FFG Forschung wirkt.
Bundesministerium Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie
AI for Green – Ausschreibung 2023

This project is funded by the FFG.
www.ffg.at

transpAIrent.energy

transpAIrent.energy

Transparent AI Forecasts for Green Energy in Austria

Project number: 910239

D4.2 Application software development report

WP4 – Platform development and implementation

30.04.2026

Authors	Organisation
Thomas BLEIER	B-SEC
Stefan STRÖMER	AIT

pbeg

B-SEC better secure

UBIMET

AIT AUSTRIAN INSTITUTE OF TECHNOLOGY

Funding Disclaimer

This project is being carried out as part of the 2023 “AI for Green” call for proposals from the Federal Ministry for Innovation, Mobility and Infrastructure (BMIMI). The processing is carried out on behalf of the BMIMI by the Austrian Research Promotion Agency (FFG). The project is funded as part of the topic of digital technologies, an initiative of the BMIMI, under the grant agreement number FO999910239. The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the texts, or in the decision to publish the results.

Table of Contents

Executive Summary

This document provides a report on the software development for the transpAIrent energy platform.

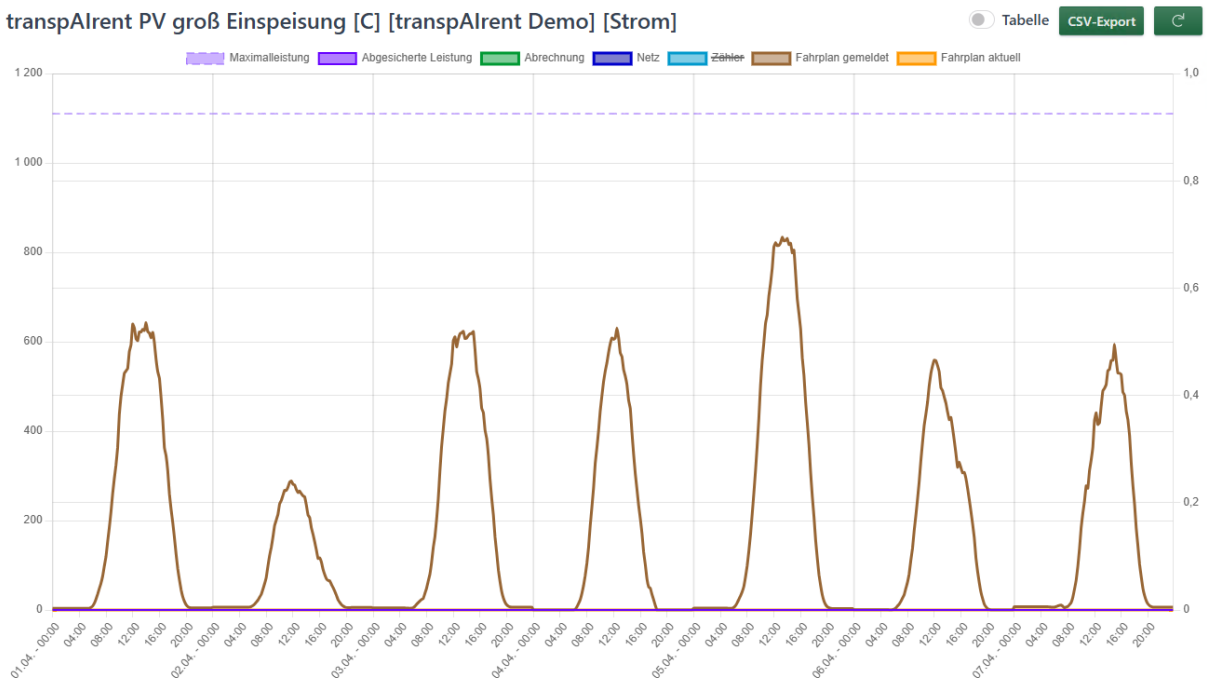
It shows the User Interface of the application running the operational optimization tool for the prototype site, as well as some API descriptions and development artefacts of the application.

1. Introduction

The operational optimization tool from AIT has been implemented in a production-grade environment for a prototype site. This site consists of a PV generation plant, a battery storage facility, local consumption and two additional consumption metering points at different geographical locations. Interfaces for programmatic access to the data have been developed, as well as an environment to run the operational optimization model in a production environment based on actual data from the prototype site. This also includes interfaces and tasks for importing PV forecasts from different sources, simple prediction algorithms for future demands on the different metering points, and market price data. Data from all this data sources is provided as input for the optimization model. The result from the optimization model is then stored in the platform, and available both for viewing via Web and Mobile UI as well as for programmatic access via API interfaces.

