

# Chapter 24

## Interfaces

### 24.1 Introduction

*PowerFactory* supports a wide set of interfaces. Depending on the specific data exchange task the user may select the appropriate interface.

The interfaces are divided as follows:

- Interfaces for the exchange of data according to *DigSILENT* specific formats:
  - DGS
  - StationWare (*DigSILENT* GmbH trademark)
- Interfaces for the exchange of data using proprietary formats:
  - PSS/E (Siemens/PTI trademark)
  - NEPLAN (NEPLAN AG trademark)
  - ELEKTRA
  - MATLAB (The MathWorks, Inc trademark)
  - INTEGRAL
  - PSS/SINCAL (Siemens/PTI trademark)
- Interfaces for the exchange of data according to standardised formats:
  - UCTE-DEF
  - CIM
  - OPC
- Programming interfaces for integration with external applications
  - C++ API

The above mentioned interfaces are explained in the following sections.

### 24.2 DGS Interface

DGS (**DigSILENT**) is *PowerFactory*'s standard bi-directional interface specifically designed for bulk data exchange with other applications such as GIS and SCADA, and, for example, for exporting calculation results to produce Crystal Reports, or to interchange data with any other software package.

Figure 24.2.1 illustrates the integration of a GIS (Graphical Information System) or SCADA (Supervisory Control And Data Acquisition) with *PowerFactory* via the DGS interface. Here, *PowerFactory* can be configured either in GUI-less or normal mode. When used in GUI-less mode (engine mode), *PowerFactory* imports via DGS the topological and library data (types), as well as operational information. Once a calculation has been carried out (for example a load flow or short circuit), the results are exported back so they are displayed in the original application; which in this example relates to the SCADA or GIS application. The difference with *PowerFactory* running in normal mode (see right section of Figure 24.2.1) is that, besides the importing of data mentioned previously, the graphical information (single line graphics) is additionally imported, meaning therefore that the results can be displayed directly in *PowerFactory*. In this case, the exporting back of the results to the original application would be optional.

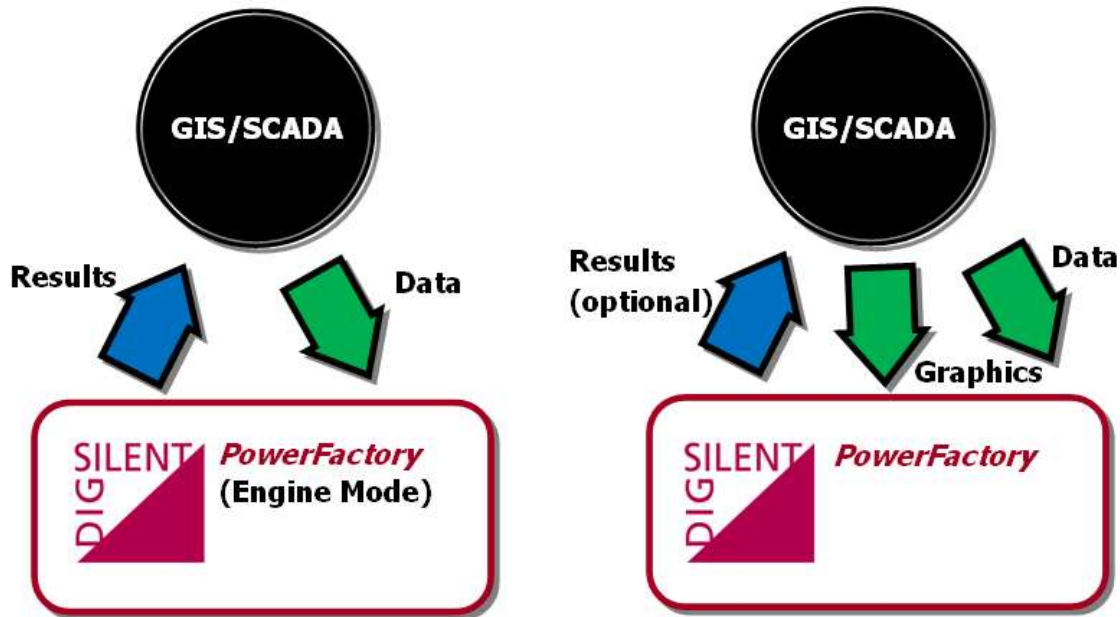


Figure 24.2.1: DGS - GIS/SCADA Integration

Although the complete set of data can be imported in *PowerFactory* every time a modification has been made in the original application, this procedure would be impractical. The typical approach in such situations would be to import the complete set of data only once and afterwards have incremental updates.

### 24.2.1 DGS Interface Typical Applications

Typical applications of the DGS Interface are the following:

- **Importing to *PowerFactory***
  - Data Import/Update into *PowerFactory* from external data sources such as GIS (Network Equipment), SCADA (Operational Data) and billing/metering systems (Load Data) in order to perform calculations.
- **Exporting from *PowerFactory***
  - Performing calculations in *PowerFactory* and exporting back the results to the original application.
- **Integration**
  - Importing data sets to *PowerFactory* from GIS or SCADA, performing calculations, and exporting back results to GIS or SCADA.

### 24.2.2 DGS Structure (Database Schemas and File Formats)

*PowerFactory's* DGS interface is strictly based on the *PowerFactory* data model. Data can be imported and exported with DGS using different file formats and database schemas.

The following databases or file formats are supported:

- **Databases**
  - Oracle DB Server (ODBC client 10 or newer)
  - Microsoft SQL Server (ODBC driver 2000 or newer)
  - System DSN (ODBC)
  - Generic ODBC
- **File Formats**
  - DGS File - ASCII
  - XML File
  - Microsoft Excel File (2003 or newer)
  - Microsoft Access File (2003 or newer)

Important to note here is that the content of the files is the same, the only difference being the format.

---

**Note:** Due to changes in the format, DGS is available in several versions. It is highly recommended to always use the latest available DGS version.

---

The core principle of DGS is to organise all data in tables. Each table has a unique name (within the DGS file or database/table space) and consists of one or more table columns, where generally all names are case-sensitive.

More information on DGS and examples can be accessed by selecting from the main menu *Help* → *Additional Packages* → *DGS Data Exchange Format*

### 24.2.3 DGS Import

To import data via the DGS interface, the general procedure is as follows:

- From the main menu go to *File* → *Import...* → *DGS Format...* which opens the DGS-Import dialog.
- Specify the required options in both the *General* and *Options* pages, and click on the **Execute** button.

When importing DGS files, the user has two options:

1. Importing into a new project. With this option selected a newly generated project is left activated upon completion.
2. Importing into an existing project. If an operational scenario and/or a variation is active at the moment the import takes place, the imported data set will be divided correspondingly. For example importing breaker status (opened/closed) while an operational scenario is active will store this information in the operational scenario.

The following sections describe each of these options.

### 24.2.3.1 General Page

**Import into New Project** By choosing this option, a project will be created where all the DGS data will be stored. The user will have the option of specifying a specific name and location (other than the default).

**Import into Existing Project** By choosing this option, the DGS data will be imported into an already existing project. Here, the data can be selective and its not required that the imported data must be complete. In some cases, most of the objects already exist and only an update is required for some of them.

**Import from** The source of the data to be imported is specified with this option. If a *File Format* source is selected then the location and type of data (DGS, XML, MDB or XLS) must be specified. If a *Database* source is selected, then a service, User and Password information is required (the SQL server option will require an extra Database information).

### 24.2.3.2 Options Page

The visible options depend on the DGS version being used, and on the users choice of the *Import Format*.

#### Options for DGS version 4.x

***Predefined Library***

A predefined library located somewhere else in the database can be selected. The option of copying the library into the project is available.

***Create Switch inside Cubicle***

In cases where the source data has no switches defined inside the cubicles, the enabling of this option will create the switches automatically during the import. If switches already exist in a certain cubicle, the creation of switches in that particular cubicle is ignored.

***Replace non-printable characters***

If the source data contains not allowed characters (~, ?, etc.), they are replaced by an underscore character.

#### Options for DGS version 5.x

***Open single line diagram(s)***

If the DGS source contains graphics objects for single line diagrams, these will be opened automatically after import.

***Dataset Import (only available if a Database Schema is selected as Import Format)***

For DGS version 5 or higher, a labelled version of the data in the source data base can be selected for import. The labelled versions are mainly used to chose a time-dependent state of the data. The label name is input in the field *Label*.

#### Options for DGS version 6.x

The options for DGS version 5.x are available for DGS version 6.x as well. In addition, the options described below can be configured.

***Global type library***

The DGS import in earlier versions only allows for references to existing objects within the active project. With the DGS version 6.x, a global type library can be selected. Elements in the DGS source can refer to the *foreign key* of types in the selected global library.

***Partial Import (only available if a Database Schema is selected as Import Format)***

For DGS version 6 or higher, labelled regions can be selected for import in the source data base.

The labelled regions can be used to select specific voltage levels or sub-grids from the bulk data set. The label names are input in the field *Labels* separated by commas.

The option pages for all DGS versions contain the field

***Additional Parameters***

This field is specified for internal use only. No extra information is required by the user.

More detailed information on DGS and examples can be accessed by selecting from the main menu *Help* → *Additional Packages* → *DGS Data Exchange Format*

## 24.2.4 DGS Export

In contrast to the *DGS Import*, where it is not relevant if a project is active or not; the *DGS Export* is based on what information is active at the moment the export takes place. In other words, only the active project, with the corresponding active *Study Case*, active *Scenario*, and active *Variations* are exported (objects are exported in their current state). Furthermore, the export can be fully configured, meaning that the user has the option of selecting the amount of information to be exported per class object. In general, the following data can be exported:

- Element data
- Type data
- Graphic data
- Result data (e.g. load flow results)

To export data via the DGS interface, the general procedure is as follows:

- Create an Export Definition
- Activate the project to be exported, considering which *Study Case*, *Scenario* and *Variations* should be active.
- From the main menu go to *File* → *Export...* → *DGS Format...* which opens the DGS-Export dialog.
- Specify the required options in both the *General* and *Options* pages, and click the **Execute** button.

The following sections describe each of these options.

### 24.2.4.1 General Page

**DGS Version**

Version of the DGS structure.

**Format**

Output format. Either as ASCII, XML, MS Excel or MS Access file (for Excel or Access, Microsoft Office must be installed on the computer) or as Oracle, MS SQL Server, ODBC DSN or generic ODBC databases (respective data base drivers must be installed.).

**File Name or Data Base Service**

Depending on the *Output Format*, a file name for the output file or the data base service and user access information are required.

**Insert Description of Variables**

If checked, a description of the column headers is included in the output file (only available for ASCII, XML and MS Excel).

**Variable Sets**

Select the variable set definition for export. The data exported will be according to the variable definition specified (see the explanation at the beginning of the section). It is required to select a folder that contains the monitor variable objects (*IntMon*) related to each class that is to be exported.

**24.2.4.2 Options Page**

The visible options mostly depend on the DGS version 4.x, 5.x or 6.x and on the users choice of the *Export Format*.

**DPL Script**

Independent of the DGS version, the user can select a DPL script. This DPL script is automatically executed before DGS export.

**Options for DGS version 5.x*****Allow hierarchy and references of exported objects to be incomplete***

If this option is set, error messages because of references to external objects (e.g. types in the global library) that will not be exported are omitted. Furthermore, the export of data classes is possible although their parent-folder class is not contained in the *variable definition set*.

***Allow user-defined table names***

With this option, a prefix and suffix can be added to all table names on DGS export. The prefix or suffix is defined in the field *Additional Parameters*.

***Export as dataset (available for export to data base formats only)***

This option is used to write the exported data to a specifically labelled data set (session state) into the data base. A label identifier must be given. Optional, a description can be added.

**Options for DGS version 6.x**

All options for DGS version 5.x are available for DGS version 6.x. In addition, the following options exist:

***Export as Update***

DGS 6.0 supports an explicit marker *OP* to mark a data record to be created (C), updated (U) or deleted (D) on DGS import. On export, the user can chose to mark all data as *Update* by means of this option. Otherwise the exported data are marked for *Create*.

***Categorisation of data for partial import (available for export to data base formats only)***

This option is used to define for each grid (ElmNet) in the active project a *data part* in the data base. In addition, certain data types and elements are labelled with respect to their meaning in the context of this partial export: global types, local types, boundary set (branches connecting elements that belong to different grids), other elements (e.g. characteristics). The names of these labels are input in the respective fields (all fields should be filled in). In addition, a global library folder can be selected to include referenced types from this library on DGS export.

**Options for DGS version 4.x*****Export Grid Name***

If this option is set, a column "Grid" is added to all tables containing network elements.

***Export Cubicles***

With this option, cubicles are exported. Cubicles describe the connectivity of nodes and branches. The export of cubicles can be omitted if the grid topology is not needed (e.g. for result export).

***Export Graphical Data***

The user can select one of three options for the export of graphical data:

**No** No graphical data are exported.

**Yes, with Graphic (*IntGrfnet*) Names** Graphical data are exported. All graphic object tables contain a column for the name of the graphic scheme (*IntGrfnet*).

**Yes, without Graphic (*IntGrfnet*) Names** Graphical data are exported. The graphic scheme is not referenced by the exported graphic objects.

The option page contains independent of the DGS version always the field

#### **Additional Parameters**

This field is specified for internal use only. No extra information is required by the user.

More detailed information on Variable Sets definitions (*IntMon*) can be accessed by selecting from the main menu *Help* → *Additional Packages* → *DGS Data Exchange Format*.

## 24.3 PSS/E File Interface

Although both import and export functions for PSS/E files are integrated commands of *PowerFactory*, the export function is licensed separately. For more information on prices and licensing contact the sales department at mail@digsilent.de.

PSS/E Import supports versions 23 to 32 and can be performed by going to the main menu and selecting *File* → *Import...* → *PSS/E*.

In the same manner, and provided the appropriate licensing exists, a project can be exported in PSS/E format by selecting from the main menu *File* → *Export...* → *PSS/E*.

