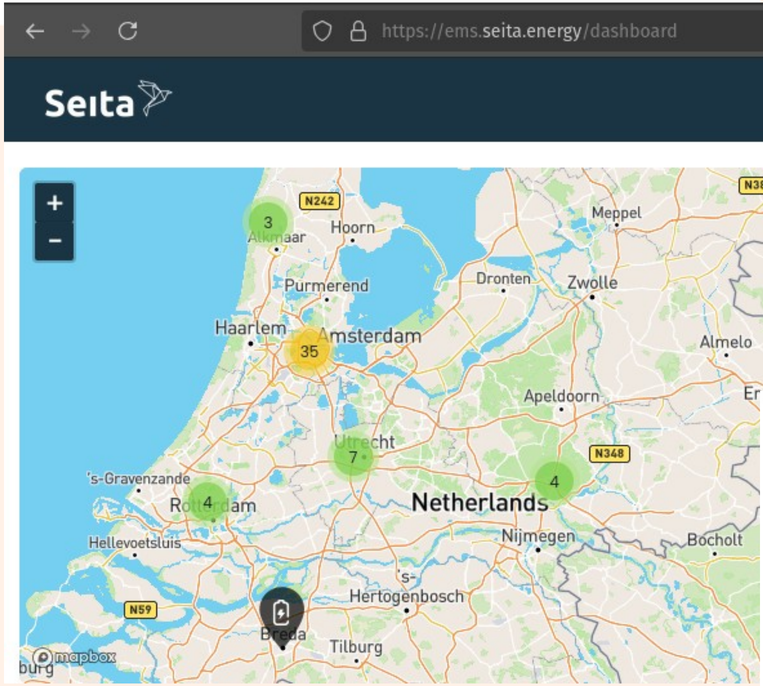



https://ems.seita.energy/dashboard



FlexMeasures – in short

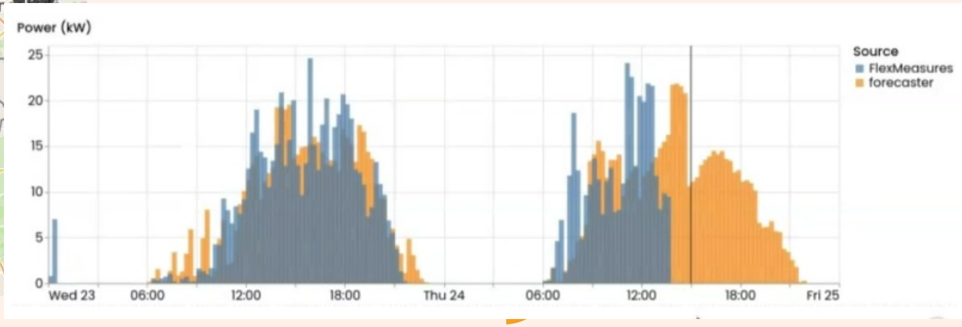




- A “Cloud EMS”
- Planning Behind-The-Meter +
- Storage / Processes
- Dev-friendly (API/CLI/docs/docker)

Proven in:

- EV / Vehicle-to-grid (V2G)
- Smart heating
- Sewage treatment



Check out previous FOSDEM talks

- 2023:
  V2GLiberty – the open stack that could

- 2024:
  Using FlexMeasures to Build a Climate Tech Startup in 15 Minutes

- 2025:
  Open protocols and S2 support

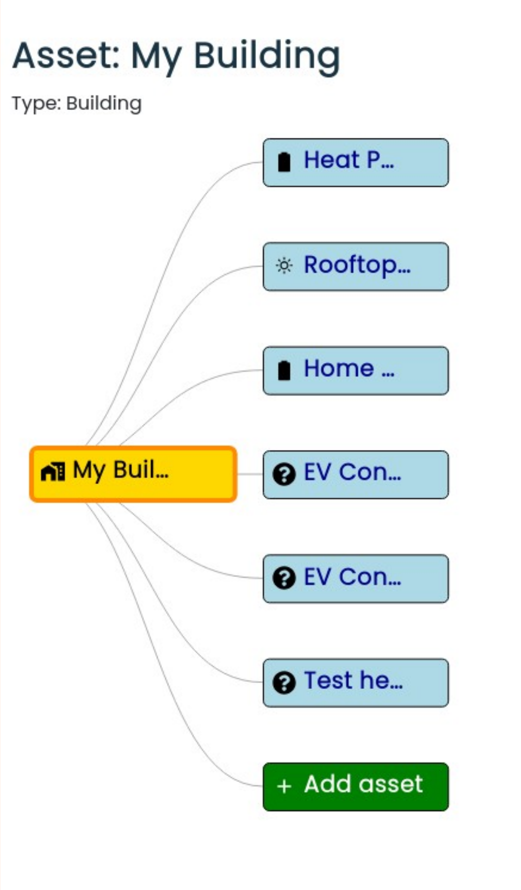
LF ENERGY FlexMeasures

Objective today: Show how to script FlexMeasures to fit your case + our approach for it as CEMS

CEMS == Community Energy Management System

FlexMeasures is a live platform ...