2. User Interface

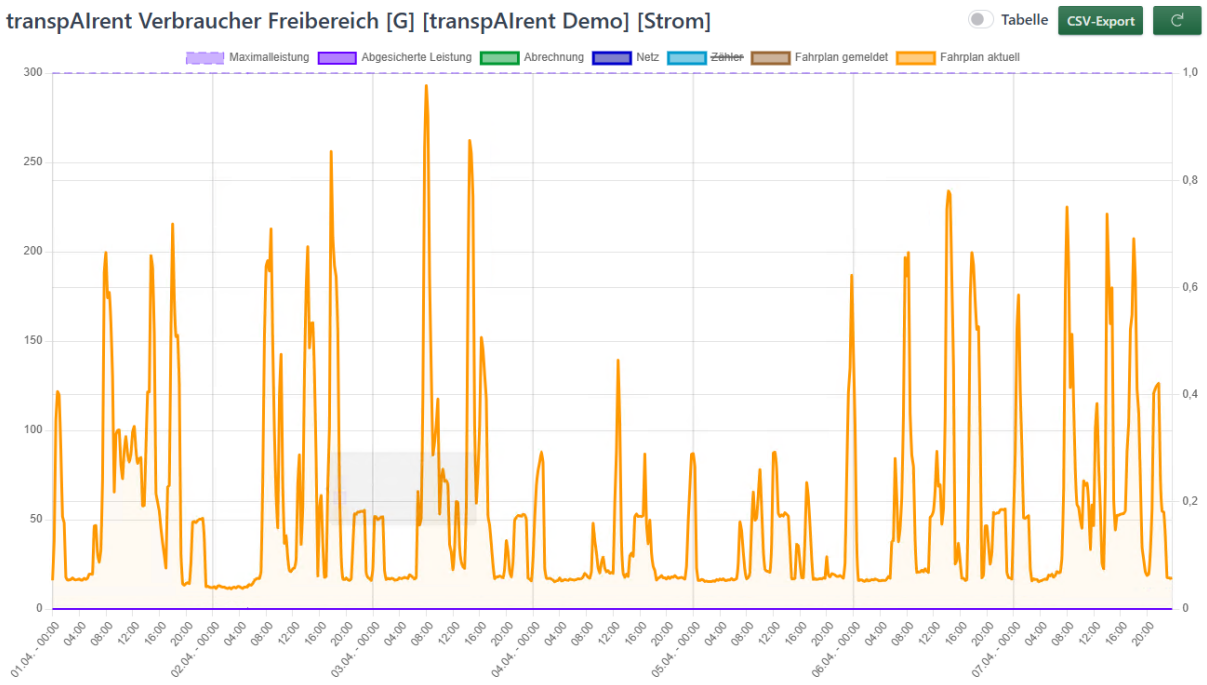
This screenshot shows the PV generation prediction, which is imported automatically every day for the following days from UBIMET, a partner in the transpAIrent project.