### 24.3.1 Importing PSS/E Steady-State Data

*PowerFactory* is able to convert both steady-state data (for load-flow and short-circuit analysis) and dynamic data files. It is good practise to first import the steady-state data (described in this section), then to add the dynamic models (described in Section 24.3.2: Import of PSS/E file (Dynamic Data)).

Before starting the next steps for importing a PSS/E file, make sure that no project is active. Once this has been confirmed, select from the main menu *File* → *Import...* → *PSS/E*. By doing so, the *Convert PSS/E Files* command dialog will be displayed, asking the user to specify various options.

#### 24.3.1.1 General Page

**Nominal Frequency** Nominal frequency of the file to be Converted/Imported.

#### **PSS/E File Type**

**PSS/E Raw data** Location on the hard disk of the PSS/E raw data file. By default the program searches for \*.raw extensions.

**Sequence Data** Location of the PSS/E sequence data file. By default the program searches for \*.seq extensions.

**Add Graphic Files** Location of the PSS/E drw files on the file system. Again by default the programs searches for files with extension \*.drw.

---

**Note:** After the Conversion/Importing has finished, the resulting project will contain a graphics folder where all of the PSS/E drw converted graphics will be stored. The user must therefore relocate each one of them to the corresponding diagram folder.

---

**Save converted data in**

**Project** The project name that will be assigned to the converted/imported file in *PowerFactory*.

**in** Location in the Data Manager tree where the imported file will be stored.

The following topics: *Dyn. Models Data*, *Composite Frame Path*, *DSL - Model Path*, *Parameter Mapping*; are not used for the import of steady-state data and will be explained in the dynamic import Section [24.3.2](#).

**24.3.1.2 Options Page**

- **Convert only sequence data file** - With this option enabled, the converter will only add the sequence data to an existing project.
- **Convert only dynamic models file** - With this option enabled, the converter will only add the dynamic data file to an existing project (only for dynamic data import).
- **Convert only graphic file** - With this option enabled, the converter will add only a single-line diagram to an existing project.
- **Only convert file (no DB action)** - Internal option used for syntax check and error messages during conversion. Normally this box should be left unchecked.
- **Output only used dynamic models** - Displays a list of used dynamic models (only for dynamic data import).
- **Unit of 'LEN' for lines in miles instead of km** - With this option enabled, all lengths will be interpreted in miles in the PSS/E raw files.
- **Consider transformer phase shift** - With this option enabled, transformer phase shifts will be considered. This option is recommended and activated by default.
- **Convert Induction Machines (Generators: P<0)** - With this option enabled, all generators in the raw data file that have negative active power will be converted to asynchronous machines. For transmission grids the option should be disabled for proper modelling of phase shift generators.
- **Automatic 3-W. Transformer detection/conversion** - In versions <27, PSS/E does not handle 3-winding transformers as a dedicated model. In such cases, the 3-winding transformer is modelled with three 2-winding transformers connected to a busbar. If this option is selected, the converter will try to detect the existence of three 2-Winding Transformers connected to a busbar. If any candidates are available, *PowerFactory* will replace them by a 3-Winding Transformer. The detection algorithm uses the impedances and the voltage control of the transformers as reference. From version 27 onwards PSS/E supports the 3W-transformer model, so that *PowerFactory* does not start an automatic detection of 3W-Trf modelled as 2W-Trfs.
- **Convert capacitive line shunts to line susceptance B'** - If a line has line shunts the converter adds automatically the line shunt capacitance to the C1' (B1') in the *PowerFactory* line type.
- **Convert Common Impedance as Transformer** - If this option is selected, the Common Impedance in PSS/E may be converted to a *PowerFactory* common impedance or to a transformer.
- **Convert Series Capacitance as Common Impedance** - Older versions of PSS/E do not handle series capacitances as a dedicated model. These elements therefore are represented by lines with negative reactances. During the conversion, *PowerFactory* detects these branches and converts them to series capacitances (by default) or to common impedances (when this option is active).
- **Convert off-nominal turn ratio to transformer tap** - Transformer ratios different from the rated ratio are automatically converted to a transformer type using taps, including the correct tap position.



- **Busbar naming: 'PSSE\_NAME'** - With this option enabled, the busbars are named similar to the PSS/E raw data file (without bus number).
- **Branch naming: 'BUSNAME1\_BUSNAME2\_ID'** - With this option enabled, the branches are named as the name of the busbars + ID.

**Additional Parameters** - This field is specified for internal use only. No extra information is required by the user.

### 24.3.1.3 Graphical Options Page

- **Rotate with respect to busbars** - The converter will rotate the graphical layout in case of the majority of busbars being in vertical or horizontal position.
- **Snap coordinates to grid** - The converter will snap to grid all objects in the single line graphics.
- **Transformer Symbol according to IEC** - This options lets the user choose the transformer symbol as IEEE (default) or IEC representation.
- **Scaling factor** - The graphic files are scaled according to the scaling factor shown.

### 24.3.2 Import of PSS/E file (Dynamic Data)

As explained in Section 24.3.1 it is good practise first to import the steady-state data and then to add the dynamic model data.

Some dynamic models used in PSS/E are available in the Global Library. User defined dynamic models should be modelled in *PowerFactory* before importing the program. In this case, an important condition for successful file conversion is that all DSL models used during the conversion process should be stored in the same model library folder.

If the original library should use specific folders for the different types of controllers (AVR,PCO,PSS, etc.), the user should copy all of the models into the same library folder, in this case the recommendation is to copy the dynamic models from the global library into the library where the rest of the user defined models are located. After the conversion, the user may re-arrange the models.

The procedure to start the import of dynamic network data is very similar to the import of steady-state data. Some parameter adjustments have to be made.

#### 24.3.2.1 General Page - Dynamic Models

On the *General* page of the import dialog the following topics have to be specified:

**Dyn. Models Data** - Location of the PSS/E Dynamic Models data file. By default the program searches for \*.dyn and \*dyr extensions.

**Use Standard Models from global library** - If this option is enabled, *PowerFactory* will automatically point to the Standard Models library located in the Global library. There will be no need of selecting the composite Frame Path and DSL Model Path.

**Composite Frame Path** - Location in the *PowerFactory* data base where the composite frames are stored (Standard Models/Composite Models Frames...).

**DSL - Model Path** - Location in the *PowerFactory* data base where the DSL models are stored (Standard Models...).

**Parameter Mapping** - Location of the *PowerFactory* mapping file. This is an option that normally will not have to be defined by the user. By default *PowerFactory* will automatically set up its own

internal mapping file. This file defines how to translate the PSS/E internal models into *PowerFactory* models, including the mapping of controller parameters. For automated conversion of user-defined PSS/E controllers the mapping file may be customised.

### 24.3.2.2 Import Options Page - Dynamic Model Import

On the *Options* page of the import dialog the following options should be considered:

**Convert only dynamic models file** - With this option enabled, the converter will only add the dynamic data file to an existing project.

**Output only used dynamic models** - Displays a list of used dynamic models.

### 24.3.3 Exporting a project to a PSS/E file

This function allows the export of the network model in PSS/E format. The export comprises both steady-state and dynamic data sets. The correct conversion of dynamic models is only possible for the standard IEEE models. Models which the user implemented in *PowerFactory*'s DSL can not be automatically translated and must be modelled as user-defined controller types separately in PSS/E.

To export a project in PSS/E format select *File* → *Export...* → *PSS/E* from the main menu.

#### 24.3.3.1 Export General Page

**RAW Conversion File** - Path and file name for the PSS/E RAW file, containing the symmetrical description of the model.

**SEQ Conversion File** - Path and file name for the PSS/E SEQ file, containing the additional description of the model necessary for unbalanced conditions.

**DYN Conversion File** - Path and file name for the PSS/E DYN file, containing the dynamic models of the project.

**PSS/E Version** - Version of the exported PSS/E file (25 to 32).

#### 24.3.3.2 Export Options Page

**Convert Motors to Generators if P<0** - With this option enabled, all asynchronous machines in generator mode will be converted to synchronous machines.

**Export branch as single equivalent line** - Selecting this option will convert the branch models to an equivalent line.

**Convert SVS to generator** - This option defines how the SVS elements will be exported. Three options are available:

- **No**: the SVS elements won't be exported.
- **Only voltage controlled**: will convert the SVS elements with control mode set to *Voltage Control* (Load Flow page of the element) to generator models.
- **Always**: all the SVS elements will be converted to generator models.

**Base Apparent Power** - Base for the power values given in per-unit system.

**Min (Zero) Impedance Branch** - Minimum impedance for ideal connections.

**PSS/E Bus Number** - This option defines the naming convention when exporting terminals *ElmTerm*. Three options are available:

- **Automatic:** the number assigned will be according to the name (in ascending/alphabetical order).
- **Use Serial Number:** the serial number information stated in the *Description* page of each terminal will be used for assigning the PSS/E bus number.
- **Use Characteristic Name:** the characteristic name information stated in the *Description* page of each terminal will be used for assigning the PSS/E bus number.

**Export PSS/E-Area index as** - The way the Area index is defined in PSS/E is defined here, two options are available:

- **Grids:** the exported file will have the areas defined according to the Grids defined in the *PowerFactory* model.
- **Areas:** the exported file will have the areas defined according to the Areas defined in the *PowerFactory* model.

**Additional Parameters** - This field is specified for internal use only. No extra information is required by the user.

## 24.4 ELEKTRA Interface

*PowerFactory* offers the user the possibility to import different types of ELEKTRA files. The files supported for import are as follows:

- **Elektra network models**
  - Element data (\*.esd) from Elektra Version 3.60 to 3.98, which contain the topological and electrical data of the elements in the grid.
  - Network diagrams (\*.enp) from Elektra Version 3.92 to 3.98, which contain the graphical representation of grids.
- **Elektra equipment type library**
  - Type data (\*.dat), which contains equipment types.

### 24.4.1 Import of Elektra Data

The general way to import data via the Elektra interface is as follows:

- From the main menu, select: *File* → *Import* → *Elektra...*. The Elektra-Import dialog will be displayed.
- Select the desired options and click on the **Execute** button.

---

**Note:** The Elektra import cannot be executed if Elektra is open. Close the software before executing the import.

---

The import will be executed regardless of whether a project is activated or not. At the end of the import, the project will be activated. If there is another project activated while importing the Elektra data, *PowerFactory* will deactivate the active project, and activate the newly-created or selected project (according to the settings).

The options available in the Elektra import dialog are described in the following section.

## 24.4.2 General Settings

### Import into

**New project** A new project will be created in which all of the Elektra data will be stored. The user can select a name and a storage location. Different versions of the same network model should be stored in new projects.

**Existing project** Elektra data will be imported into an existing project. Use this option if grids from different regions will be connected and should be calculated together in one project.

### Files

**Kind of data** Within the Elektra import, *Element/graphic data* (data type \*.esd and \*.enp) or *Type data* (data type \*.dat) can be imported, according to the selection.

**Element data** If *Element/graphic data* is selected, set the storage location of the Elektra element data by clicking the "... " icon.

**Graphical data** Add graphical data for the element. Select *Delete* to remove the data from the list.

**Type data** If *Kind of Data: Type data* is selected, click on *Add* to select the Elektra type library (\*.dat) for import. Repeat this step if more type libraries should be added to the import. Select *Delete* to delete single files from the selection.

## 24.4.3 Advanced Settings

On the *Advanced* settings page, the following options can be used to simplify the imported network. In addition, there are two options to activate the import of coupling impedances and active/reactive power characteristics (Q(P) curves).

### General Options

**consider graphical node representation** If a node is set to *Internal Node* in the Elektra element data, *PowerFactory* will also set the node to *Internal Node*. That is, the *usage* of the node in *PowerFactory* is set according to the *usage* in Elektra element data.

**create detailed busbar systems for single busbars** By default, a detailed representation of substations is generated for all Elektra busbars in a *PowerFactory* substation. This is done regardless of whether it is a single or double busbar. This option should be chosen to set locations where only single busbars exist, to single busbars in *PowerFactory*.

**create auxiliary graphic objects in annotation layer** Objects in the Elektra open graphic (open texts, memos, rectangles, pictures, ...) will be transformed into the *annotation layer* of *PowerFactory* by default. These layers can be scaled and changed in *PowerFactory*. As an alternative, graphical objects can be split into parts in the import process. This leads to limited options in later adaptations of the objects.

**create element names with reference to the node name** In *PowerFactory*, every element must have a unique name. To ensure this uniqueness for the Elektra import, the names are comprised of the following parts: *Elektra element name - Elektra name of terminal 1 - Elektra name of other terminal*. If this name has more than 40 letters it will be shortened.

**coupling impedances** Coupling impedances between adjacent overhead lines in Elektra network data are converted into corresponding tower elements (*ElmTow*) and tower types *TypTow* in *PowerFactory*.

**convert Q(P) curves** The reactive power behaviour of generator units or synchronous machines in Elektra data can be given as an active/reactive power characteristic. These curves are converted into a Q(P) characteristic in *PowerFactory* and assigned to the corresponding static generator/s or synchronous machine/s.

### Individual scaling factors at Elektra node elements

Active and reactive power can be modified through scaling factors in Elektra on different layers. These factors are transformed into scalar *PowerFactory* characteristics, upon import of Elektra element data. If there are many individual scaling factors for Elektra node elements, one of the following options can be chosen. These options may assist in reducing the number of characteristics in *PowerFactory*.

**Ignore all scaling factors** The factors for active and reactive power for Elektra node elements are ignored within the data import. The results of the load flow calculation are influenced by this option.

**Calculate resulting power quantities** The multiplication of the active and reactive power by the Elektra node element factor is transferred into *PowerFactory*.

**Create individual scale factor objects** For all factors for Elektra node elements that are set to a value different to '1', corresponding scalar characteristics are created in *PowerFactory*. This is the default option.