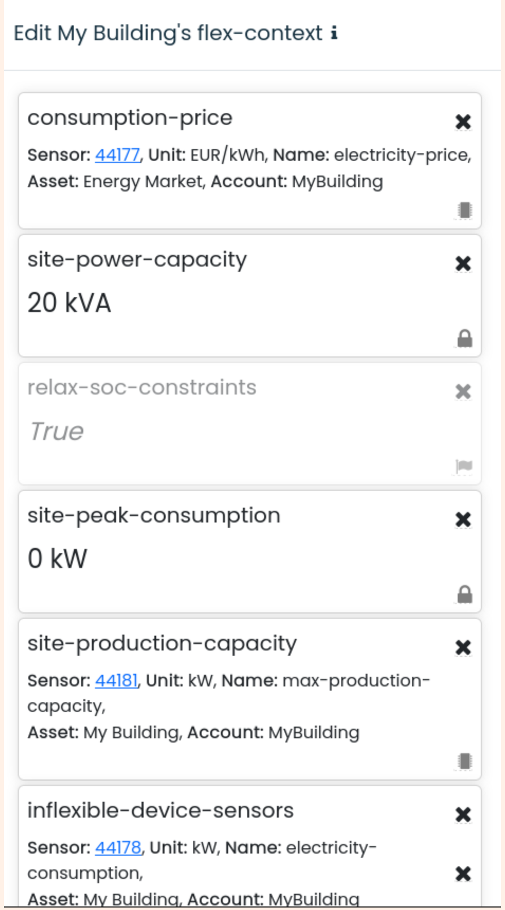
UI: Assets



Asset: My Building

Type: Building

UI: Flex-Config



Edit My Building's flex-context ℹ

consumption-price ✖

Sensor: 44177, **Unit:** EUR/kWh, **Name:** electricity-price, **Asset:** Energy Market, **Account:** MyBuilding

site-power-capacity ✖

20 kVA

relax-soc-constraints ✖

True

site-peak-consumption ✖

0 kW

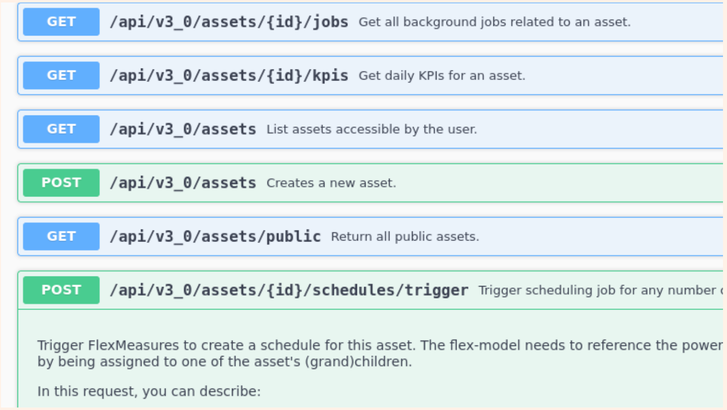
site-production-capacity ✖

Sensor: 44181, **Unit:** kW, **Name:** max-production-capacity,
Asset: My Building, **Account:** MyBuilding

inflexible-device-sensors ✖

Sensor: 44178, **Unit:** kW, **Name:** electricity-consumption, ✖
Asset: My Building, **Account:** MyBuilding

API: SwaggerDocs



GET `/api/v3_0/assets/{id}/jobs` Get all background jobs related to an asset.

GET `/api/v3_0/assets/{id}/kpis` Get daily KPIs for an asset.

GET `/api/v3_0/assets` List assets accessible by the user.

POST `/api/v3_0/assets` Creates a new asset.

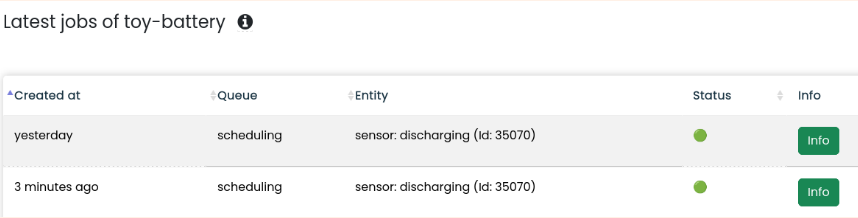
GET `/api/v3_0/assets/public` Return all public assets.

POST `/api/v3_0/assets/{id}/schedules/trigger` Trigger scheduling job for any number o

Trigger FlexMeasures to create a schedule for this asset. The flex-model needs to reference the power by being assigned to one of the asset's (grand)children.

In this request, you can describe:

Status page

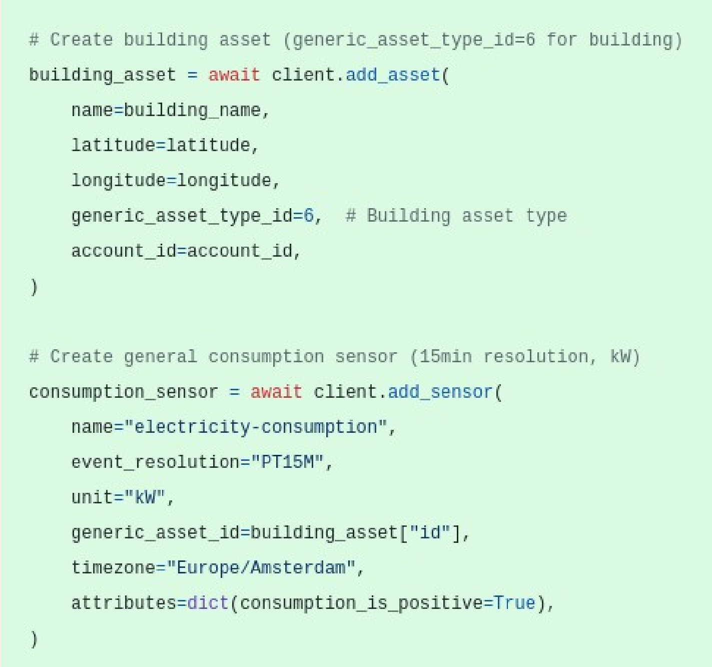


Latest jobs of toy-battery ℹ

Created at	Queue	Entity	Status	Info
yesterday	scheduling	sensor: discharging (Id: 35070)	●	Info
3 minutes ago	scheduling	sensor: discharging (Id: 35070)	●	Info