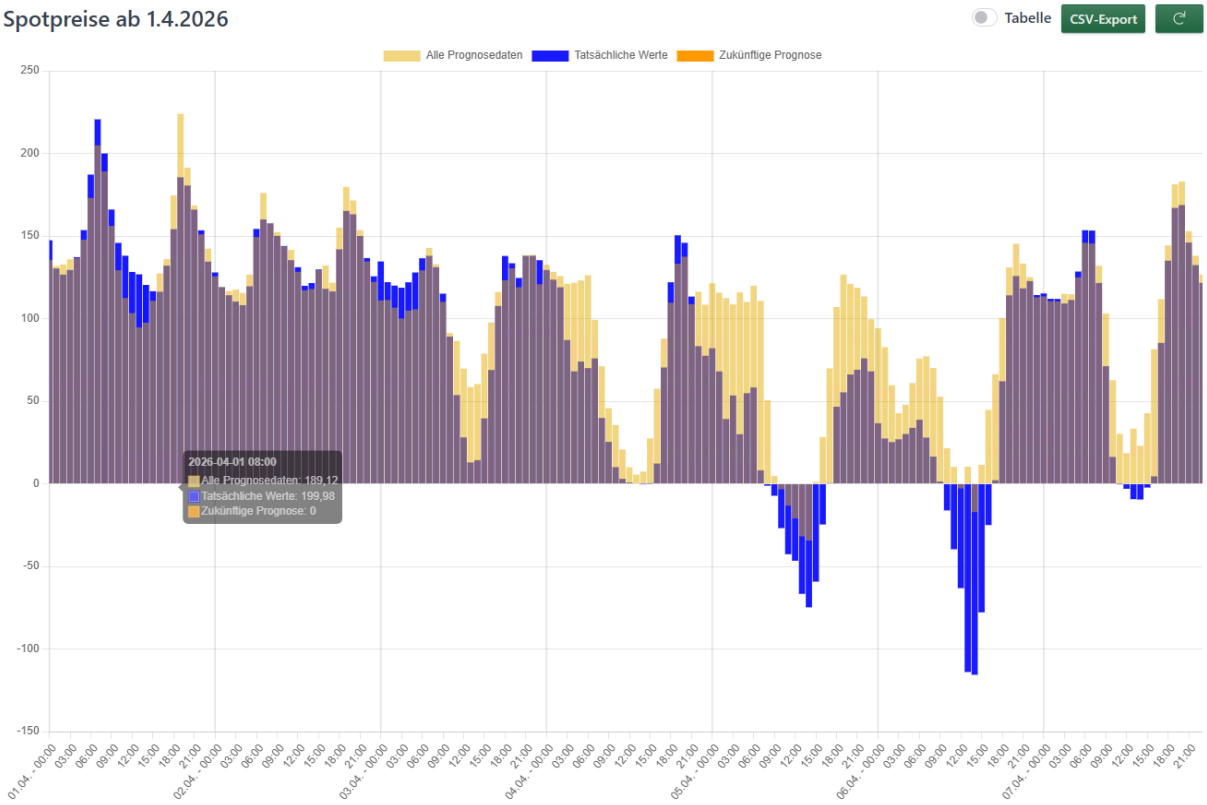
This screenshot shows the predicted consumption of a metering point, calculated based on the historic consumption data which is collected for this metering point. This is one of the inputs for the optimization model.

FFG Bundesministerium Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie
AI for Green – Ausschreibung 2023

This project is funded by the FFG.
www.ffg.at



Another input is the predicted energy market price, which is shown in this screenshot:



With this data the optimization model developed by AIT is called, which is running in a docker container in the production environment. For each call, detailed logs are being stored:

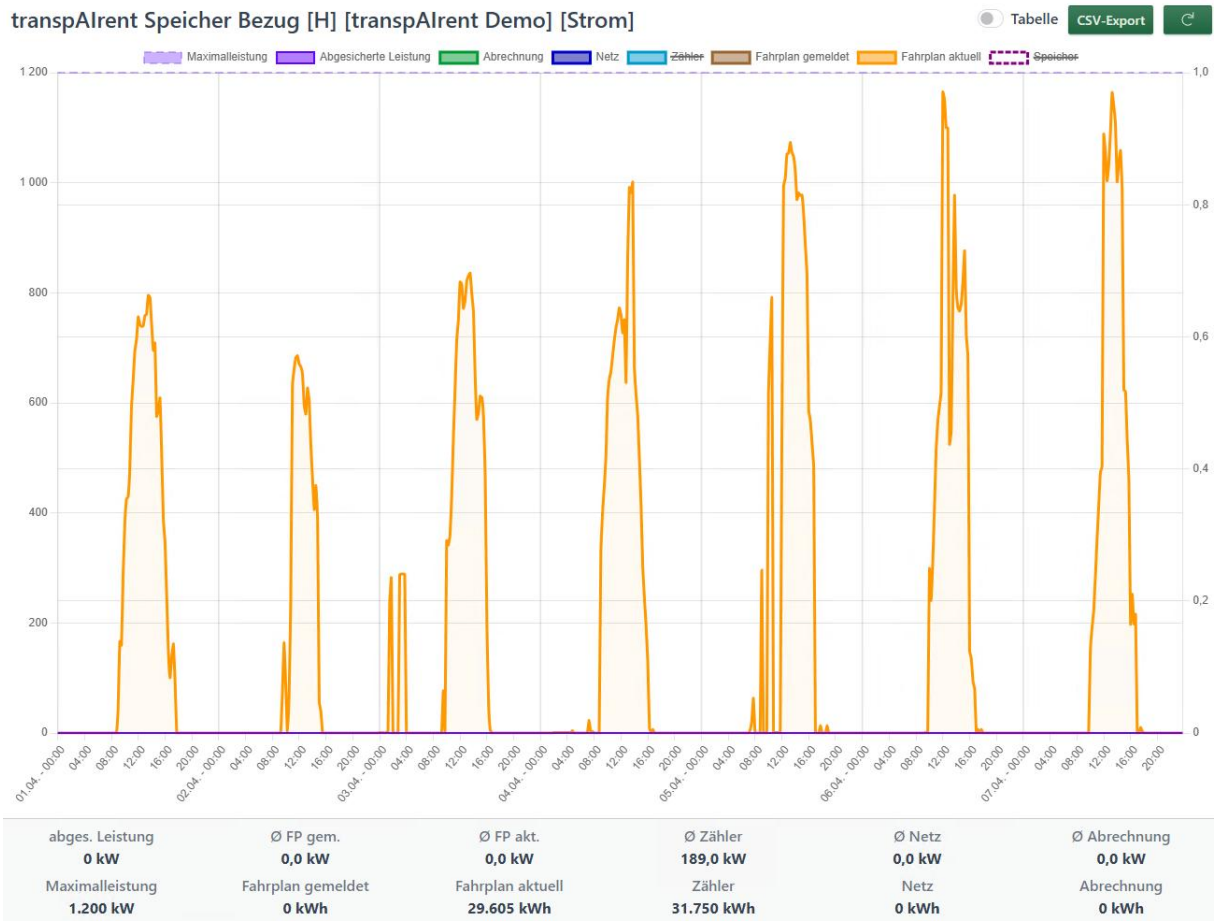
AI for Green – Ausschreibung 2023
This project is funded by the FFG.
www.ffg.at

