Adam Salomonsson

Summary

This application provides a Python-based interface for reading and processing energy tariff data structured according to RISE specifications.  
It uses OpenAPI-generated client code and includes an interaction layer for accessing and optimizing pricing data from both APIs and local JSON files.  
The project is modular, testable, and built to support fixed, energy, and power tariff structures.

**Python Application for Handling JSON APIs Using RISE’s Energy Tariff Data Format**

Documentation for Api\_interactionlayer.py

# **Abstract**

This document outlines the structure and functionality of a Python-based application framework built on the RISE **project's gridtariff-v0.1.0.json** API specification. The system uses OpenAPI tools to generate API client files (**GeneratedApiFiles**), enabling programmatic access to electricity tariff data. The interaction with these APIs is encapsulated in the **api\_InteractionLayer.py** script, which abstracts complex API operations.

At the core of the script is the **Tariffs** class, which initializes the API client and provides methods to retrieve tariff information from either the live API or local JSON files. A nested **Price** class handles pricing-related data and calculations, including fixed, energy, and power components in the tariff.

Two further subclasses, **Energy** and **Power**, provide support for data handling and cost optimization over time, such as computing average usage, estimating cost at specific times and identifying optimal start times for minimizing energy or power costs.

This modular design enables flexible integration of tariff and pricing logic into larger applications focused on energy cost optimization.

**Table of contents:**

[**Abstract** 1](#_Toc205282720)

[**Documentations on application** 2](#_Toc205282721)

[**Design of script** 3](#_Toc205282722)

[Main Class: **Tariffs** 3](#_Toc205282723)

[Tariff Methods 3](#_Toc205282724)

[Subclass: **Price** 4](#_Toc205282725)

[Price Methods 4](#_Toc205282726)

[Nested Subclasses of **Price** 5](#_Toc205282727)

[Energy and Power Methods 5](#_Toc205282728)

[**Further Development** 6](#_Toc205282729)

# **Documentations on application**

The code framework is based on the API structure of the RISE project and the **gridtariff-v0.1.0.json** file. From this file, a structure is generated using an OpenAPI script, which sets up the **GeneratedApiFiles**. These files define how the code interacts with the different APIs. To run OpenAPI, use the following command in a Bash or PowerShell terminal:

* **npx @openapitools/openapi-generator-cli generate -i gridtariffapi-v0.1.0.json -g python -o GeneratedApiFiles**

This command must be run from within the project's Openapi directory. It creates a structured codebase that can be used by the application. The interaction layer uses functions from the generated files to retrieve data from the web via HTTPS requests to the APIs and is therefore very important to the script.

The **api\_InteractionLayer.py** script abstracts and simplifies access to tariff, pricing, and optimization logic for power and energy. It can retrieve information from both the API client and local JSON files. The script is centred around the core class **Tariffs**, which initializes the API client and provides access to pricing logic through nested subcomponents. The **Tariffs** class is designed to interact with the higher-level components of each JSON structure, while the nested **Price** class focuses on extracting and working with all pricing-related components. Handling the three types of pricing: fixed, energy, and power. The two nested classes in **Price**, **Energy** and **Power**, are implemented in a similar structure. If specific logic is needed for either type, it should be added within their respective subclasses. A dedicated Fixed class does not currently exist because the number of fixed-price components is small, and the occurrence of such pricing objects in the application is relatively low and thy often apply only once a year.

Helper functions are categorized as functions that reshape input data or extract specific information from the JSON structure, supporting higher-level functions such as operation cost and optimization calculations.

In the project on GitHub there exists an example file on how functions should be used. This project also includes a test file which is implemented to test functions. If changes are made these tests insure that helper and main functions interact in the intendent way.

# **Design of script**

## Main Class: **Tariffs**

Handles tariff-related queries and data retrieval.

**Tariffs(v: str, url: Optional[str], json\_path: Optional[str], tariff\_id: Optional[str])**

#### **Initialization:**

* v: API version (e.g., "v0").
* url: API base URL.
* tariff\_id: Optional default tariff to use in subsequent queries.

#### **Attributes:**

* api\_instance: The OpenAPI-generated TariffApi client.
* tariff\_id: Default tariff ID.
* price: An instance of the nested Price class.
* \_json\_path: (Optional) Path to a local JSON fall back.