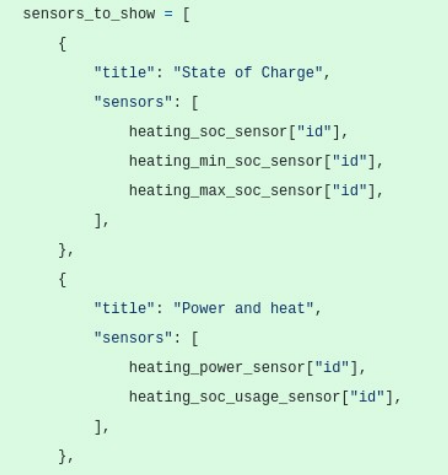
FlexMeasures can also be scripted ...

Create assets/sensors

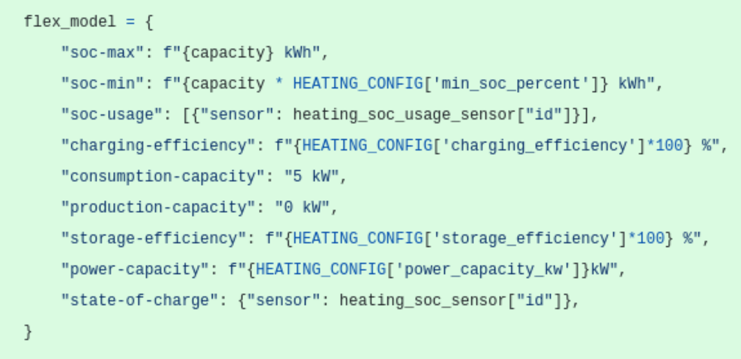


```python
# Create building asset (generic_asset_type_id=6 for building)
building_asset = await client.add_asset(
    name=building_name,
    latitude=latitude,
    longitude=longitude,
    generic_asset_type_id=6,  # Building asset type
    account_id=account_id,
)

# Create general consumption sensor (15min resolution, kW)
consumption_sensor = await client.add_sensor(
    name="electricity-consumption",
    event_resolution="PT15M",
    unit="kW",
    generic_asset_id=building_asset["id"],
    timezone="Europe/Amsterdam",
    attributes=dict(consumption_is_positive=True),
)
```

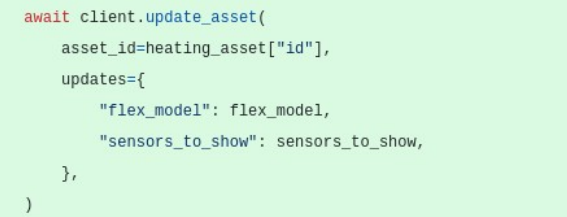
Configure flexibility & dashboard



```python
sensors_to_show = [
    {
        "title": "State of Charge",
        "sensors": [
            heating_soc_sensor["id"],
            heating_min_soc_sensor["id"],
            heating_max_soc_sensor["id"],
        ],
    },
    {
        "title": "Power and heat",
        "sensors": [
            heating_power_sensor["id"],
            heating_soc_usage_sensor["id"],
        ],
    },
```