Task Log Messages

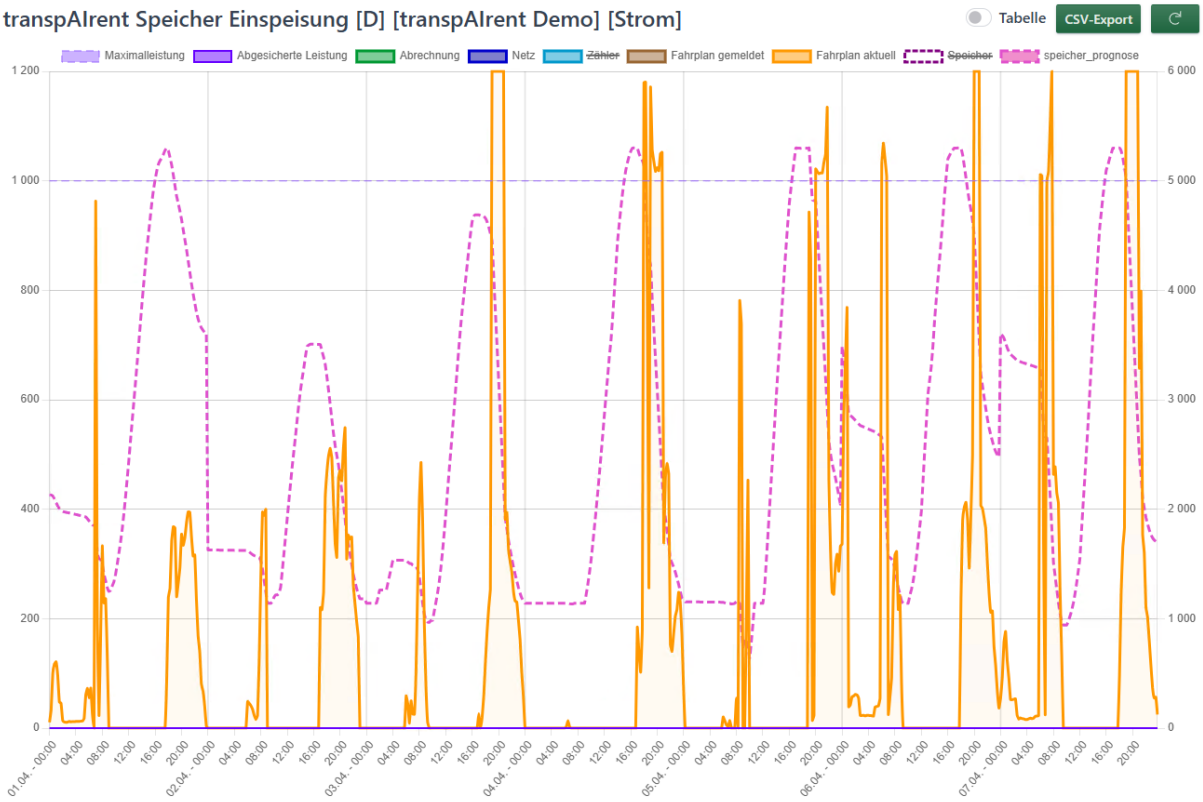
```
0000.973 [I] TaskRun [3960] started for Task ait-scheduler1 [3]: AIT Scheduler1 2026-04-01 bis 2026-04-06
0001.267 [I] Fetched energie_zaehler [prev_weeks_avg] timeseries [PoolZP 1: 1326]: 2026-03-04T06:15:00+01:00 - 2026-04-01T06:15:00+02:00 - 2682 Slots / 2682 Values
0001.468 [I] Fetched energie_zaehler [prev_weeks_avg] timeseries [PoolZP 1: 1327]: 2026-03-04T06:15:00+01:00 - 2026-04-01T06:15:00+02:00 - 2682 Slots / 2682 Values
0001.597 [I] Fetched energie_zaehler [prev_weeks_avg] timeseries [PoolZP 1: 1346]: 2026-03-04T06:15:00+01:00 - 2026-04-01T06:15:00+02:00 - 2684 Slots / 2684 Values
0001.801 [I] Fetched energie_zaehler [prev_weeks_avg] timeseries [PoolZP 1: 1347]: 2026-03-04T06:15:00+01:00 - 2026-04-01T06:15:00+02:00 - 2684 Slots / 2684 Values
0001.980 [I] Fetched energie_fahrplan [slotwert] timeseries [PoolZP 1: 1325]: 2026-04-01T06:15:00+02:00 - 2026-04-06T06:15:00+02:00 - 480 Slots / 480 Values
0002.111 [I] Fetched energie_fahrplan [slotwert] timeseries [PoolZP 1: 1323]: 2026-04-01T06:15:00+02:00 - 2026-04-06T06:15:00+02:00 - 480 Slots / 480 Values
0002.156 [I] Fetched speicher [slotwert_at_start_time] timeseries [PoolZP 1: 1324]: 2026-03-31T18:15:00+02:00 - 2026-04-01T06:30:00+02:00 - 49 Slots / 48 Values
0002.200 [I] Fetched energie_zaehler [slotwert_for_month] timeseries [PoolZP 1: 1321]: 2026-04-01T00:00:00+02:00 - 2026-04-01T06:15:00+02:00 - 25 Slots / 25 Values
0002.225 [I] Fetched energie_zaehler [slotwert_for_month] timeseries [PoolZP 1: 1346]: 2026-04-01T00:00:00+02:00 - 2026-04-01T06:15:00+02:00 - 25 Slots / 25 Values
0002.254 [I] Fetched energie_zaehler [slotwert_for_month] timeseries [PoolZP 1: 1347]: 2026-04-01T00:00:00+02:00 - 2026-04-01T06:15:00+02:00 - 25 Slots / 25 Values
0002.950 [I] Saved aitscheduler1 request to /srv/spm-app/aitscheduler1-log/aitscheduler1-20260401_042536_478631.txt
0002.951 [I] Called AIT Scheduler1 [0:00:00.679292]: 200