### Tariff Methods

| **Method** | | **Description** | |
| --- | --- | --- | --- |
| **get\_tariffs()** | | Retrieves all tariffs from the API. | |
| **get\_tariffs\_byJson(path)** | | Loads tariffs from a local JSON file. | |
| **get\_tariff(tariff\_id)** | | Retrieves a specific tariff object. | |
| **get\_tariff\_byName(name, company)** | | Fetch tariff by name and company. | |
| **get\_companys()** | | Returns all company names. | |
| **get\_company(tariff\_id)** | | Returns the company of a specific tariff. | |
| **get\_tariffs\_ids()** | | Returns all tariff IDs. | |
| **get\_tariffs\_names()** | | Returns all tariff names. | |
| **get\_id(index)** | | Gets the ID of the tariff at a given index. | |
| **set\_id(tariff\_id)** | | Sets a default tariff ID. | |
| **check\_id(tariff\_id)** | | Checks if the ID exists. | |
| **get\_id\_byName(name, company)** | | Finds a tariff ID by name and company. | |
| **set\_json\_path(path) / get\_json\_path()** | | Set or get the fall back JSON file path. | |
| **start\_tariffs\_background\_update(interval\_seconds)** | Starts thread to auto-update all tariffs at regular intervals. | |
| **start\_single\_tariff\_background\_update(tariff\_id, time\_interval)** | Starts thread to auto-update one specific tariff. | |
| **get\_latest\_tariffs()** | Retrieves the most recently updated full tariffs (from background thread). | |
| **get\_latest\_single\_tariff()** | Retrieves the most recently updated single tariff (from background thread). | |

## Subclass: **Price**

Handles retrieval and calculation of price components for a tariff.

### Price Methods

| **Method** | **Description** |
| --- | --- |
| **get\_price(tariff\_id, datetime)** | Retrieves all pricing components (fixed, energy, power). |
| **get\_fixed\_price(...)** | Retrieves fixed pricing for a timestamp. |
| **get\_energy\_price(...)** | Retrieves energy pricing for a timestamp. |
| **get\_power\_price(...)** | Retrieves power pricing for a timestamp. |
| **find\_matching\_time\_period(...)** | Finds pricing periods valid for a given timestamp. |

|  |  |
| --- | --- |
|  |  |

## Nested Subclasses of **Price**

**Energy and Power**

The Energy class focuses on energy-based pricing and optimization while the Power class dose the same for power-based pricing.

### Energy and Power Methods

| **Method** | **Description** |
| --- | --- |
| **set\_data(...)** | Loads energy usage data (datetime, kW). |
| **get\_mean(...)** | Computes average kW usage. |
| **get\_duration(...)** | Calculates total time span of usage data. |
| **extract\_price\_value(...)** | Gets numeric price from object or dict. |
| **extract\_price\_function(...)** | Extracts cost function string. |
| **get\_optimal\_start(...)** | Calculates cheapest start time within 24h. |
| **get\_cost(now, kW)** | Computes cost at a specific time. |
| **get\_operation \_cost(start, duration, data)** | Computes total cost over a duration with average energy usage. |
| **start\_background\_cost\_updates(kW, interval)** | Starts background thread to continuously update energy cost. |
| **get\_latest\_cost()** | Retrieves the latest calculated energy cost from the thread queue. |

The methods of the Power subclass are functionally identical to those in Energy, with the key difference being that they access power-specific data using get\_power\_\* functions and use method names related to "power" instead of "energy." It is not necessary for these classes to have identical methods this is simply how they are implemented at the moment. The differences can be found in the ”get cost” functions, where the calculations of mean values differ depending on the tariffs. The implementation prioritizes peaks based on the highest price and calculates their value according to the tariff's structure.

# **Further Development**

Future development should focus on implementing more advanced optimization functions and enhancing them to fully utilize the features of continuous background updates.

Another important direction is integrating the script with **Home Assistant**, including support for sensor handling and recording of user energy and power usage data. This would enable real-time data input and allow the optimization functions to operate based on actual user behaviour and usage patterns.

Important changes to the script will be required to align with upcoming structural changes in how tariff definitions are implemented in future updates to the API specifications from RISE. These changes are not yet present in the current API version but will likely break the get\_tariffs function. Updating the script should be straightforward by regenerating the OpenAPI files using the new specification and adjusting the function to utilize the updated structure.  
  
For free use of all files in project visit [TekverkWattTariff](https://github.com/TekverkWattTariff)/[EnergyTariff\_api](https://github.com/TekverkWattTariff/EnergyTariff_api)**.**