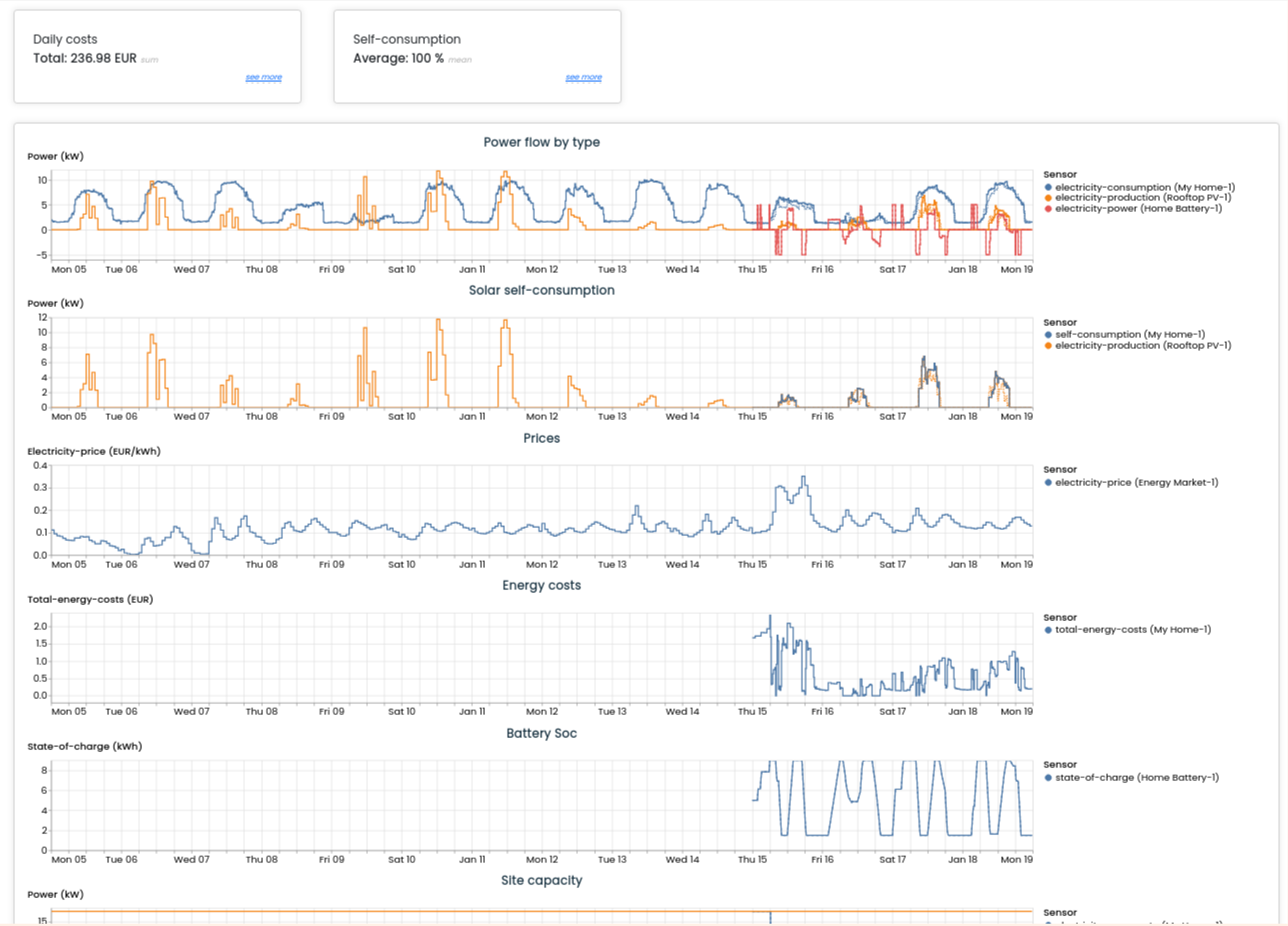
```python
flex_model = {
    "soc-max": f"{capacity} kWh",
    "soc-min": f"{capacity * HEATING_CONFIG['min_soc_percent']} kWh",
    "soc-usage": [{"sensor": heating_soc_usage_sensor["id"]}],
    "charging-efficiency": f"{HEATING_CONFIG['charging_efficiency']*100} %",
    "consumption-capacity": "5 kW",
    "production-capacity": "0 kW",
    "storage-efficiency": f"{HEATING_CONFIG['storage_efficiency']*100} %",
    "power-capacity": f"{HEATING_CONFIG['power_capacity_kw']}kW",
    "state-of-charge": {"sensor": heating_soc_sensor["id"]},
}
```



```python
await client.update_asset(
    asset_id=heating_asset["id"],
    updates={
        "flex_model": flex_model,
        "sensors_to_show": sensors_to_show,
    },
)
```

LF ENERGY FlexMeasures

.. which then can look like this.



Daily costs
Total: 236.98 EUR *sum*
see more

Self-consumption
Average: 100 % *mean*
see more

CEMS: Community Energy Management System

Motivation:

- Common grid connection
- Share local energy
- Collective flex actions
- ...

Use cases:

- Residential neighbourhoods
- Business parks

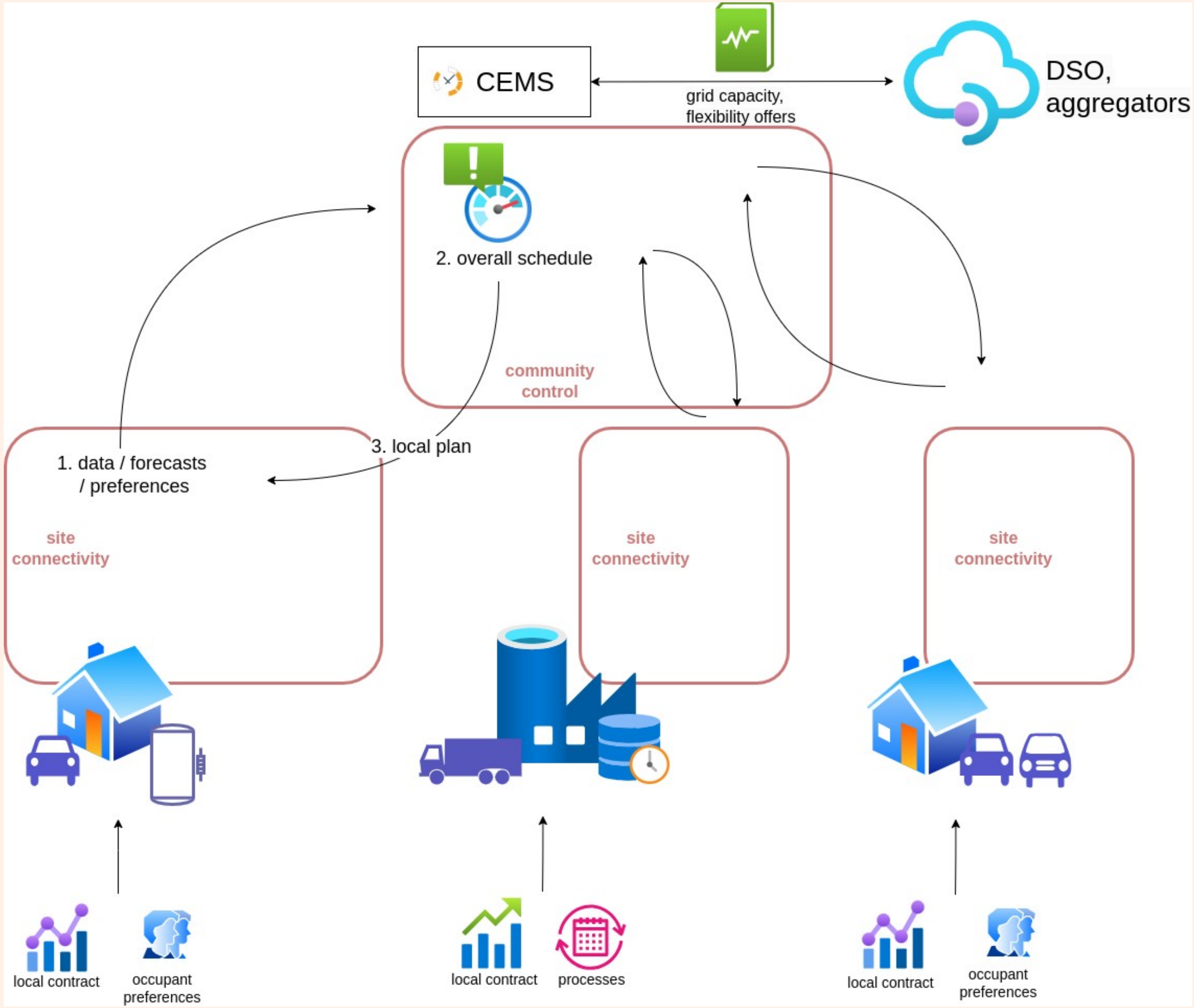
→ Break bottlenecks and increase savings.