0002.980 [I] Stored positive Fahrplan from 'battery_setpoint_kW' [PoolZP: 1324]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:15:00+02:00, new, 409 values
0002.999 [I] Stored negative Fahrplan from 'battery_setpoint_kW' [PoolZP: 1330]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:15:00+02:00, new, 409 values
0003.177 [I] Stored 'battery_soc' [PoolZP 1: 1324]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:14:00+02:00, 96 new / 304 updated slots, 96 new / 304 updated values
0003.540 [I] Stored positive Fahrplan from 'schedule_bromberg_kW' [PoolZP: 1321]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:15:00+02:00, new, 409 values
0003.556 [I] Stored negative Fahrplan from 'schedule_bromberg_kW' [PoolZP: 1322]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:15:00+02:00, new, 409 values
0003.572 [I] Stored positive Fahrplan from 'schedule_brunn_kW' [PoolZP: 1346]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:15:00+02:00, new, 409 values
0003.589 [I] Stored positive Fahrplan from 'schedule_kirchschlag_kW' [PoolZP: 1347]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:15:00+02:00, new, 409 values
0003.608 [I] Stored combined Fahrplan from 'demand_bromberg_1' [PoolZP: 1326]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:15:00+02:00, new, 409 values
0003.629 [I] Stored combined Fahrplan from 'demand_bromberg_s' [PoolZP: 1327]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:15:00+02:00, new, 409 values
0003.638 [I] Stored combined Fahrplan from 'demand_brunn' [PoolZP: 1346]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:15:00+02:00, unchanged, 409 values
0003.648 [I] Stored combined Fahrplan from 'demand_kirchschlag' [PoolZP: 1347]: 2026-04-01T22:00:00+00:00 - 2026-04-06T06:15:00+02:00, unchanged, 409 values
0003.652 [I] TaskRun [3960] for Task [3] completed: AIT Scheduler1 - 480 Werte, warnings: Battery SoC softmin constraint was violated.
```

