The Journal of Open Source Software

PowerFactory-Tools: A Python Package to Facilitate the Control of DIgSILENT PowerFactory

Sebastian Krahmer [1*¶], **Sasan Jacob Rasti** [1*], **Laura Fiedler** [1], and **Maximilian Schmidt** [1]

1 Institute of Electrical Power Systems and High Voltage Engineering, TUD Dresden University of Technology, Germany ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Kyle Niemeyer

Reviewers:

- @SebastianPalmDD
- @DorotheeNitsch

Submitted: 17 April 2025
Published: unpublished

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

PowerFactory-Tools is a Python package that facilitates the control of PowerFactory, a worldwide used network calculation program. The software provides a well-structured and type-safe interface for PowerFactory, thereby simplifying the development, testing and verification of custom Python scripts. The package also includes a network exporter that converts PowerFactory data into the open-source *Power System Data Model* format. This enables users to access the nodal admittance matrix of the network, which is restricted in PowerFactory. *PowerFactory-Tools* also provides type hints and autocomplete suggestions, safe unit handling, and a temporary unit conversion to default values. The package has been utilised in a variety of research projects. This software is a valuable tool for power system engineers and researchers who require network calculations to be automated and streamlined.

Statement of need

PowerFactory-Tools is a power system affiliated Python package for the control of the commercial network calculation program PowerFactory (DIgSILENT GmbH, 2025). When it comes to calculations based on use case variations or the need for reproducible control actions, PowerFactory can be called and controlled via user-defined Python scripts. PowerFactory-Tools eases developing, testing and verification by providing a well-structured and type-safe Python interface for PowerFactory. This interface is established on top of the PowerFactory-Python-API, but has undergone a process of refinement and augmentation through the incorporation of individually parameterisable functions that prove to be of considerable practical benefit. A common task in respect to case studies that can be implemented more conveniently with *PowerFactory-Tools* is, for example, the automated replacement of generators with one's own templates and their parameterisation.

Furthermore, a main functionality is the network exporter from PowerFactory to the open-source *Power System Data Model* (PSDM) (2023). In terms of network optimisation, user-defined network reduction or stability analysis, users may require an explicitly accessible nodal admittance matrix (NAM) of the network. Since access to this is still restricted for PowerFactory users, exporting the PowerFactory network to a well structured and human readable exchange format is a huge benefit. Due to this, users can (a) export to *PSDM* Python objects and build the NAM by your own without changing the programming language or (b) export to *PSDM*-formatted JSON files, then import these files using the programming language of your choice and build the NAM. It has to mention, that PowerFactory provides a built-in export with DGS, the bidirectional, flexible DIgSILENT data exchange format (ascii, xml, csv, odbc). While it is intended to support GIS and SCADA connections, the drawback is that the DGS export is typeless and not Python native. Due to this, a significant effort for parsing may occur.

PowerFactory-Tools was used in Krahmer et al. (2022), Krahmer et al. (2023), Krahmer et al. (2024) and Fiedler et al. (2025) as well as is currently in use in the research project SysZell, ZellSys and digiTechNetz.

Application Benefits

By implementing a type wrapper for internal PowerFactory element types, users receive type hints and autocomplete suggestions to increase the safety and productivity. Furthermore, *PowerFactory-Tools* guarantee safe unit handling. A temporary unit conversion to default values is automatically performed to have a project setting independent behavior. The units are reset when the interface is closed. During an active connection to PowerFactory, the following units apply: power in MVA (resp. MW, Mvar), voltage in kV, current in kA and length in km.

A broad range of application examples is provided in the *PowerFactory-Tools* repository (2022), which encourage beginners.

Power System Data Model

As previously stated, the *PSDM* constitutes a secondary open-source toolbox that has been developed in conjunction with the *PowerFactory-Tools*, but not exclusively for them. It utilizes a hierarchical structure/schema to describe unique entity relations as well as parameter sets. *PSDM* uses the BaseModel class from Pydantic as a technique for defining schema classes. The PSDM consists of three parts covering different types of information and each part can be stored as a human-readable JSON file: - Topology: plain network model with nodes, edges and connected devices - TopologyCase: information about elements that are disconnected, e. g. out-of-service or via open switches - SteadystateCase: operational case specific information.