→ Flexible planning is a challenge.

The FLEXED project is co-financed by TKI Energie from the Top Consortia for Knowledge and Innovation (TKIs) Allowance from the Ministry of Economic Affairs and Climate, The Netherlands.

LF ENERGY FlexMeasures

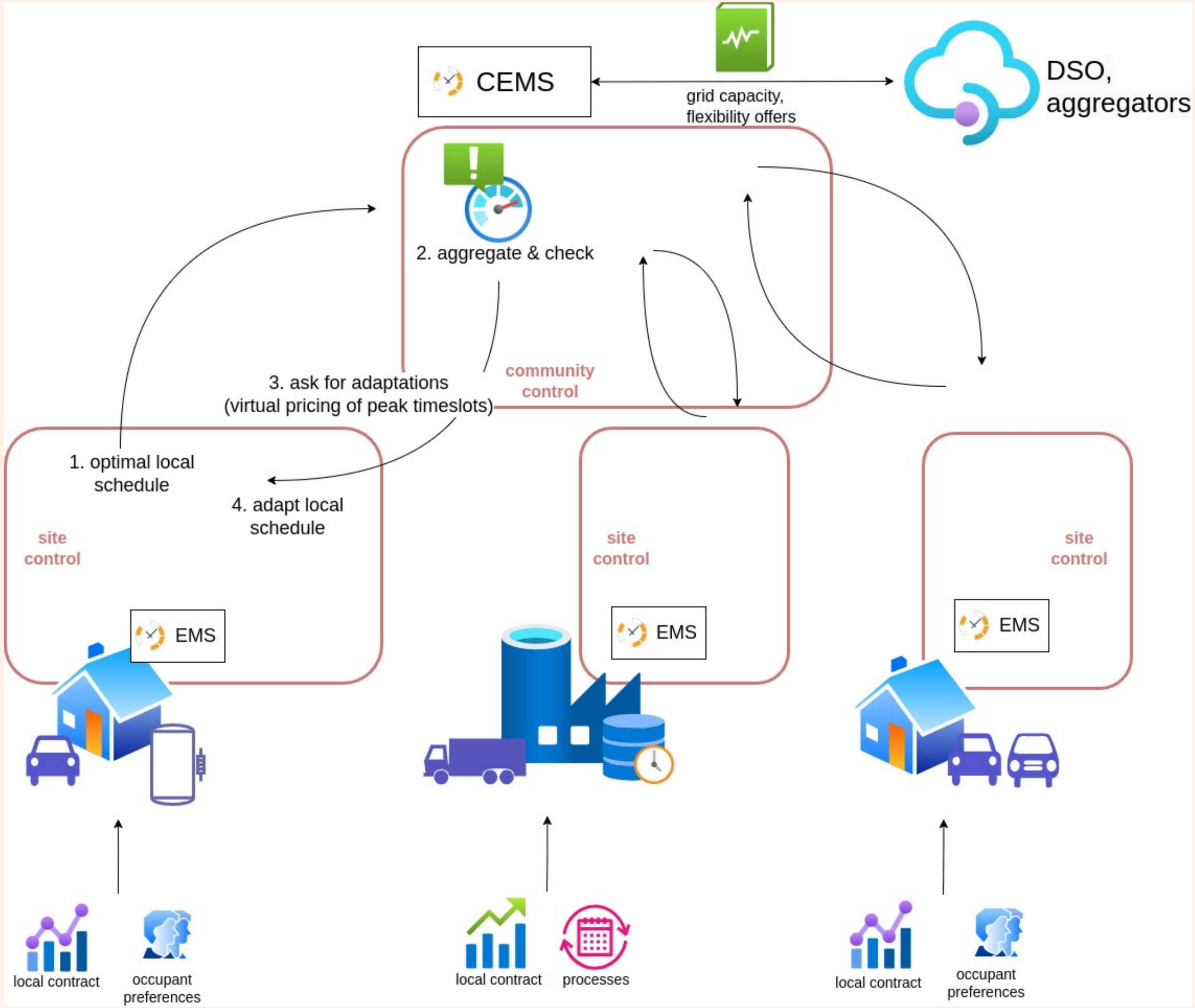
CEMS: “Fleet” Design



The FLEXED project is co-financed by TKI Energie from the Top Consortia for Knowledge and Innovation (TKIs) Allowance from the Ministry of Economic Affairs and Climate, The Netherlands.