These logs fully document each scheduling call, including the complete set of inputs and outputs. Based on this information, a replay mechanism was implemented, which allows scheduling requests from a specific point in time to be reproduced and analysed in detail. This supports the assessment of model performance under real operational conditions and enables proposed model improvements to be tested against historic calls, providing a fact-based basis for iterative development of the platform.

The result of the optimization model is then also stored in the database

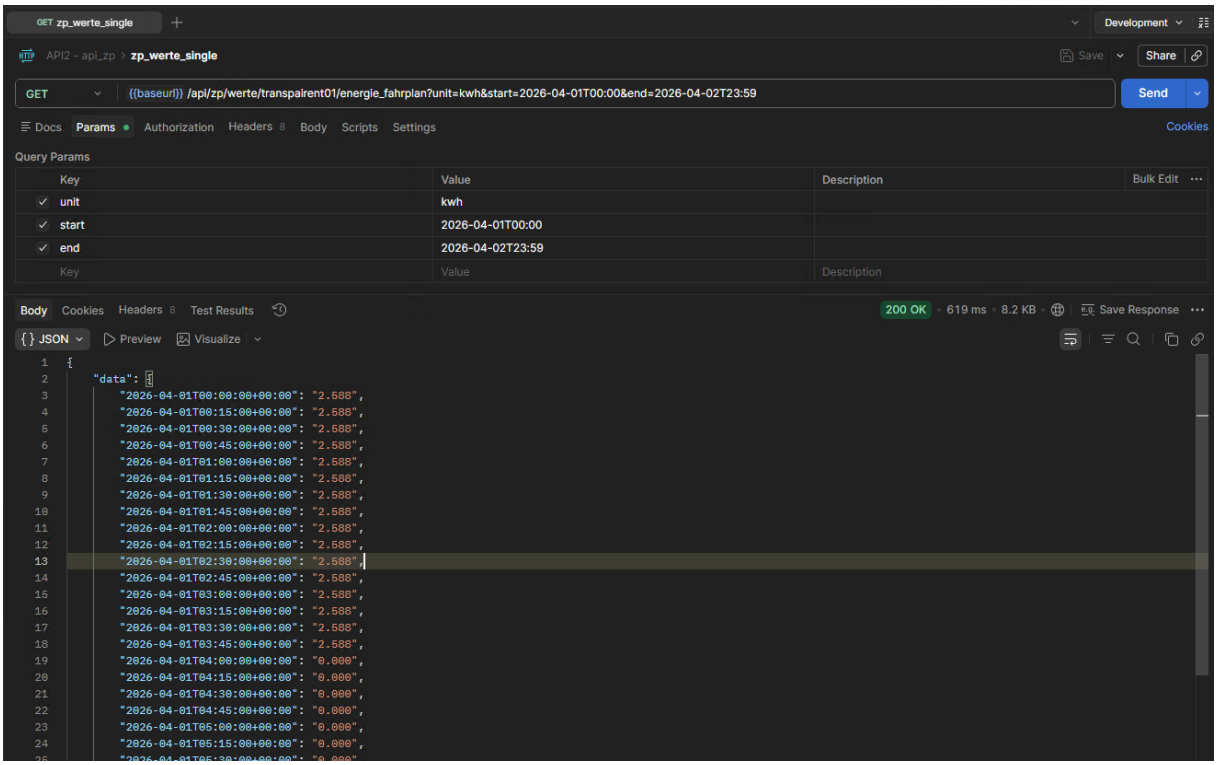


This project is funded by the FFG.
www.ffg.at

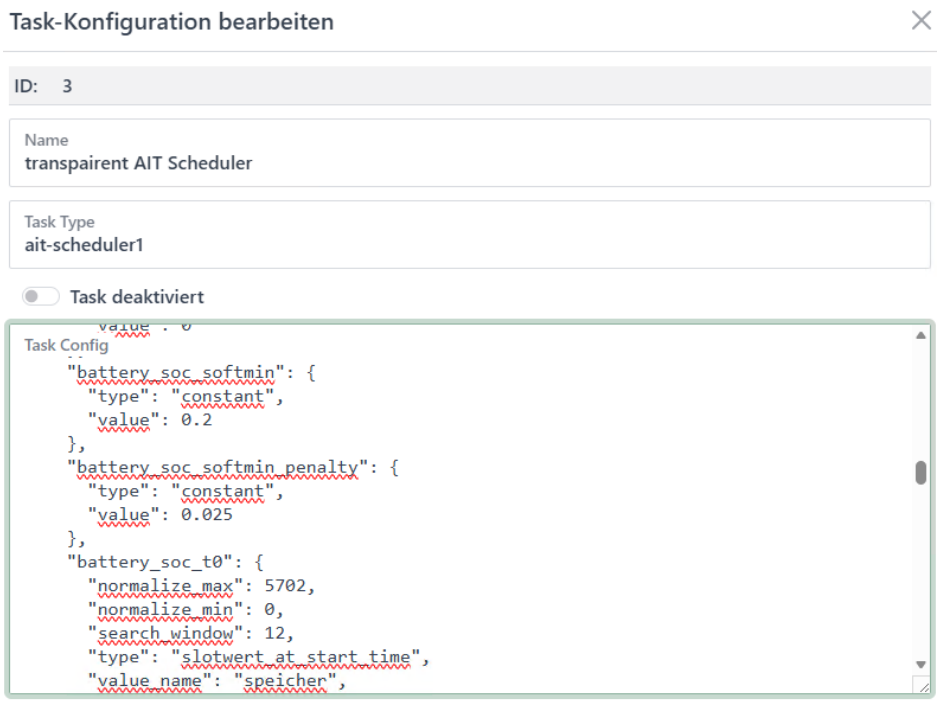


3. APIs and development artefacts

The system provides access not only via a web-browser based, graphical user interface, but also via a REST API for programmatic access. This is used for analyzing the data by the AIT team for continuous improvement of the model.



The interface to the optimization model has been encapsulated in a webservice, with configurable input- and output parameters, so that development of the application and the optimization model can be decoupled.



The following shows how the scheduler task is being called:

This project is funded by the FFG.
www.ffg.at

```python
def ait_scheduler1_task(cfg: task_utils.TaskRunConfig, trm: task_utils.TaskRunManager, doit: bool):  Thomas Bleier
    """Call AIT scheculer via Webinterface with data fetched from DB, and import results back to DB."""
    # get task config
    input_map = cfg.get_dict('input_map')
    input_parameters = cfg.get_dict('input_parameters')
    output_map = cfg.get_dict('output_map')
    days = cfg.get_int('input_days')
    ignore_output_first_day = cfg.get_bool( key: 'ignore_output_first_day',  default: False,  required: False)
    start_offset = cfg.get_int( key: 'start_offset_hours',  default: 0,  required: False)
    start_ts = timetools.get_slot_start(datetime.now(model.HomeTimeZone), exact=True) + \
               timedelta(hours=start_offset)
    end_ts = start_ts + timedelta(days=days)
    offset_txt = f" ({start_offset}h)" if start_offset != 0 else ""
    trm.start_task_run( run_info: f"AIT Scheduler1 {start_ts.isoformat()[:10]}{offset_txt} bis {end_ts.isoformat()[:10]}",