**Additional Parameter** This field is for internal use. No additional information is required from the user.

### 24.4.4 Importing Elektra Network Data

To import Elektra network data, choose *Kind of data: Element/graphic data*. The following combinations of element and graphic data exist:

1. Selection of Elektra element data (\*.esd) without graphic data  
The element and topological data from the \*.esd file will be imported. Type data for the element data will be created. There is no creation of a network diagram.
2. Selection of Elektra element data (\*.esd) and one or more corresponding graphic files (\*.enp)  
The included topological and type data from the \*.esd file will be imported. Type data for the element data will be created. Additionally, a network diagram for every selected Elektra graphical data will be created and elements are linked to the graphical objects (if present in both files).
3. Selection of Elektra graphical data (\*.enp), without element data.  
If only graphical data has been selected, for each graphic file one network diagram will be created. From the topological information in the \*.enp file, network data will be created. This network data does not contain technical parameters or type references.

### 24.4.5 Importing Elektra Type Data

To import Elektra type data, select one or more \*.dat files.

In the folder *Library/Equipment Type Library* from the import project, a new Equipment Library will be created for each file and relevant kind of element.

If the successfully imported type data should be used in *PowerFactory*'s global library, continue as follows:

1. Change the user to Administrator by selecting *Tools* → *Switch User...* → *Administrator* via the main menu.
2. Open the *PowerFactory* Data Manager, and create a new folder of type *Library* within the directory Database.
3. Copy the Equipment Library from the import project into this folder.

### 24.4.6 Output Window

During the import the following information is provided in the output window:

- Network elements which do not coexist in the Elektra element and in the Elektra graphical data (multiple entries while importing multiple graphical files are possible).
- Network elements which are generated from power ratings in Elektra nodes.
- Coupling objects between different locations, which cannot be converted.
- Graphical objects whose names are adapted during import.
- Inconsistent or incomplete element parameters.

## 24.5 NEPLAN Interface

*PowerFactory* offers to the user the option of importing different types of NEPLAN files. The files supported for importing are the following:

- **NEPLAN 4**
  - Project File Data (\*.mcb) containing the topological, electrical and graphical data.
  - Line Data Type (\*.ldb) containing the line type information.
- **NEPLAN 5**
  - Node Table (\*.ndt) containing the node data, such as rated voltages and loads.
  - Element table (\*.edt) containing the branch data, such as lines and transformers.
  - GIS/NMS Interface (\*.cde) containing the graphical information of all the networks which are part of the NEPLAN project.

### 24.5.1 Importing NEPLAN Data

To import data via the NEPLAN interface, the general procedure is as follows:

- From the main menu go to *File* → *Import...* → *Neplan...* which opens the NEPLAN-Import dialog.
- Specify the required options and click on the **Execute** button.

The NEPLAN data import always creates a new *PowerFactory* project. Once the import process has been executed, the newly generated project is left activated upon completion.

Independent of the NEPLAN file version (4 or 5), the user has the option of importing the data with or without graphical information. That is, if the user selects importing the data without graphical information, only the topological and electrical data will get imported, and no single line graphic will be generated. In order to import NEPLAN 5 graphics, the path to the NEPLAN files should not contain spaces.

#### Importing NEPLAN 4 Files

When importing NEPLAN 4 files, the user has basically two options:

1. Selection of a \*.mcb file.  
If the user selects this type of file and if a corresponding \*.ldb file is present (should be in the same directory where the \*.mcb is stored), then the information of both files gets imported. If only the \*.mcb file exists, then only the information regarding this file is imported (which can also contain line data).
2. Selection of a \*.ldb.  
If the user selects this type of file only the information regarding this file (line data) is imported.

## Importing NEPLAN 5 Files

When importing NEPLAN 5 files, the user is only required to select the \*.ndt. By doing so, the corresponding \*.edt file is automatically imported also. This basically means that a \*.edt file must be present otherwise the import will not be executed. The \*.cde file is however optional. Additionally, all three files must have the same name and must be in the same directory! As a recommendation, create a separate folder and place all the files there.

The following section describes each of the NEPLAN import dialog options.

### 24.5.1.1 General Settings

#### File Type

**Neplan Data** Location on the hard disk of the NEPLAN data file. Three types of files are available: \*.mcb, \*.ldb and \*.ndt.

#### Save converted data in

**Project** The project name that will be assigned to the converted/imported file in *PowerFactory*.

**in** Location in the Data Manager tree where the imported file will be stored.

#### Common Conversion Settings

##### Automatic busbar system detection

**Import Graphic Information** If this option is enabled then the graphical information is imported and the single line diagram is generated. In case of NEPLAN 5 import the \*.cde file is required.

##### Graphic Import Options (only for NEPLAN 5 import)

**Additional Rotation Angle for 1-port Elements (deg)** If a value different than 0 is stated, then the single port elements (loads, generators, motors, etc.) are rotated counter clockwise (degrees) with respect to the original position.

**Automatically scale to A0** If this option is selected, then the graphic is rescaled according to the A0 page format.

##### User defined scaling factor

##### Additional Parameters

This field is specified for internal use only. No extra information is required by the user.

## 24.6 INTEGRAL Interface

*PowerFactory* offers the user the option to import Integral files for Load Flow and Short Circuit analysis. The following files are supported:

- \*.dvg
- \*.dtf
- \*.xml

Furthermore Integral files can be export as \*.xml files.

### 24.6.1 Importing Integral Data

To import Integral data, the procedure is as follows:

- From the main menu go to *File* → *Import...* → *Integral...* (this will open the Integral import dialog).

In the '**Save converted data in**' field the user can enter a project name, and the *PowerFactory* user for this project can be selected. The Integral data import always creates a new *PowerFactory* project. The \*.xml Integral files contain graphical information. However, for older Integral files with the ending \*.dvg and \*.dtf it is necessary to select graphical data with the ending \*.bild.

More information about the Integral Import is available in the German version of the User Manual.

## 24.6.2 Export Integral Data

The Integral export converts the *PowerFactory* project into an \*.xml file in Integral format. Therefore '**XML Data**' must be defined as the path where to store the xml file. If the ending .xml is not given, it will automatically added.

More information about the Integral Export is available in the German version of the User Manual.

## 24.7 PSS SINCAL Interface

*PowerFactory* offers the user the option to import MS Access database files from PSS SINCAL for Load Flow and Short Circuit analysis. The following files are supported:

- \*.mdb

### 24.7.1 Importing PSS SINCAL Data

The procedure to import PSS SINCAL data is as follows:

- From the main menu go to *File* → *Import...* → *Sincal...* (this will open the PSS SINCAL import dialog).
- Select the file location of the MS Access database file of the SINCAL project (usually named *database.mdb*) in the field *Database name*.
- In the *Save converted data in* field, the user can enter a project name, and the *PowerFactory* user for this project can be selected.

The PSS SINCAL data import will always create a new *PowerFactory* project.

The SINCAL \*.mdb database files contain graphical information. This information is converted into a *PowerFactory* network diagram.

## 24.8 UCTE-DEF Interface

In *PowerFactory*, both export and import of **UCTE-DEF** (**U**nion for the **C**o-ordination of **T**ransmission of **E**lectricity - **D**ata **E**xchange **F**ormat) is supported. The UCTE interface is currently intended for importing/exporting grid data of a country belonging to the former UCTE community.

The data contained in these files correspond basically to load flow and short circuit (3 phase) type data. Furthermore, it only considers specific UCTE voltage levels according to voltage level codes, as well as UCTE specific country codes, such as DK for Denmark, P for Portugal, etc.



Important to note here is that from 1<sup>st</sup> of July 2009, **ENTSO-E** (European Network of Transmission System Operators for Electricity) took over all operational tasks of the 6 existing TSO associations in Europe, including the Union for the Coordination of Transmission of Electricity (UCTE).

For more information related to the UCTE format, refer to the ENTSOE website: <https://www.entsoe.eu>

### 24.8.1 Importing UCTE-DEF Data

To import data via the UCTE interface, the general procedure is as follows:

- From the main menu go to *File* → *Import...* → *UCTE...* which opens the UCTE-Import dialog.
- Specify the required options and click on the **Execute** button.

Once the import process has been executed, the project (new or existing) is left activated upon completion.

The following section describes each of the UCTE import dialog options.

#### 24.8.1.1 General Settings

##### Import into

**New Project** By choosing this option, a project will be created where all the UCTE data will be stored. The user will have the option of specifying a specific name and location (other than the default).

**Existing Project** By choosing this option, the UCTE data will be imported into an already existing project.

##### File Type

**Add UCTE Files** Location on the hard disk of the UCTE files. Two types of files are available: \*.uct and \*.ucte.

##### Options

**Import for DAF process** With this setting the user has the option to import the Day Ahead Forecast.

**Convert negative loads to generators** With this option enabled, negative loads defined in the UCTE file will be converted to generators in the *PowerFactory* model.

**Convert transformer equivalent to common impedance** With this option enabled, transformer equivalents defined in the UCTE file will be converted to common impedances in the *PowerFactory* model.

**Ignore reactive power limits for generators** With this option enabled, the reactive power limits of the generators defined in the UCTE file will be ignored .

**Additional Parameters** This field is specified for internal use only. No extra information is required by the user.

### 24.8.2 Exporting UCTE-DEF Data

As in the other export interfaces, the *UCTE Export* is based on the **active** project at the moment the export takes place. To export data via the UCTE interface, the general procedure is as follows:

- Activate the project to be exported, considering the which *Study Case*, *Scenario* and *Variations* should be active.
- From the main menu go to *File* → *Export...* → *UCTE...* which opens the UCTE-Export dialog.

- Specify the required options, and click on the **Execute** button.

The following section describe each of these options.

### 24.8.2.1 General Settings

#### File Type

**UCTE Data** Location on the hard disk where the UCTE files will be stored. Two types of files are available: \*.uct and \*.ucte.

**Grids** Selection of which grids to export.

#### Options

**Export UCTE voltage >=** Only the elements having a voltage greater than the UCTE voltage specified are exported.

**Export branch as single equivalent line** By enabling this option the export will convert the *PowerFactory* branch definitions into single equivalent lines.

**Use first character of characteristic name as branch order code** If checked, the characteristic name (first character) is used in the branch order code of the exported UCTE file.

**Additional Parameters** This field is specified for internal use only. No extra information is required by the user.

## 24.9 CIM Interface

In *PowerFactory*, both export and import of **CIM** (**C**ommon **I**nformation **M**odel) is supported. The CIM interface is currently intended for importing/exporting the following profile:

- ENTSO-E 2009

(Options “ENTSO-E 2010” and “ENTSO-E 2009 Dynamic Models” in the drop-down menu relate to profiles which were never formally released. Therefore, although these options are available to the user, they are not supported.)

CIM is defined in IEC-61970, and its purpose is to allow the exchange of information related to the configuration and status of an electrical system.

For information relating to CGMES, please see section [24.10](#) below.

### 24.9.1 Importing CIM Data

To import data via the CIM interface, the general procedure is as follows:

- From the main menu go to *File* → *Import...* → *CIM...* which opens the CIM-Import dialog.
- Specify the required options and click on the **Execute** button.

Once the import process has been executed, the project (new or existing) is left activated upon completion.

The following section describes each of the CIM import dialog options.

## 24.9.2 General Page

### Import into

**New Project** By choosing this option, a project will be created where all the CIM data will be stored. The user will have the option of specifying a specific name and location (other than the default).

**Active Project** By choosing this option, the CIM data will be imported into the active project.

### Import from

**Profile** Currently the profile ENTSO-E 2009 is supported.

**separated Files** With this setting the user has the option to import the equipment, topology and solved state files separately.

**CIM File** Location on the hard disk of the CIM files. Two types of files are supported: \*.zip and \*.xml.

**Additional Parameters** This field is specified for internal use only. No extra information is required by the user.

## 24.9.3 Exporting CIM Data

As in the other export interfaces, the *CIM Export* is based on the **active** project at the moment the export takes place. To export data via the CIM interface, the general procedure is as follows:

- Activate the project to be exported, considering which *Study Case*, *Scenario* and *Variations* should be active.
- From the main menu go to *File* → *Export...* → *CIM...* which opens the CIM-Export dialog.
- Specify the required options, and click on the **Execute** button.

The following sections describe each of these options.

### 24.9.3.1 General Page

#### Export to

**Profile** Currently the profile ENTSO-E 2009 is supported.

**separated Files** With this setting the user has the option to export the equipment, topology, and solved state files separately.

**CIM File** Location on the hard disk where the CIM files will be stored. Two types of files are supported: \*.zip and \*.xml.

#### Export Selection

**Grids** Selection of which grids to export.

**Border Nodes Grid** Selection of the grid which contains the X-nodes.

## 24.10 CGMES Tools

The CGMES Tools provide an additional interface to CIM. These tools are accessible via the main menu in *PowerFactory* under *Tools* → *CGMES Tools*. This section describes these tools and uses the following naming conventions:

- **Model folder** stores all CIM data.
- **Archive** contains all corresponding CIM Models.
- **CIM Model** stores all data contained in a single instance file (mainly CIM Objects and namespaces).
- **CIM Object** stores all data contained in a single object.

The CGMES Tools separates each action (i.e. the import and export of CIM data) into two steps. To import data from a CIM file (XML format) into *PowerFactory*, the first step is to import the CIM data into CIM objects (*CIM Data Import*). This offers the user the possibility to directly interact with the CIM data from within *PowerFactory*. The second step is to convert the CIM objects into a grid model (*CIM to Grid Conversion*). To export grids from *PowerFactory*, they must first be converted to CIM objects (*Grid to CIM Conversion*). These may still be modified if required, and also directly reimported. The newly-created CIM objects can then be exported as a ZIP or XML file in conformity with the CIM data structure (*CIM Data Export*).

### 24.10.1 CIM Data Import

The CIM Data Import is accessible in *PowerFactory* under *Tools* → *CGMES Tools* → *CIM Data Import*. The following options for the import location are available:

- **New archive in new project** creates a new project with the given name and imports the data into an archive with the same name.
- **New archive in existing project** imports the data into a new archive with the given name; it requires an active project.
- **Existing archive in active project** imports the data into the given archive; data models that are already contained in the archive will not be modified.