LF ENERGY FlexMeasures

CEMS: “Federated” Design



The FLEXED project is co-financed by TKI Energie from the Top Consortia for Knowledge and Innovation (TKIs) Allowance from the Ministry of Economic Affairs and Climate, The Netherlands.

LF ENERGY FlexMeasures

Why did we choose the federated design?

1) Split problem → scalable
2) Autonomy & open for individual approaches
3) Preserve privacy

→ The CEMS layer governs contractual commitments.
→ It is not the all-powerful oracle.

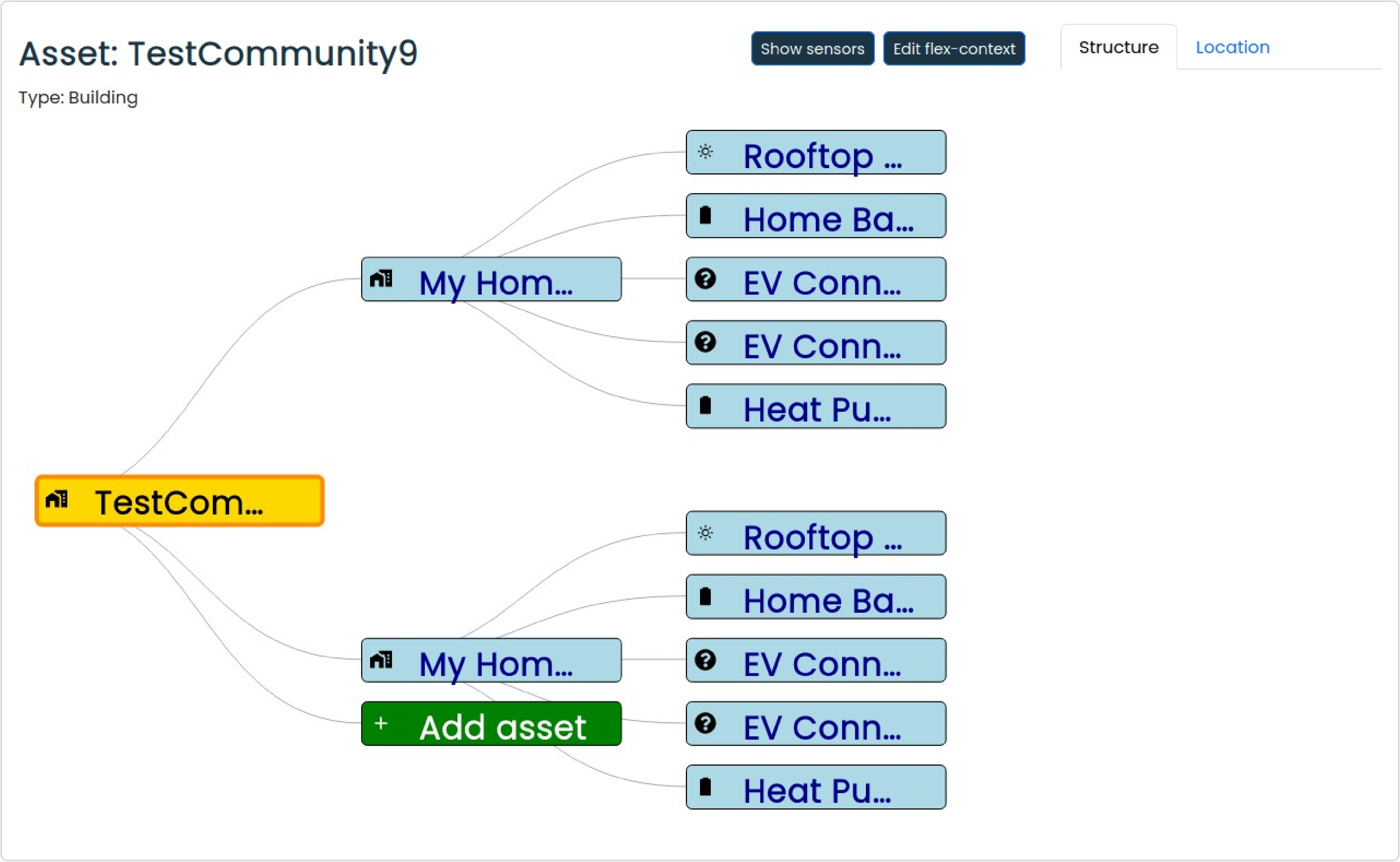
The FLEXED project is co-financed by TKI Energie from the Top Consortia for Knowledge and Innovation (TKIs) Allowance from the Ministry of Economic Affairs and Climate, The Netherlands.

Register as a custom FlexMeasures scheduler

```
# define in a FlexMeasures plugin; register plugin in flexmeasures.cfg

class CommunityScheduler(Scheduler):

     __author__ = "Seita"
    __version__ = "1"

    def compute(self, *args, **kwargs) -> list[dict]:
        """Compute new peak prices for the community's site schedulers.
        :returns: time series to have FlexMeasures save as sensor data
        """
        self._compute_aggregate_from_site_schedules()
        self._compute_breaches()
        new_prices = self._compute_peak_prices()
        new_capacities = self._compute_site_capacities()
        return new_prices + new_capacities
```