                        run_json: {'start_ts': start_ts.isoformat(), 'end_ts': end_ts.isoformat()})
    # fetch input data from DB
    input_tseries, input_info = get_input_slots(input_map, start_ts, end_ts)
    input_data = [input_tseries[ts] for ts in sorted(input_tseries.keys())]
    trm.result_json['input'] = input_info
    # prepare input parameters
    param_data, param_info = get_input_parameters(input_parameters, start_ts, end_ts)
    trm.result_json['input_parameters'] = param_info
    # prepare request
    input_data = {
        "start_time": start_ts.isoformat(),
        "data": input_data,
        "parameters": param_data
    }
    req_session = requests.Session()
    req_session.headers.update({
        'X-Scheduler-Token': current_app.config.get('AIT_SCHEDULER1_TOKEN', ''),
        'Accept': 'application/json'
    })
    req_timeout = current_app.config.get('EEA_DEAL_API_TIMEOUT', 30)
    start_time = datetime.now().astimezone(timezone.utc)
    # call AIT scheduler via Webservice
    url = urljoin(current_app.config['AIT_SCHEDULER1_API_URL'],  url: 'schedule')
    try:
        resp = req_session.request( method: "POST", url, json=input_data, timeout=req_timeout)
    except requests.exceptions.ConnectionError as e:
        model.db.session.rollback()
        raise task_utils.TaskAbortException(f"Error calling webservice: {get_root_error(e)}")
    # save and log result
    req_duration = datetime.now().astimezone(timezone.utc) - start_time
    save_request(resp)
    req_duration_str = int(req_duration.total_seconds() * 1000) if req_duration else 0
    level = logging.ERROR if resp.status_code != 200 else logging.INFO
    current_app.logger.log(level,  msg: f"Called AIT Scheduler1 [{str(req_duration)}]: {str(resp.status_code)}",
                           extra={'spmextra': {'request_time': req_duration_str}, 'resp_status': resp.status_code})
    if resp.status_code != 200:
        model.db.session.rollback()
        raise task_utils.TaskAbortException(f"Webservice returned {str(resp.status_code)}")
    resp_json = resp.json()
    if 'data' in resp_json and isinstance(resp_json['data'], list):
        try:
            output_info = process_output_slots(output_map, resp_json['data'], start_ts, end_ts, ignore_output_first_day,
                                               doit)
```