#### Import of instance data belonging to a profile (XML file)

By selecting “**New archive in new project**”, the name of the project and the file to import can be specified. Upon clicking the **Execute** button, the content of the XML file will be imported. This step results in an active project containing the “CIM Model” folder, which itself has a single archive. This archive contains the model representations from the XML file.

#### Import of multiple profiles instance data (ZIP file)

By selecting “**New archive in new project**”, the name of the project and the file to import can be specified. Upon clicking the **Execute** button, the content of the ZIP file will be temporarily extracted and imported. This step results in an active project containing the “CIM Model” folder, which has a single archive. This archive contains the model representations from the ZIP file.

#### Import of multiple files

To import several files in a row, the destination **Existing archive in active project** must be used. The archive created by the first step must be selected as the “Path” for all subsequent files. This will import the data into the archive provided.

### 24.10.2 CIM Data Export

The CIM Data Export is accessible via *Tools* → *CGMES Tools* → *CIM Data Export* or *right-click* → *CIM Data Export*. The archive selected as “Source data” will be exported. This can be fine-tuned by an option for each available profile instance.

The option “Create archive for each CIM model” can be used to match the DACF and D2CF requirements by ENTSO-E to upload each individual profile within a separated zip file.

In both cases the naming rules can be defined per profile. As these rules might be different for various processes, the naming rules can be configured on the “Advanced Options” page according to the individual needs for each profile.

### 24.10.3 CIM to Grid Conversion

The CIM to Grid Conversion is accessible under *Tools* → *CGMES Tools* → *CIM to Grid Conversion* or *right-click* → *CIM to Grid Conversion*

To convert all data contained in an imported archive, select “Source Archives” followed by **Execute**. This will additionally consider any valid difference models contained in the archive. To import a base model only, the difference models must be deleted before conversion. If only specific profile information should be exported, select “Convert selected profiles” and specify those that should be considered. It should be noted that the resulting subset of profiles still needs to be complete in terms of dependencies.

Additional information (e.g. SSH data) can be added to already converted models (i.e. EQ/TP); these must be selected as “Additional Archives”.

To convert an archive including models that have dependencies on other models not included in the archive (e.g. the boundary grid), the other models must be specified as “Additional Archives”.

For both selections “Additional Archives” as well as “Source Archives” the user may either select a single or multiple CIM archives in order to convert multiple archives at once or reference to multiple ones (e.g. the Boundary and the used EQ file) accordingly.

All available profiles in the selected archive will be shown on per MAS basis in the “Modelling Authority Sets” table. These MAS will be linked to existing grid elements if a matching rdfID is found, but can be adjusted manually if needed. Additionally the user can define how to deal with the models per MAS. Therefore, the options convert, link and ignore are available, where link means that the model is not converted, but only linked (e.g. an old version of a boundary is used). In case of conversion of an already existing (referenced) model, this will be updated in terms of new elements are added.

---

**Note:** A regular task in CGMES is to update the currently used boundary file. This can be achieved by referencing the old boundary grid in the CIM to Grid conversion and converting it. This way a merged boundary will be created (nodes that are used in the original Boundary that are not existing anymore in the new one will remain).

---

### 24.10.4 Grid to CIM Conversion

The Grid to CIM Conversion is accessible under *Tools* → *CGMES Tools* → *Grid to CIM Conversion*.

The following options for the destination are available:

- **New archive** converts the networks into a new archive with the given name.
- **Existing archive** imports the data into the archive specified in “Target Archive”; if the archive already contains models for the selected networks, the original models will be preserved.
- **Additional Archives** is used as a reference (e.g. original imports) to ensure the persistence of RDF IDs for CIM objects not represented in the *PowerFactory* model (e.g. cim:PowerTransformerEnd).

For “Additional Archives” the user may either select a single or multiple CIM archives for example to get dependencies on the correct Boundary grid as well as keeping IDs persistent.

A model for each profile selected will be created per network in the target archive. By default all profiles are selected. Boundary grids will only be exported with EQ and TP profiles (the respective boundary versions).

The option **Create difference models** creates a difference model, when selected. Note that difference models can only be created if *Existing archive* is selected as the destination. This archive should contain the base models for the difference models.

In order to create a bus-branch model, the option **Create bus-branch model** can be selected. When ticked, an internal reduction from a node-breaker model to a bus-branch model is done automatically. The resulting CIM archive will be a bus-branch model according to CGMES.

### Convert network selection

The *PowerFactory* networks to be converted can be selected by ticking the checkbox. Only activated networks can be converted. If one of the networks is to be treated as boundary grid, the corresponding checkbox must be ticked as well. Each network to be converted must have a Modelling Authority Set URI.

- Two or more networks can be associated to a common Modelling Authority Set. In such a case, all networks and data associated to a MAS will be exported into the same model set.
- If the selection contains two or more Modelling Authority Sets (apart from the Boundary network), and SV and/or DL profiles are selected for export, these instance data will be exported into an "Assembled" model set. Otherwise, if only one MAS is converted (apart from the Boundary network) SV and DL data will be exported into the same model set as EQ and TP models.

### Advanced Options

Further information for the model to be converted can be altered under the *Advanced Options* page:

- **Version** is an optional parameter to define the model version
- **Description** is an optional parameter to describe the model
- **Additional options** are inputs causing specific behaviour and do not need to be used.

## 24.10.5 CIM Data Validation

The CIM Data Validation is accessible under *Tools* → *CGMES Tools* → *CIM Data Validation* or *right-click* → *CIM Data Validation*.

This data validation is based on the UML profile information and can be used on CIM archives to validate their CGMES profile compliance. The archives used for this validation can be selected via "CIM Archives or Models". Therefore a multiselection is possible.

---

**Note:** This build-in validator can be used to validate archives of third parties or in case there are issues in the conversion (e.g. missing dependencies). The validator is not officially supported by the ENTSO-E.

---

## 24.10.6 Import and Export of the EIC as additional parameter

### Grid to CIM Conversion

The "IdentifiedObject.energyIdentCodeEic" attribute is not applicable to *PowerFactory* data-model. However, it is possible to assign an "EIC" to a *PowerFactory* element, by adding the following "User Attribute" entry at the end of the description field of *PowerFactory* elements:

```
<Attribute Name="EIC" Type="string">the code</>
```

For *PowerFactory* elements where no "EIC" is provided, no "IdentifiedObject.energyIdentCodeEic" is set in the corresponding CIM object.

## CIM to Grid Conversion

The “IdentifiedObject.energyIdentCodeEic” attribute is not applicable to *PowerFactory* data-model, thus is converted as a “User Attribute” entry at the end of the description field in *PowerFactory* elements as follows:

```
<Attribute Name=“EIC” Type=“string”>the code</>
```

If no “IdentifiedObject.energyIdentCodeEic” is set in a CIM object, the entry will not be created in the corresponding *PowerFactory* element. For CIM object classes which have no representation in *PowerFactory*, the “EIC” code is not converted, thus not visible in *PowerFactory* data-model (e.g. RegulatingControl, GeneratingUnit etc.).

## 24.11 MATLAB Interface

For a detailed description on the MATLAB interface refer to Chapters Stability and EMT Simulation and Modal Analysis, Sections [30.6.3: MATLAB Interface for DSL models](#) and [32.2.3: Output Options Modal Analysis](#).

## 24.12 OPC Interface

*PowerFactory*’s **OPC** interface is an asynchronous communication and data exchange mechanism used in process interaction and is widely applied in SCADA and control systems. This OPC implementation assumes that the *PowerFactory* software is executed as an OPC Client while the OPC Server is controlled via the external source. OPC server libraries are available from various manufacturers. An example of a freeware OPC Server is that available from Matrikon (“MatrikonOPC Simulation Server”).

*PowerFactory* supports both OPC DA (data access) and OPC UA (unified architecture) standards.

Figure [24.12.1](#) illustrates the integration of a SCADA system with *PowerFactory* via the OPC interface. In this OPC implementation, *PowerFactory* can be used either in GUI-less or normal mode. Some further characteristics of this integration include:

- OPC Client/Server exchange of any *PowerFactory* object parameter as well as any signal (bi-directional Data Exchange).
- *PowerFactory* listening mode to receive any data or signal from a registered OPC Server.
- *PowerFactory* sending mode to write back any data or signal to a registered OPC Server.

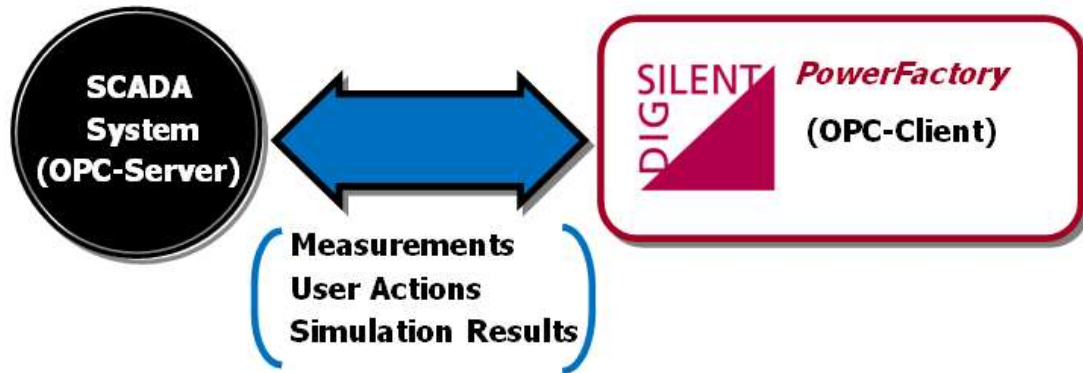


Figure 24.12.1: SCADA -*PowerFactory* integration via the OPC interface.

The OPC interface can be configured in two different modes:

- **Offline**
  - The bi-directional data exchange is carried out through an explicit command given by the user in *PowerFactory*. For example, by pressing a button predefined by the user in *PowerFactory*.
- **Online**
  - The bi-directional data exchange is automatically carried out at a certain frequency rate; where the frequency rate is determined by the user.

### 24.12.1 OPC Interface Typical Applications

Some typical applications of the OPC Interface are the following:

- **SCADA Online State Estimation**
- **SCADA Simulation Mode**, for example dispatcher load flow, switching validation.
- **SCADA Training Simulator**
- **Importing to *PowerFactory***
  - in order to update the operational data.
  - in order to reflect the Operator actions, such as breaker status and tap positions.
  - in order to perform state estimation based on the measured network data.
- **Exporting from *PowerFactory***
  - in order to update the SCADA interface with the calculated results.

## 24.13 StationWare Interface

This chapter describes the *StationWare* interface. An introduction into *StationWare* is provided in Section [24.13.1](#).

The following two sections describe the overall *StationWare* architecture (Section [24.13.2](#)) and the conceptual differences between *PowerFactory* and *StationWare* (Section [24.13.3](#)).

Both *PowerFactory* and *StationWare* have to be configured before they can be used together (Section [24.13.4](#)).



The *Getting Started* section (Section [24.13.5](#)) provides an introduction to the most important features. The complete documentation can be found in the section 'Description of the Menu and Dialogues' (Section [24.13.6](#)).

The terms *StationWare* and **PSMS** are used synonymously throughout this chapter. **PSMS** stands for Protection Settings Management System, and stresses the more internal and technical part of *StationWare*.

### 24.13.1 About StationWare

*DigSILENT StationWare* is a centralised asset management system for primary and secondary equipment. It provides a reliable central protection settings database and management system for the complete power system data, both to manage the various control parameters and to centrally store power system related information and data, based on the latest .NET technology.

*StationWare* stores and records all settings in a central database, allows modelling of all relevant work flow sequences, provides quick access to device manuals, interfaces with manufacturer-specific relay settings software, and integrates with *PowerFactory* software, allowing powerful and easy-to-use settings co-ordination studies.

Modern numerical relays have a large number of settings that are determined, stored and communicated by proprietary software solutions (these may be suitable for only one particular manufacturer or only one series or type of relay). This results in a fragmented and distributed settings "database". *DigSILENT StationWare* provides a single system that incorporates all different device protocols, thereby providing one manageable software data storage system, based on modern IT techniques, facilitating data interfacing and exchange in a transparent and straightforward manner.

*PowerFactory's* data exchange facility allows it to access the settings stored in *StationWare* such that these may be used as input to the powerful *PowerFactory* system simulation and protection settings tools. Settings that are calculated by using these tools may then be transferred back to *StationWare*.

### 24.13.2 Component Architecture

*DigSILENT StationWare* is a so-called *Client-Server Application*: the functionality is distributed over at least two computers: client and server. Figure [24.13.1](#) gives an overview of the components.