A full PSDM-representation of a network can be viewed in the example section of the *PowerFactory-Tools* repository (2022). The following code snippet shows how to use the library to export a PowerFactory 2024 network to the *PSDM* format.

```
pip install ieeh-powerfactory-tools
```

```python
import pathlib
from powerfactory_tools.versions.pf2024 import PowerFactoryExporter
from powerfactory_tools.versions.pf2024.interface import ValidPythonVersion

PF_PATH = pathlib.Path("C:/Program Files/DIgSILENT")
PF_SERVICE_PACK = 2 # mandatory
PF_USER_PROFILE = "" # specification may be necessary
PF_PYTHON_VERSION = ValidPythonVersion.VERSION_3_12
# project name may be also full path "dir_name\project_name"
PROJECT_NAME = "my-pf-project"

with PowerFactoryExporter(
powerfactory_path=PF_PATH
powerfactory_service_pack=PF_SERVICE_PACK,
powerfactory_user_profile=PF_USER_PROFILE,
python_version=PF_PYTHON_VERSION,
project_name=PROJECT_NAME,
) as exporter:
    # Option I: Export to PSDM Python objects
    grids = self.pfi.independent_grids(calc_relevant=True)
    for grid in grids:
```

DRAFT

```python
        grid_name = grid.loc_name
        data = self.pfi.compile_powerfactory_data(grid)
        meta = self.create_meta_data(data=data, case_name="study_case_name")

        # Create the three PSDM base classes
        topology = self.create_topology(meta=meta, data=data)
        topology_case = self.create_topology_case(meta=meta, data=data,
            topology=topology)
        steadystate_case = self.create_steadystate_case(meta=meta, data=data,
            topology=topology)

    # Option II: Export to PSDM-formatted JSON files
    exporter.export(
        export_path = pathlib.Path("export_dir"),
        study_case_names=["study_case_name"],
    )
```

Software Dependencies

The software is written in Python and uses the data validation library pydantic (2025). In respect to the export functionality, the *PSDM* (2023) is used as schema for network entity relations. Ultimately, the responsibility falls upon the user to ensure the accurate compilation of software versions. Should any reader require assistance with this topic, they will find an up-to-date list of compatible software available at the repositories readme. For example, the *PowerFactory-Tools* version 3.2.0 is related to the *PSDM* version 2.3.3 and brings built-in support for PowerFactory version 2022 and 2024.

Acknowledgements

The tool was developed during work related to the projects STABEEL (project no. 442893506), SysZell (funding code 03EI4074D) and digiTechNetz (funding code 03EI6075A), first funded by the Deutsche Forschungsgemeinschaft (DFG, DOI: 10.13039/501100001659), the latter funded by the German Federal Ministry for Economic Affairs and Climate Action.

DRAFT

References

DIgSILENT GmbH. (2025). *PowerFactory*. https://www.digsilent.de/en/powerfactory

Fiedler, L., Schmidt, M., Schegner, P., Reichardt, S., Weisenstein, M., Schmidt, U., Wagner, T., Hänchen, H., Braun, L., & Menke, L. (2025). Digitechnetz – a smart grid platform enabling an active operation of low voltage distribution grids. *IET Conference Proceedings*, *2024*, 904–908. https://doi.org/10.1049/icp.2024.1991

Institute of Electrical Power Systems and High Voltage Engineering - TU Dresden. (2022). *PowerFactory Tools - A toolbox for Python based control of DIgSILENT PowerFactory*. GitHub Repository. https://doi.org/10.5281/zenodo.7074968

Institute of Electrical Power Systems and High Voltage Engineering - TU Dresden. (2023). *Power System Data Model - A data model for the description of electrical power systems*. GitHub Repository. https://doi.org/10.5281/zenodo.7781375

Krahmer, S., Ecklebe, S., Schegner, P., & Röbenack, K. (2022). Analysis of the Converter-Driven Stability of Q(V)-Characteristic Control in Distribution Grids. *5th International Conference on Smart Energy Systems and Technologies*. https://doi.org/10.1109/SEST53650.2022.9898506

Krahmer, S., Ecklebe, S., Schegner, P., & Röbenack, K. (2023). Zur Notwendigkeit von Stabilitätsbetrachtungen von Umrichterinteraktionen bei der Sicherheitsbewertung in Verteilnetzen. *At - Automatisierungstechnik*, *71*(12), 1051–1064. https://doi.org/10.1515/auto-2023-0142

Krahmer, S., Ecklebe, S., Schegner, P., & Röbenack, K. (2024). Application of Stability Analysis of Q(V)-Characteristic Controls Related to the Converter-Driven Stability in Distribution Networks. *IEEE Transactions on Industry Applications*, *60*(3), 5002–5011. https://doi.org/10.1109/TIA.2024.3360023

Pydantic: Data validation using python type hints. (2025). In *GitHub repository*. GitHub. https://github.com/pydantic/pydantic

DRAFT