The scheduler itself is developed in a coordinated fashion using GitHub and, at the time of writing, is in its 82nd iteration. It directly includes a documentation of all parameters and input data:

This project is funded by the FFG.
www.ffg.at

Data

Time Series Inputs

Data	Unit
PV forecast	kW
Demand forecast	kW
Day-ahead market prices	€/MWh

Time Series Outputs

Data	Unit
"Grid" schedules	kW
Battery setpoints	kW
Battery SoC	% (0-1)

Parameters

Default values as of 20.12.2025 are shown below as sample refeference:

Parameter	Value	Unit
self_consumption_penalty	0.015	€/kWh (applying 50:50 to consume/feed-in)
battery_soc_softmin	0.20	% (0-1)
battery_soc_softmin_penalty	0.025	€/kWh per hour
battery_p	1200.0	kW
battery_e	5702.0	kWh
battery_eta	0.90	% (round trip)
battery_loss	0.0006	%/h (0-1/h)
battery_soc_min	0.00	% (0-1)
battery_soc_max	0.90	% (0-1)
battery_vom	0.0075	€/kWh (applying 50:50 to charge/discharge)

Further, results and internal specifics are documented:

Results

Results are - as most values - given in `kW` for setpoints, etc., and unitless (0-1) for the state of charge (SoC). The SoC is given as a fraction of the battery's total energy capacity (so `0.9` refers to the upper bound of the allowed range of operation if `battery_soc_max` is set to `0.9`).

This is especially important for schedule related results, where the 15-minute intervals implicate that results (in terms of power) are not (numerically) equal to the corresponding energy values (in kWh) for the same time period.

Storage levels

The scheduler currently starts at `00:00` and runs until the end of the passed data window, forcing the storage to have at least the initial state of charge at the end of the scheduling period (ending with more stored energy is allowed).

Storage & self-consumption

The storage is currently configured to not allow discharging to sell on the day-ahead market - assuming that forecasts are not accurate enough to ensure large enough spreads (when accounting for grid costs, degradation, etc.) to allow trading spreads across the day. This is done by only allowing it to discharge to cover the local demand. Charging is allowed from either the PV or the grid.

To further ensure that development follows sustainable practices, the integration of the scheduler into the overall software harness is supported by a test suite:

Running Tests

The project uses pytest for automated testing. To run the tests:

```
# Install development dependencies (includes pytest)
uv sync --dev

# Run all tests
uv run pytest

# Run tests with verbose output
uv run pytest -v

# Run tests from a specific file
uv run pytest tests/test_util.py -v
```


This project is funded by the FFG.
www.ffg.at

Project coordinator:

Klara Maggauer, M. Sc., B. Sc.

AIT Austrian Institute of Technology GmbH

Center for Energy

Giefinggasse 4

A-1210 Vienna

E-mail: klara.maggauer@ait.ac.at

This project is funded by the FFG.
www.ffg.at