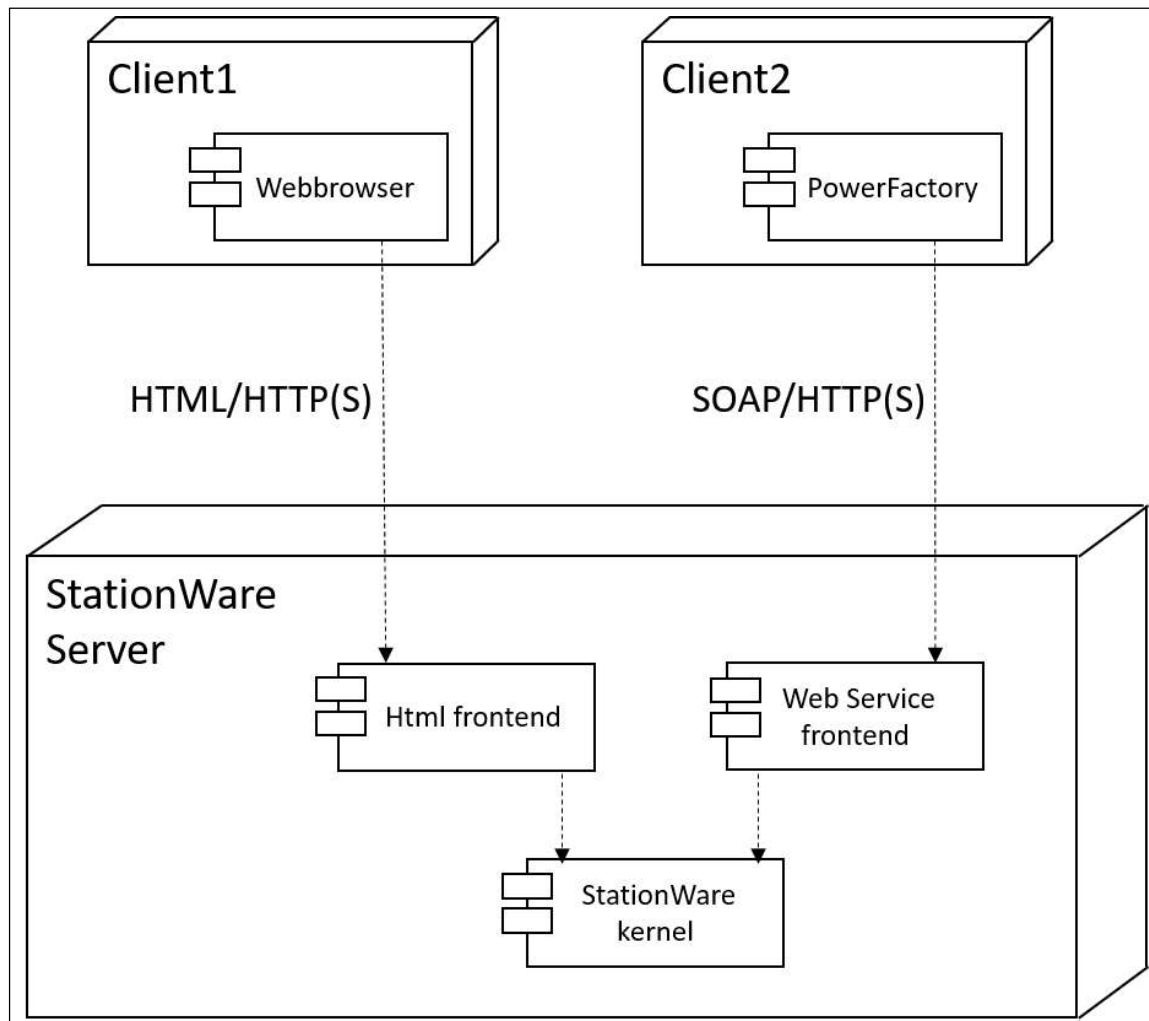


Figure 24.13.1: Architecture overview

There are usually several clients. One main advantage of this architecture is that the data is stored in one central database on the server. One client connects to the server and fetches the data from there, modifies it, and then stores it back to the server. These changes are visible on other clients.

*DlgSILENT StationWare* server provides two interfaces to access from client machines:

- Visualisation by means of a standard web browser. The HTML interface can be used with an usual web browser (e.g. Microsoft Internet Explorer or Mozilla Firefox). The browser displays HTML pages which are created by *StationWare's* HTML front end. The HTML pages are transferred using the HTTP(S) protocol on top of the TCP/IP internet protocol. HTML allows to present all kind of data e.g. plain text, tables or images. Additionally HTML provides concepts to achieve interactivity: by submitting HTML forms or pressing on hyperlinks data is sent to the server. The server interprets such requests and creates new HTML pages which are displayed by the browser again.
- The web service interface, similar to the HTML interface uses the HTTP(S) protocol to communicate with the web service frontend, though no HTML pages are transferred but lower-level data (SOAP/XML encoded). The web service client application is responsible to present this data conveniently. *PowerFactory* is able to play the role of a web service client. It integrates parts of *StationWare's* data and concepts smoothly into its own world.

**Note:** The default *StationWare* configuration requires SSL for the *StationWare* applications (web GUI and web services). Please use HTTP instead of HTTPS, if SSL is not enabled for your *StationWare* applications. In the following, the expression HTTP(S) is used.

The functionality of the HTML interface is covered in the *StationWare* manual. The remainder of this chapter focuses on *PowerFactory* as client.

### 24.13.3 Fundamental Concepts

Although *StationWare* and *PowerFactory* store data and settings associated with primary devices such as lines, transformers, ... and secondary devices, i.e. relays, CTs, VTs and circuit breakers, the two systems utilise different concepts to deal with this data.

In *StationWare* it is possible to model a location hierarchy and associate the devices to nodes in this hierarchy (e.g. substations). This has no equivalent in *PowerFactory*, where the devices are stored inside the parent grid (*ElmNet*) object.

Conversely, *PowerFactory* allows to the creation of a topological representation of networks which is not supported in *StationWare*.

This section describes the concept mismatch between *PowerFactory* and *StationWare*. In order to use the *StationWare* interface, it is important to understand the differences between both applications.

#### Location

In *StationWare* each device belongs to exactly one location. There are different location types e.g. *Region*, *Area*, *Site*, *Substation*, or *Bay*. The locations are organised in a hierarchy tree as shown in Figure 24.13.2.

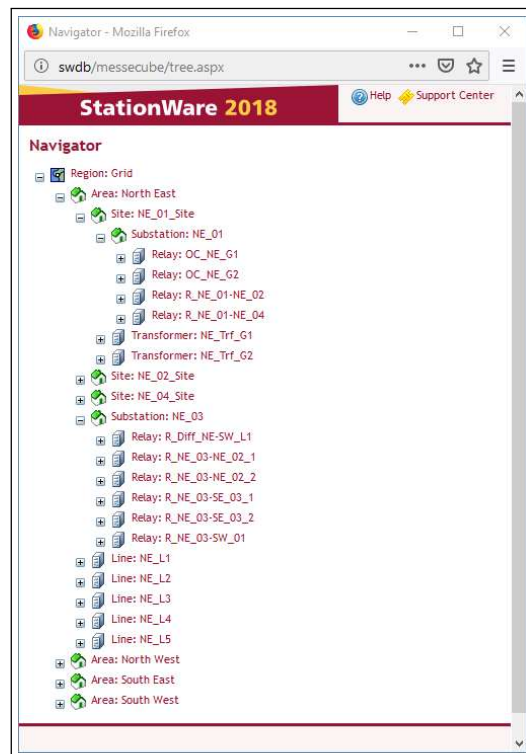


Figure 24.13.2: *StationWare* locations

In *PowerFactory* the data is organised in projects (*IntPrj*). A project may have one or more grids (*ElmNet*) which in turn contain net elements e.g. terminals, cubicles, and relays (*ElmRelay*). See Figure 24.13.3 for a typical *PowerFactory* project.

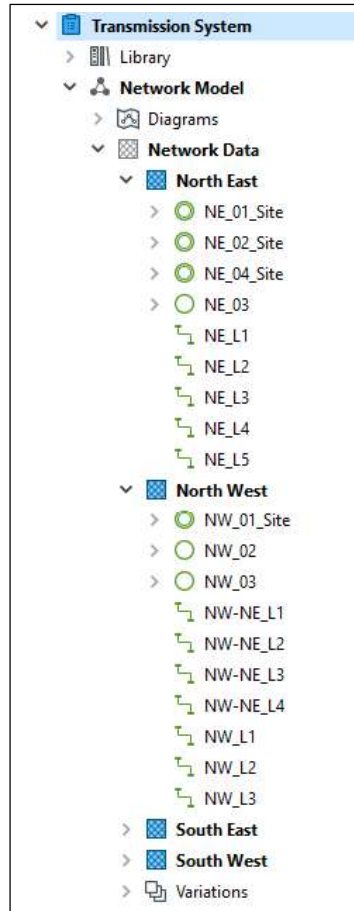


Figure 24.13.3: *PowerFactory* project

*StationWare*'s location concept and *PowerFactory*'s project/grid concept hardly fit together. That's the reason why the data mapping between *PowerFactory* and *StationWare* begins at the device level which is the subject of the next sections.

### Device

*StationWare* manages a set of devices e.g. relays, CTs, VTs, circuit-breakers, .... Each device is associated with a device type e.g. *ABB DPU2000R* or *SEL421 003*. In addition, each device has a unique ID: the *device ID*.

In *PowerFactory* a relay is represented by an *ElmRelay* object which references exactly one *TypRelay* object. The *ElmRelay* object contains several sub-components e.g. the I> component (a *RelToc* object), the Logic component (*RelLogic*), or the Ios component (*RelMeasure*). See Figure 24.13.4 for an example. The device ID is used to link one *StationWare* device to one *PowerFactory* device. The *PowerFactory* device e.g. an *ElmRelay* object stores the *StationWare* device ID as foreign key.

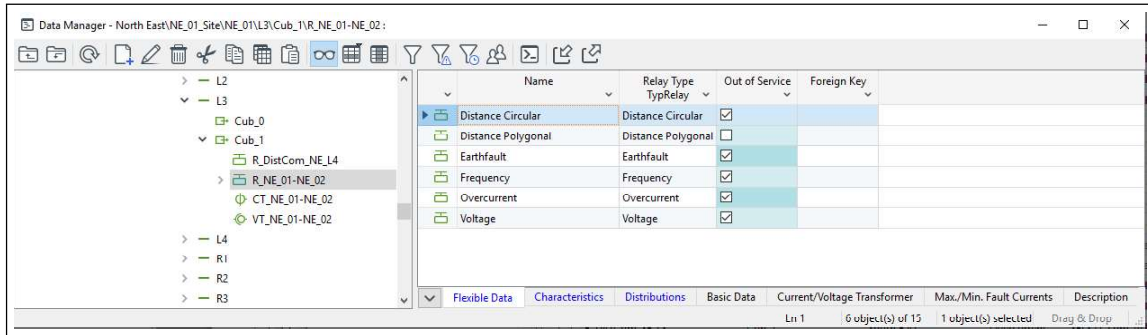


Figure 24.13.4: PowerFactory relay

**Device State**

A device’s state is in *StationWare* called setting. A setting is a list of parameters, and describes the state of one device completely. An parameter is a tuple of

- parameter *name*,
- parameter *type* which can be an arbitrary integer or floating point number, optionally with a range restriction, or a string, or a enumeration type.,
- a *default* value,
- an optional *unit*.

A complex relay may have thousands of parameters. In *StationWare* the setting parameters are organised in so-called setting groups. A setting group groups the parameters together which belong somehow together. It’s often defined by the device manufacturer. Each parameter belongs to exactly one setting group. Inside a group the parameter name is unique.

The device type defines which parameters and groups characterise a device. Table 24.13.1 shows an example of a possible device type. There are two setting groups G and H. Group G has the parameters a, b, and c, group H has the parameters d and e.

Group	Name	Type	Default	Unit
G	a	integer in [0,10]	0	A
	b	float	-0.32	l/s
	c	float in [0.03, 4.65]	1.0	
H	d	string	'DEFAULT'	
	e	enum 'yes', 'no', 'maybe'	'yes'	

Table 24.13.1: Settings Definition

According to this parameter definition a device can have settings as shown in tables 24.13.2 or 24.13.3.

Group, Name	Value
G,a	7
G,b	23.43
G,c	1.1
H,d	'abc'
H,e	'maybe'

Table 24.13.2: Settings Example 1

Group, Name	Value
G,a	8
G,b	0
G,c	1.1
H,d	'abcdef'
H,e	'yes'

Table 24.13.3: Settings Example 2

On the *PowerFactory* side there are neither settings nor groups. There is the *ElmRelay* object and its sub-objects. These objects can have parameters. See Table 24.13.4 for a definition and Table 24.13.5 for an example. The *TypRelay* type defines components and parameters.

*StationWare* parameters are mapped to *PowerFactory* parameters and vice versa. The mapping is non-trivial since only a small subset of the parameters (the calculation-relevant data) is modelled in *PowerFactory* and vice versa. Additionally there is no one-to-one relationship between the *StationWare* and *PowerFactory* parameters; i.e. a *PowerFactory* parameter may be calculated from several *StationWare* parameters.

Component	Parameter	Type
i>	o	integer
Logic	p	string
	q	enum 'enabled','disabled'
los	r	float
	s	float

Table 24.13.4: Parameter Definition

Some relays support *multiple setting groups* (MSG) also called *parameter sets*. Such relays have the same group many times (c.f. table 24.13.5). The groups H1, H2, and H3 have the same set of parameters (c and d). The relay models in *PowerFactory* do not support this concept. Instead of modelling all MSGs, only one instance of the H groups is provided.

In this case a group index parameter defines which of the MSGs actually is transferred from *StationWare* to *PowerFactory*.

### Lifecycle Phase

In *StationWare* each setting has one lifecycle phase e.g. *Planning* or *Applied*. At each point in time a device can have a set of settings e.g. three *Planning* settings, one *Applied* setting and 12 *Historic* settings.

Component Parameter	Value
i>:o	8
Logic:p	'HIGH'
Logic:q	'enabled'
los:r	18,5
los:s	19,5

Table 24.13.5: Parameter Example

Group	Name	Type	Default	Unit
G	a	integer in [0,10]	0	A
	b	float	-0.32	l/s
H1	c	string	'DEFAULT'	
	d	float in [0.03,1.65]	1.0	
H2	c	string	'DEFAULT'	
	d	float in [0.03,1.65]	1.0	
H3	c	string	'DEFAULT'	
	d	float in [0.03,1.65]	1.0	

Table 24.13.6: Multiple Setting Group Definition

In *PowerFactory* a device has exactly one state (or setting). Therefore when data is transferred between *PowerFactory* and *StationWare* always a concrete device setting in *StationWare* must be specified.

For *PowerFactory* purposes a special *PowerFactory* planning phase is introduced. The transfer directions are specified as follows:

- Imports from *StationWare* into *PowerFactory* are restricted to `Applied` and `PowerFactory` settings. `Applied` denotes the current applied setting (`Applied`) or a previous applied (`Historic`) setting.
- Exports from *PowerFactory* to *StationWare* are restricted to the `PowerFactory` setting. (`Applied` and `Historic` settings are read-only and can never be changed).