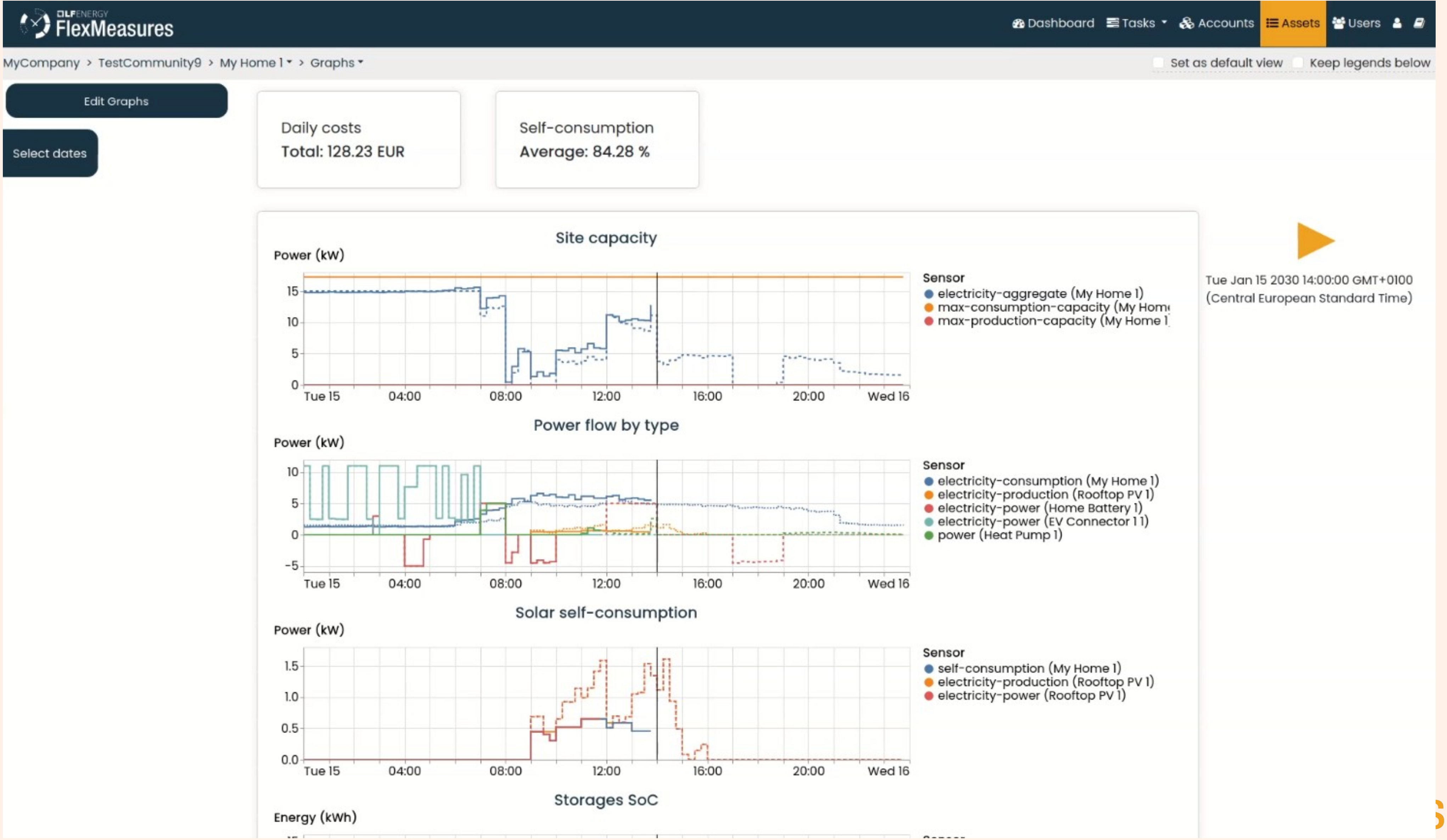
Control loop to schedule the community

```
async def run_scheduling(community: Community):
    """Iterate between async site scheduling and community optimization."""
    for rescheduling_iteration in range(MAX_RESCHEDULING_ITERATIONS):
        await asyncio.gather(site.schedule() for site in community.sites)
        if await not community_schedules_need_recomputation(community):
            break  # we’re fine

async def community_schedules_need_recomputation(community: Community) -> bool:
    """Run community scheduler, returns whether sites needs recomputing."""
    schedule = await community.schedule()
    if schedule["scheduler_info"]["changes made"]:
        return True   # CommunityScheduler updated each site's flex-context
                      # (peak prices, capacities)
    return False
```