(Actually *PowerFactory*'s sophisticated variant management is similar to the phase concept, but there is no obvious way how to bring them together.)

## 24.13.4 Configuration

In order to transfer data between *PowerFactory* and *StationWare* both systems must be configured.

### StationWare Server

An arbitrary *StationWare* user account can be used for the *StationWare* interface in *PowerFactory*. The user must have enough access rights to perform operations e.g. for the export from *PowerFactory* to *StationWare* write-rights must be granted.

The bi-directional transfer of settings is restricted to lifecycle phases

1. of the phase type `PLANNING` or `REVIEW` and
2. with a cardinality constraint of 1 i.e. there may exist one or no such setting for one device.

Ensure that at least one phase fulfils these requirements, and there exists a setting of this phase.

### PowerFactory Client

The client operating system must allow connections to the server (network and firewall settings etc.).

Nothing has to be done in the *PowerFactory* configuration itself. The `TypRelays` in the Library must of course support *StationWare* - *PowerFactory* mapping.

## 24.13.5 Getting Started

The mapping between *PowerFactory* object attributes and calculation results with *StationWare* device settings or process attributes, or additional attributes of devices, is done via flexible DPL scripts. These

scripts have access not only to data in *PowerFactory* objects themselves, but also to other related objects e.g a relay type object or relay sub-blocks.

To be able to transfer data from *PowerFactory* to *StationWare* and vice versa, suitable DPL scripts have to be created and placed in an appropriate location in the project library folder.

```

Project.IntPrj
+- Library.IntPrjfolder
  +- Equipment Type Library.IntPrjfolder
    +- Relay Type X.IntFolder                                //TypRelay-specific scripts
      +- Relay Type X.TypRelay
        +- PsmExport.ComDpl
        +- PsmImport.ComDpl
      |
    +- Operational Library.IntPrjfolder
    +- Scripts.IntPrjfolder
    +- StationWare.IntPrjfolder                             //StationWare stuff
      +- Attributes.IntFolder                               //Folder for additional attributes
        |
        +- ElmLne.IntFolder
        | | +- PsmExport.ComDpl
        | | +- PsmImport.ComDpl
        +- ElmTr2.IntFolder
        | +- PsmExport.ComDpl
        | +- PsmImport.ComDpl
        |
      +- Results.IntFolder                                  //Folder for results export
        |
        +- arcflash.IntFolder
        | | +- ElmTerm.IntFolder
        | | | +- PsmExport.ComDpl
        | | +- ElmLne.IntFolder
        | | | +- PsmExport.ComDpl
        | |
        +- shc.IntFolder
        | +- ElmTerm.IntFolder
        | | +- PsmExport.ComDpl
        | +- ElmLne.IntFolder
        | | +- PsmExport.ComDpl
        |
      +- Settings.IntFolder                                //Device settings export/import
        |
        +- ElmLne.IntFolder
        | | +- PsmExport.ComDpl
        | | +- PsmImport.ComDpl
        +- ElmTr2.IntFolder
        | +- PsmExport.ComDpl
        | +- PsmImport.ComDpl
  
```

Figure 24.13.5: Structure of the project library folder

The scripts for importing/exporting device settings should be located in the sub-folder “Settings” of the *StationWare* folder inside the Project in *PowerFactory*.

Project\Library\StationWare\Settings.

For importing/exporting additional attributes from *StationWare* DPL scripts should be located in the sub-folder “Attributes”. To be able to export results from *PowerFactory* to *StationWare* the corresponding script should be located in the sub-folder “Results”. None of these folders are by default in project library folder and must therefore be created.

**Important:** DPL scripts for import/export relay settings must be saved in the same folder as the relay model (as contents of a *TypRelay* object). In difference to the data exchange of device settings, additional attributes and results, the DPL import/export scripts for relay settings can refer to mapping tables which simplify the mapping of individual parameters and the implementation of dependencies between parameters. Therefore, there are two different ways of exchanging relay settings. Either the mapping of the parameters in the DPL script code or the parameter mapping in mapping tables. Depending on which variant is selected, the basic DPL scripts differ. More information about the different mapping possibilities can be found in the documentation for [Protection Devices](#).



### 24.13.5.1 Import/Export of Relay Settings

This section is a simple walk-through and covers the most essential *StationWare* interface functionality.

By using a basic *PowerFactory* project and basic *StationWare* substation, it describes

1. how relays in *StationWare* and *PowerFactory* are created,
2. how these relays are linked,
3. how settings can be exported from *PowerFactory* to *StationWare*
4. how settings can be imported again into *PowerFactory*.

All (especially the more advanced) options and features are described in the section 'Description of the Menu and Dialogues' (see Section 24.13.6).

#### Prepare substation in StationWare

We begin with the *StationWare* side. We create a substation and two relays within:

- Start the web browser,
- log on to the *StationWare* system,
- create a new substation titled *Getting Started*,
- create two relays named *Getting Started Relay 1* and *Getting Started Relay 2* in the *Getting Started* substation.

In the HTML interface the station detail page should look as shown in Figure 24.13.6.

- Go to the detail page of the *Getting Started Relay 1* (Figure 24.13.7).

Since we have just created the device it has no settings, yet. Later it will contain a *PowerFactory* setting which reflects the relay state on the *PowerFactory* side.

The screenshot shows the StationWare 2018 web interface. The top navigation bar includes 'My StationWare', 'Hierarchy', 'Reports', 'Scripts', 'History', 'Library', and 'Administration'. The breadcrumb trail is 'Location 0 > Region: Grid > Area: North East > Substation: Getting Started'. The main content area is divided into two sections: 'Substation' and 'Actions'.

The 'Substation' section displays the following details:

ID	17701
Name	Getting Started
Description	
Foreign Key	
Created	11/11/2019 11:21:48 AM [Administrator]
Last Change	11/11/2019 11:21:48 AM [Administrator]
Additional Attributes	
SubAbb	
Maximo ID	
Inspection Period [months]	
Overall Status	Settings missing
Revision Number	

The 'Actions' section contains a grid of buttons for various operations:

Edit...	Create Device...	Create Process from Template...	[Report] Document Search...	[Report] Settings Views...
Copy...	Create Device from Template...	[Report] Additional Documents...	[Report] Lifecycle Changes...	[Report] Simple Location List...
Move...	Create Substation...	[Report] Advanced Search...	[Report] Overall Status...	[Report] Template Usage...
Detach...	Create Location from Template...	[Report] Audit Trail...	[Report] Settings by Additional Attributes...	[Report] Usage of Templates...
Delete...	Create Process...	[Report] Devices in Use...	[Report] All Settings of a Location...	

Below the actions is a navigation bar with tabs: 'Sublocations', 'Devices', 'All devices', 'All sub locations', 'Processes', 'Search', 'Additional Documents', 'Notes', 'Links', and 'Audit Trail'. The 'Devices' tab is active, showing a table of relays:

Name	Manufacturer	Usage	Type	Category	Description	Foreign Key
Getting Started Relay 1	Siemens		75A61_generic	Relay		
Getting Started Relay 2	Siemens		75A61_generic	Relay		

Figure 24.13.6: Substation

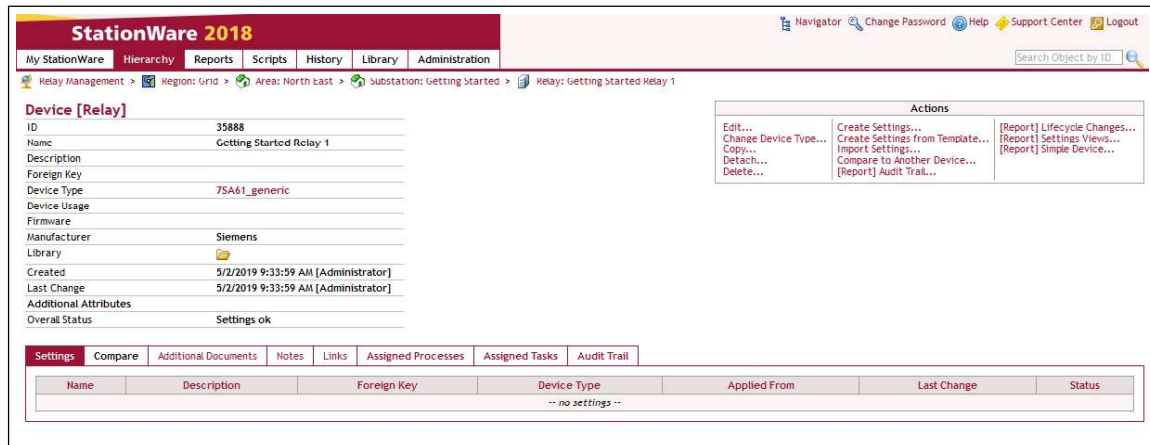


Figure 24.13.7: Device

### Prepare project in PowerFactory

Create a new *PowerFactory* project and create a simple grid within:

- Start *PowerFactory*,
- create a new project titled *GettingStarted*,
- draw a simple grid with two terminals (*ElmTerm*) connected by a line (*ElmLne*) as shown in Figure 24.13.8.

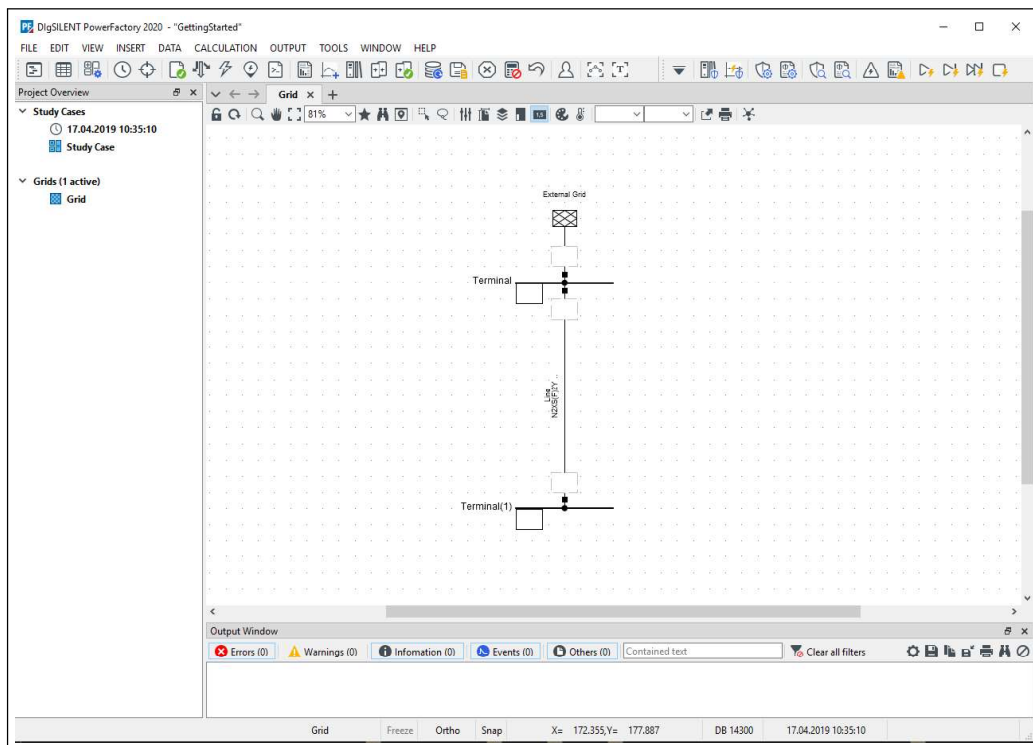


Figure 24.13.8: Grid

Now add a relay to the upper terminal:

- Right-click the cubicle quadrangle with the mouse. A context menu pops up.
- Select *New Devices.../Relay Model...* as shown in Figure 24.13.9.

A dialog pops up that allows you to specify the settings of the new relay (*ElmRelay*).

- Insert *Getting Started Relay 1* as Name,
- select an appropriate *Relay Type* which supports *StationWare* import/export (see Figure 24.13.10),
- press **OK**,
- in the same way add a relay *Getting Started Relay 2* to the second terminal.

*PowerFactory*'s object filter mechanism gives an overview over all devices inside the current project.

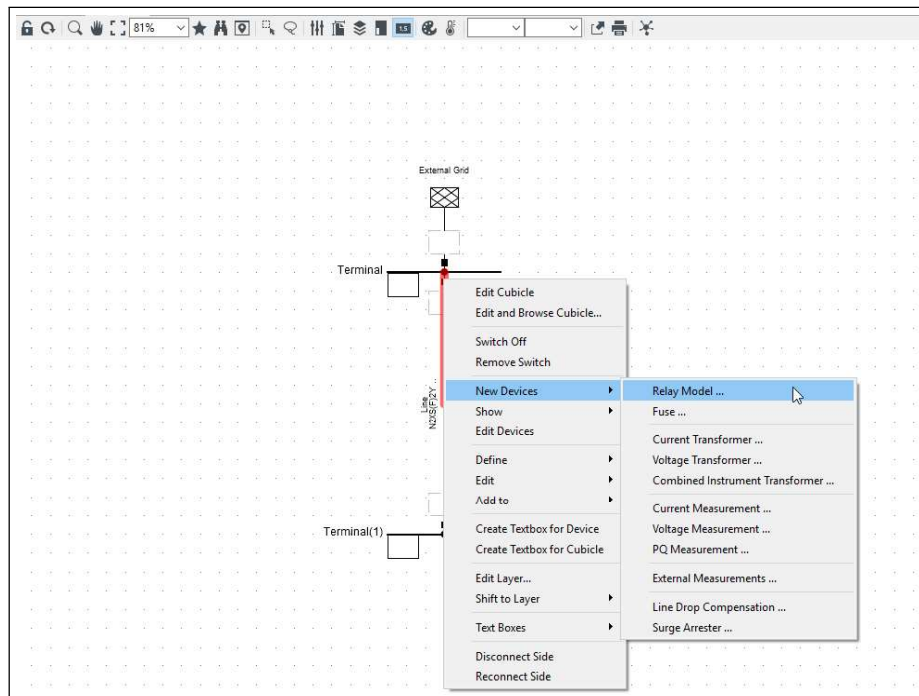



Figure 24.13.9: Cubicle context menu

- Press the icon  (*Open Network Model Manager...*) in the toolbar and select the class  (*ElmRelay*) to filter out all non-relay objects.

All calculation relevant relays (actually there only the two we created above) are displayed in a table (see Figure 24.13.11).

### Link Relays and establish a Connection

Now the *PowerFactory* relays must get linked to the *StationWare* relays. To be able to make a connection:

- Ensure that the DPL Import/Export scripts are saved in the same folder as the relay model. If mapping tables are used, ensure that the path and the name of the mapping tables is set in the DPL Import/Export scripts.
- Mark both relay  icons with the mouse.
- Press the right mouse button.

A context menu pops up as shown in Figure 24.13.12.

- Select the *StationWare* menu item,
- select the *Select Device ID* item.

A Log on to *StationWare* server dialog pops up. Since this is the first time *PowerFactory* connects to the *StationWare* server some connection settings must be entered.

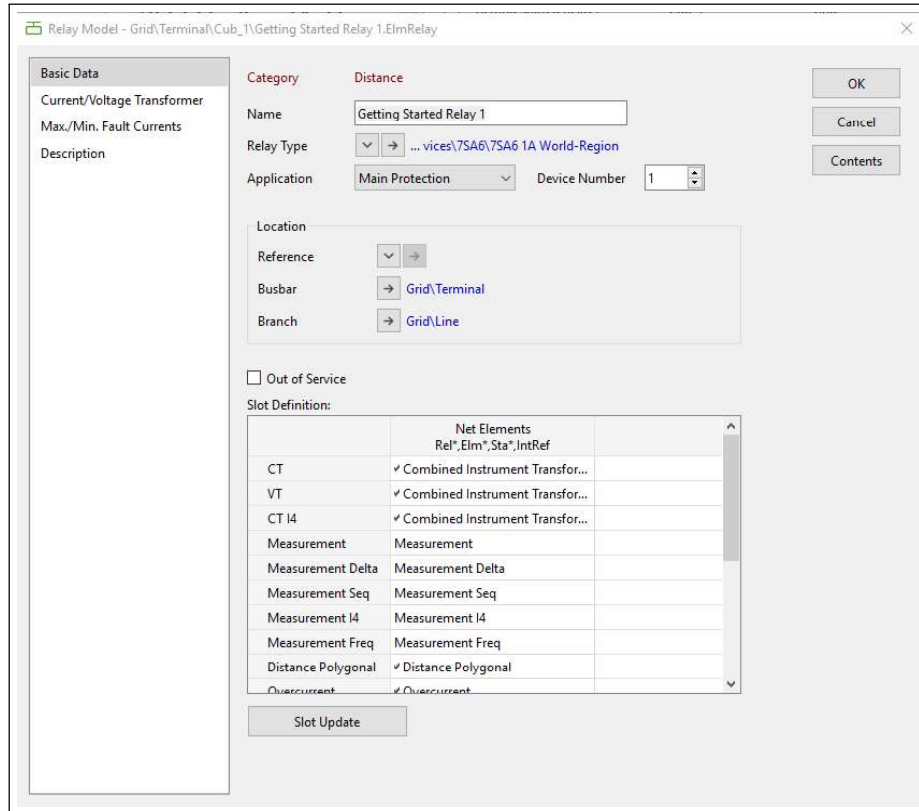


Figure 24.13.10: Relay dialog

- Enter the Server Endpoint URL of the *StationWare* server. The URL should have a format similar to `http(s)://192.168.1.53/psmsws/PSMSService.asmx`.
- Enter *Username* and *Password* of a valid *StationWare* user account.

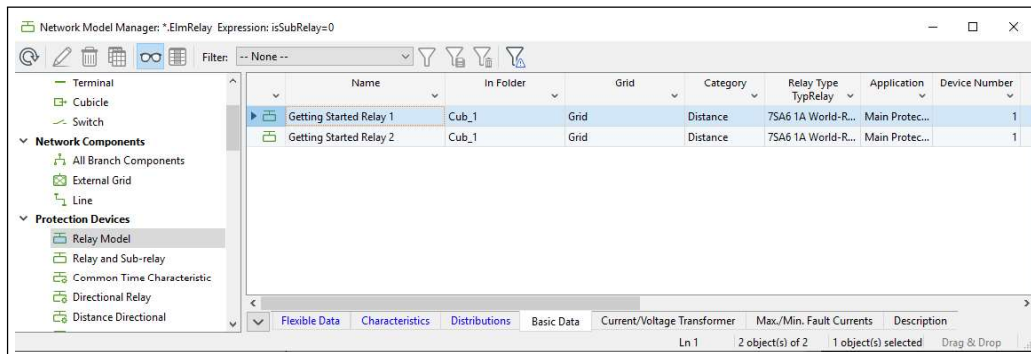


Figure 24.13.11: Relay display

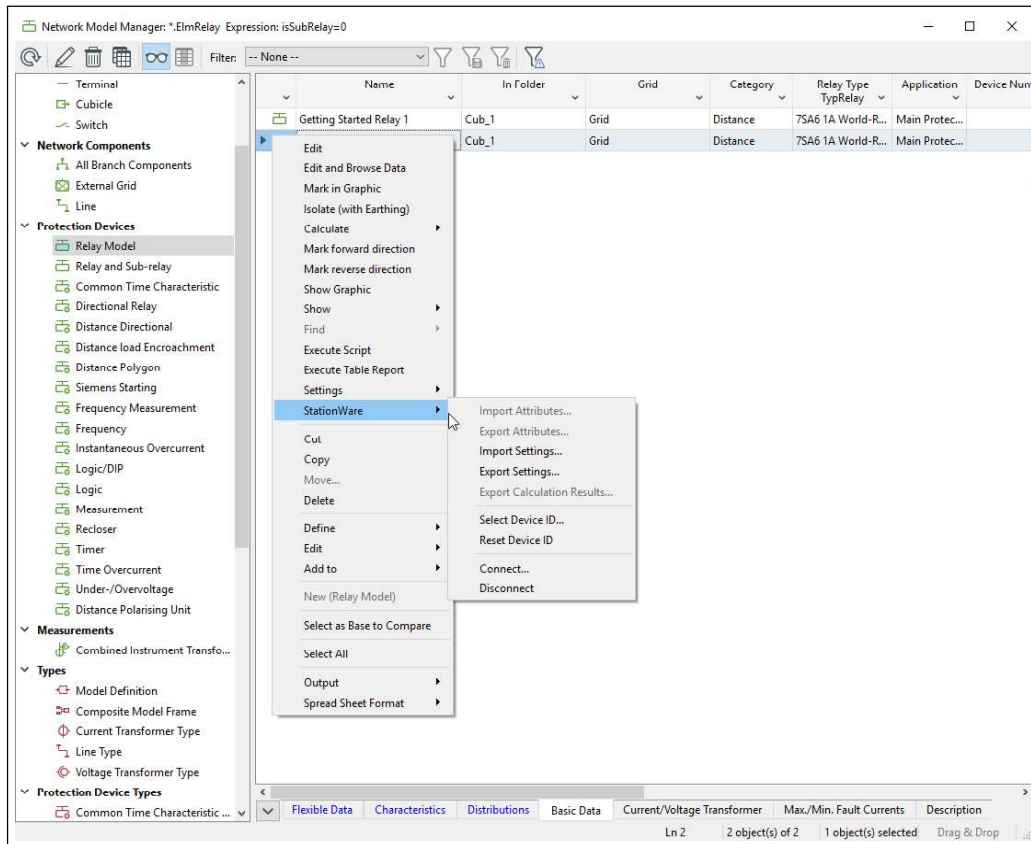


Figure 24.13.12: Device context menu

Figure 24.13.21 shows the dialog settings.

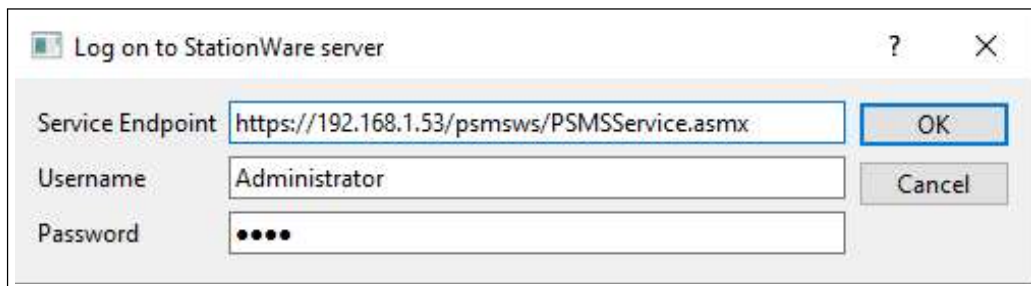


Figure 24.13.13: Log on dialog

- Press **OK**.

The connection procedure may take some seconds. If the server could be accessed and the user could be authenticated a success message is printed into the output window

```
Established connection to StationWare server
'https://192.168.1.53/psmsws/PSMSService.asmx' (version 18.2.6981) as user
'Administrator'
```

Otherwise an error dialog pops up. Correct the connection settings until the connection is successfully created. The section 'Description of the Menu and Dialogues' (Section 24.13.6) explains the connection options in detail.

Having established a connection to the server, a browser dialog pops up which displays the location hierarchy as known from the *StationWare* HTML interface. The dialog is shown in Figure 24.13.14.

- Navigate to the *Getting Started* substation,
- select the *Getting Started Relay 1* device,
- press **OK**.

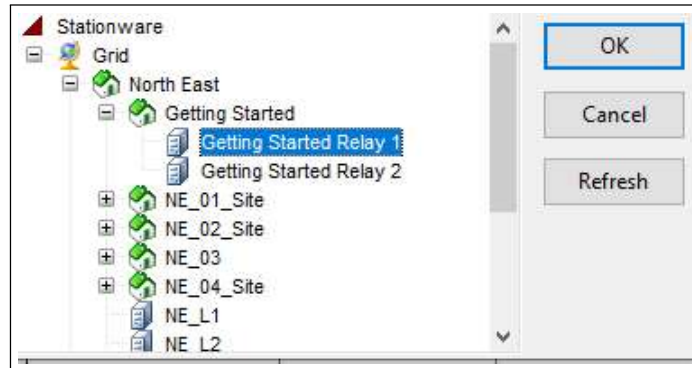


Figure 24.13.14: Browser dialog

Now the *PowerFactory* relay is “connected” to the *StationWare* device.

- In the same way select *Getting Started Relay 2* for the second *PowerFactory* relay.

### Export and Import Settings

Having linked *PowerFactory* to *StationWare* devices, the transfer between both systems can be started.

- Mark the relays with the mouse and right-click to get the relay context menu as shown in Figure 24.13.12.
- Select the *Export Settings...* in the *StationWare* menu entry.

A *ComStationware* dialog is shown which allows to specify the export options. See section ‘Export and Import Settings’ in the chapter 24.13.6 for all export options.

- Select *PowerFactory* as lifecycle phase,
- press **Execute**.

After a few seconds the relay settings are transferred to the server, and the output window contains the message

```
Exported 2 of 2 device settings successfully
```

The result can now be observed in the *StationWare* HTML interface.

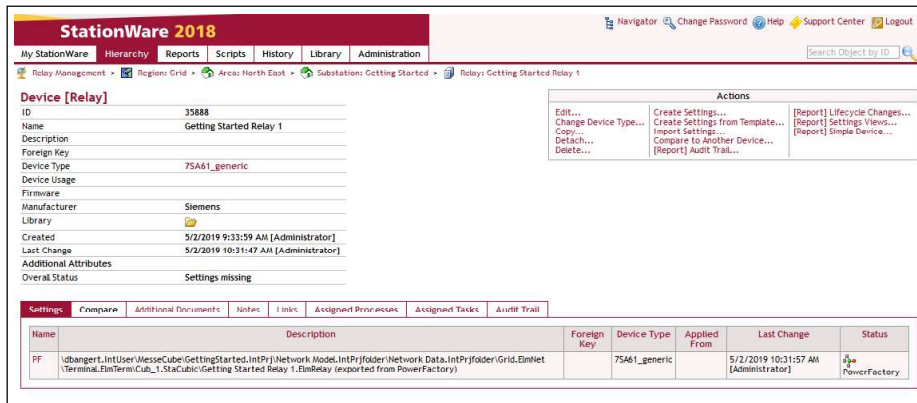


Figure 24.13.15: Device detail page

- Navigate to the relay detail view of the *Getting Started Relay 1* relay (see Fig. 24.13.15)

Observe the new created PF setting. The phase of this setting is *PowerFactory*.

- Switch to the settings detail page of the new *PowerFactory* setting (see Fig. 24.13.16).