Asset: TestCommunity9

Type: Building

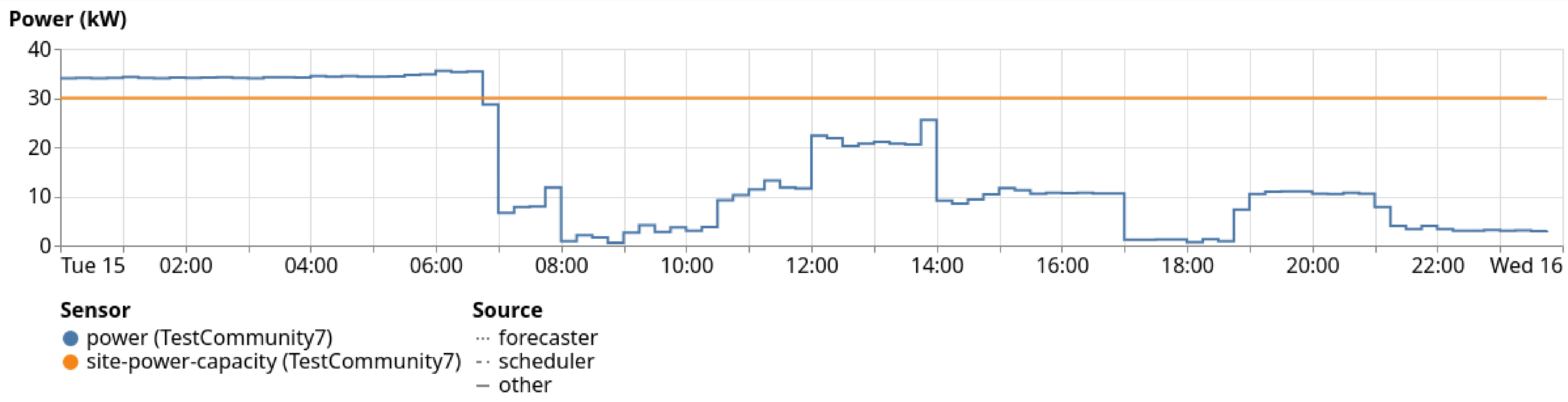

LF ENERGY
FlexMeasures
Dashboard
Tasks
Accounts
Assets
Users
MyCompany > TestCommunity9 > My Home 1 > Graphs
Set as default view
Keep legends below
Edit Graphs
Select dates
Site capacity
Power (kW)
15
10
5
0
Tue 15
04:00
08:00
12:00
16:00
20:00
Wed 16
Sensor
● electricity-aggregate (My Home 1)
● max-consumption-capacity (My Home
● max-production-capacity (My Home 1
Power flow by type
Power (kW)
10
5
0
-5
Tue 15
04:00
08:00
12:00
16:00
20:00
Wed 16
Sensor
● electricity-consumption (My Home 1)
● electricity-production (Rooftop PV 1)
● electricity-power (Home Battery 1)
● electricity-power (EV Connector 1 1)
● power (Heat Pump 1)
Solar self-consumption
Power (kW)
1.5
1.0
0.5
0.0
Tue 15
04:00
08:00
12:00
16:00
20:00
Wed 16
Sensor
● self-consumption (My Home 1)
● electricity-production (Rooftop PV 1)
● electricity-power (Rooftop PV 1)
Storages SoC
Energy (kWh)

Daily costs
Total: 128.23 EUR

Self-consumption
Average: 84.28 %

Tue Jan 15 2030 14:00:00 GMT+0100 (Central European Standard Time)

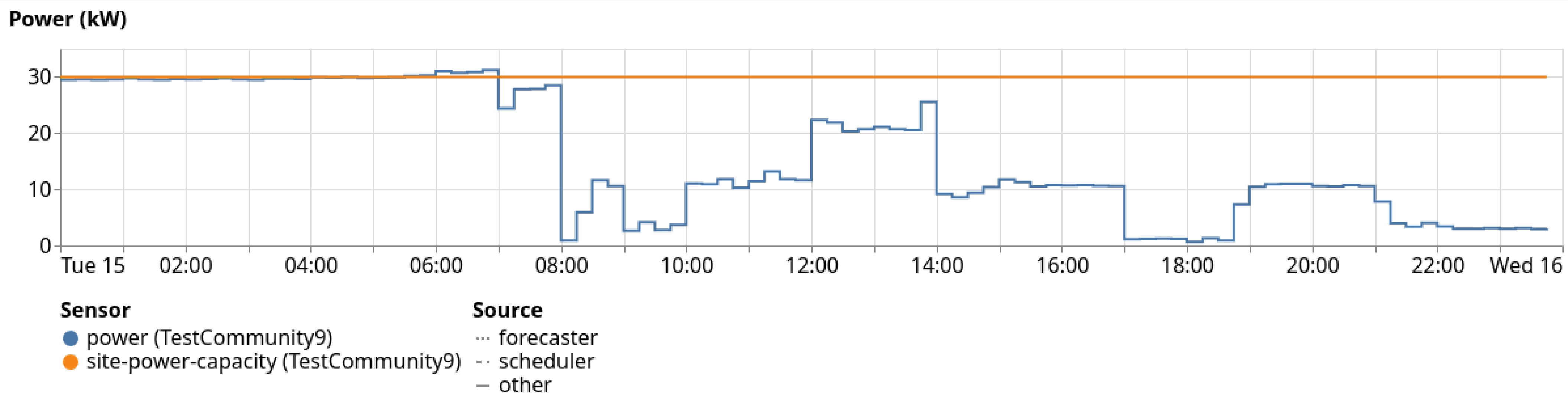
No CEMS: capacity breach



The FLEXED project is co-financed by TKI Energie from the Top Consortia for Knowledge and Innovation (TKIs) Allowance from the Ministry of Economic Affairs and Climate, The Netherlands.

LF ENERGY FlexMeasures

With CEMS: communal peak spread out



The FLEXED project is co-financed by TKI Energie from the Top Consortia for Knowledge and Innovation (TKIs) Allowance from the Ministry of Economic Affairs and Climate, The Netherlands.

LF ENERGY FlexMeasures

Community energy management with FlexMeasures, fully scriptable

Thanks !!

Happy to hear from you ...

- http://flexmeasures.io
- nicolas@seita.nl

Roadmap

- Dynamic capacity constraints
- Local “PPAs” for shared generation
- CEMS taking part in flex opportunities (CEMS collects & bids)

Avoiding the “waterbed effect”

- Discriminate peak pricing e.g. based on peak contribution, service level or randomization
- Kernel Density Estimation, to avoid sharp peak price transitions in favour of more Gaussian price peaks
- Randomized rewards to move demand/production to different periods
- Switch to tuning capacity limits if a desirable aggregate schedule has not been reached within a limited number of iterations.