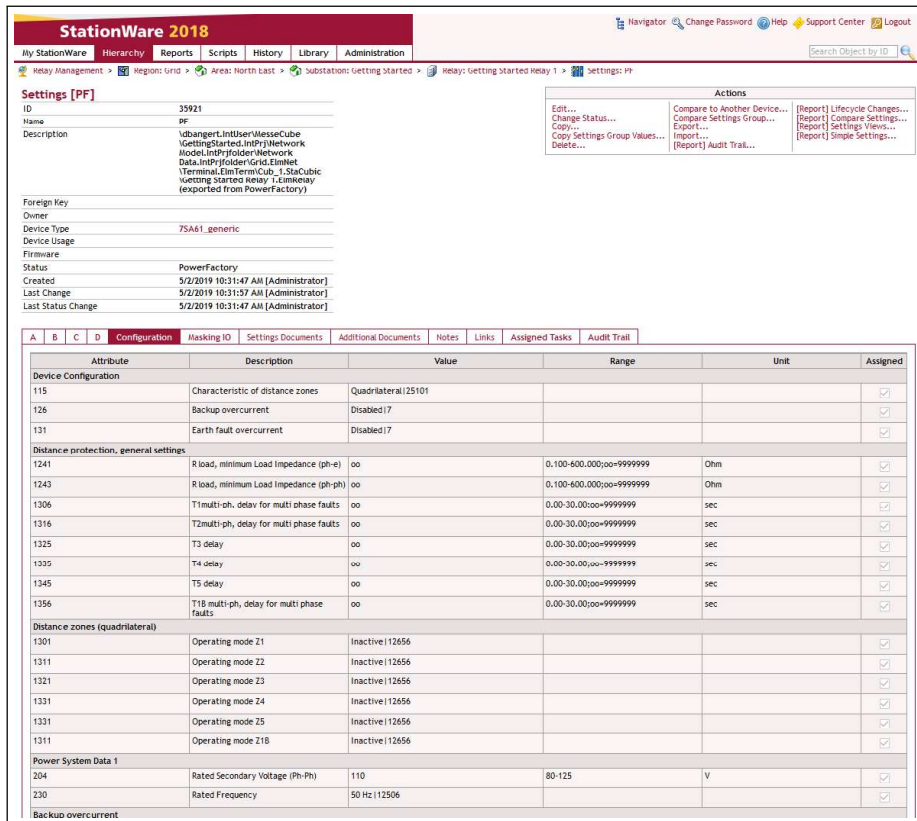


Figure 24.13.16: Setting detail page

The setting values should correspond to the relay state in *PowerFactory*. In the same way the *Getting Started Relay 2* relay has a new PF setting.

Now try the opposite direction and import a setting from *StationWare* into *PowerFactory*.

- Modify the PF settings in *StationWare* by entering some other values.

- In *PowerFactory* mark the relays with the mouse and right-click to get the relay context menu as shown in Figure 24.13.12.
- Select the *Import Settings...* in the *StationWare* menu entry.

Again the *ComStationware* dialog pops up as known from the export.

- Leave the default settings,
- press **Execute**.

Again the result of the settings transfer is reflected in the output window:

```
Imported 2 of 2 device settings successfully
```

- find *ElmRelay* object parameters changed according to the changes on the *StationWare* side

All import options are described in detail in the section 24.13.6: Import/Export Options.

### 24.13.5.2 Import/Export of the Additional Attributes

Additional attributes represent additional information which users may find useful for a location, device or settings within a device. These are not directly part of a settings record but are user-defined. For example, a common additional attribute that is useful for a feeder or substation location is the nominal voltage level in kV. Primary elements such as lines do not possess settings but instead parameters. Parameters such as length or impedance are then presented by the use of additional attributes.

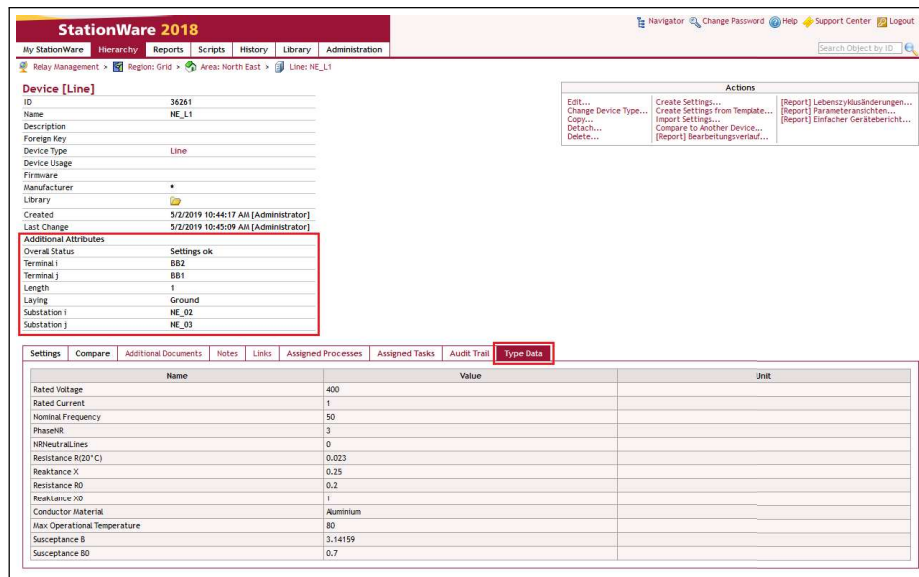


Figure 24.13.17: Additional attributes on the 'Device' page

The following information for additional attributes can be imported/exported:

- Name of the attribute
- Description
- Unit
- Value (Bool, String, Integer, Real, Enumeration, Data Time)
- Type (Attribute, Propagate, Overall Status, Revision Number)



Import/export of additional attributes also requires that the DPL script be saved in the appropriate place (see Section 24.13.5). All actions are similar to those described for settings (see Section 24.13.5.1) .

### 24.13.5.3 Export of the calculation results

Calculation results data exchange is only possible in one direction: from *PowerFactory* to *StationWare*. It is important to know that *PowerFactory* stores calculation results in attributes of temporary so-called “calculation objects”.

This data will be exchanged between “calculation objects” and *StationWare* process objects.

#### Preparation of *StationWare* for importing result data

Inside a *StationWare* project, define the process lifecycle, category and type. This process object should be configured to be capable of result data storage and presentation (e.g. “ArcFlashLabel Type” see 24.13.18).

**Important:** Process lifecycle must possess a phase named “*PowerFactory*” of type “Planning”.

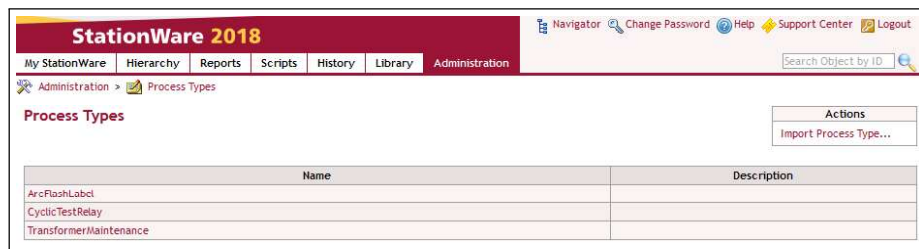


Figure 24.13.18: Process types page in *StationWare*

After being defined, the process should be created and have a device assigned to it.

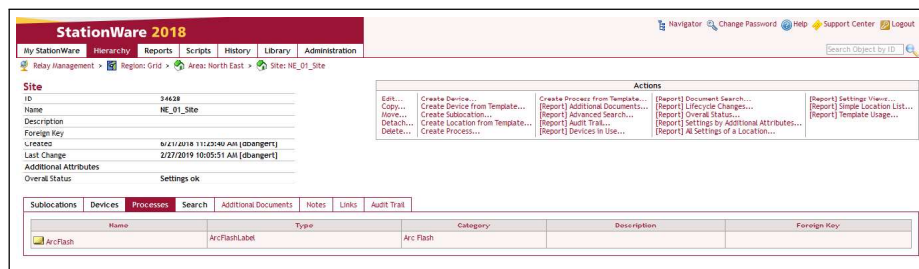


Figure 24.13.19: Location page where process is created

#### Preparing *PowerFactory* for export of result data

In *PowerFactory* it is important to have the DPL transfer script created and saved in a proper place inside the project library folder (see Section 24.13.5). It is necessary to use separate scripts for each calculation type and for each *PowerFactory* object class.

#### Connection of *PowerFactory* and *StationWare*

Refer to Section 24.13.5.1.

#### Export of results

Refer to similar section 24.13.5.1.

### 24.13.6 Description of the Menu and Dialogues

This section describes all options and features concerning the *StationWare* interface.

#### The Device Context Menu

Almost all functionality can be accessed by the device context menu. Mark one or more objects which supports the *StationWare* transfer e.g. *ElmRelay*

- in the object filter (Figure 24.13.12)
- in the Data Manager as shown in Figure 24.13.20.

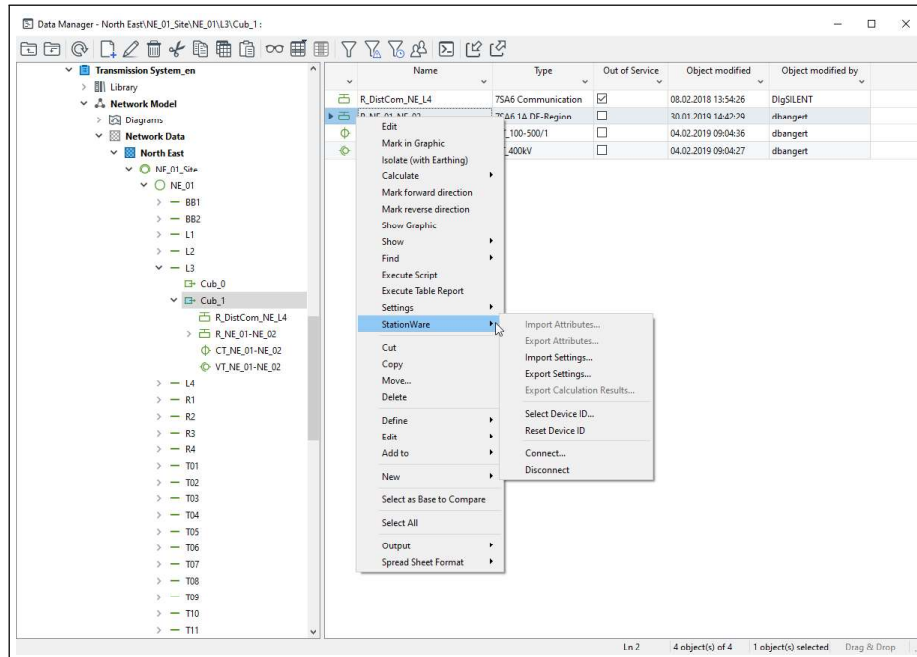


Figure 24.13.20: Device context menu

The *StationWare* submenu contains the entries as follows:

**Import X...** opens the *ComStationware* dialog and sets the device selection according to the above selected device objects. The *ComStationware* dialog settings are explained in detail in Section 24.13.6: The *ComStationware* Object.

**Export X...** does the same for the export direction.

**Select Device ID...** starts the Browser dialog (Figure 24.13.24) to link this device to a *StationWare* device. The dialog is subject of Section 24.13.6 : The Browser dialog.

**Reset Device ID** resets the device ID.

**Connect...** terminates the current *StationWare* session if it's already existing. Shows a Log On dialog. The connection settings are covered by Section 24.13.6. This may be useful when you are using several *StationWare* accounts and want to switch between them.

**Disconnect** terminates the *StationWare* session

#### Connection

Similar to the HTML interface the *StationWare* interface in *PowerFactory* is session - oriented: when

a user logs on to the system by specifying a valid *StationWare* account (username and password) a new session is created. Only inside such a session *StationWare* can be used. The account privileges restrict the application functionality e.g. an administrator account is more powerful than a usual user account.

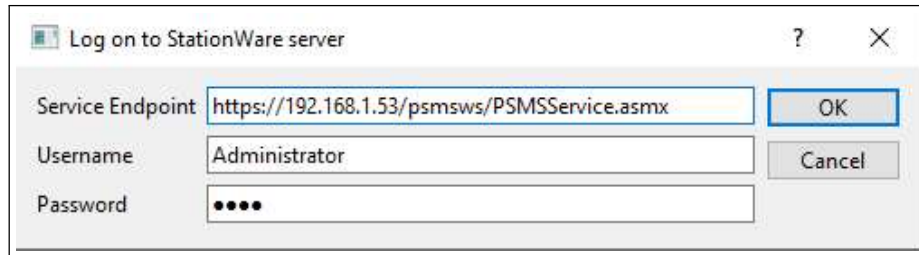


Figure 24.13.21: Log on dialog

Working with *PowerFactory* for the first time, the *StationWare* server is required, and the Logon dialog is as shown in Figure 24.13.21.

The *StationWare* connection options are stored in the user settings (Figure 24.13.22). After each successful logon the user settings are updated.

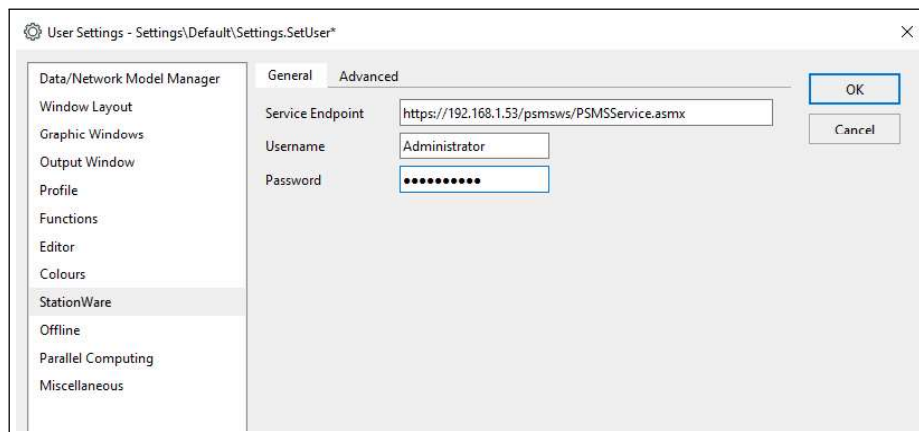


Figure 24.13.22: Log on dialog

As mentioned in the Architecture section (Section 24.13.2) *StationWare* is a client-server application. The *StationWare* server component is located on a server machine in the internet. The client component is the *PowerFactory* application which is running on a client machine.

The technology *PowerFactory* and *StationWare* use to communicate is called web services and is standardised like many other internet technologies (HTML, HTTP(S)). The server computer (or more exactly the *StationWare* service application on the server computer) has a 'name' by which it can be accessed. This 'name' is called service endpoint and resembles a web page URL:

```
https://the.server.name/psmsws/PSMSService.asmx
```

or

```
https://192.168.1.53/psmsws/PSMSService.asmx
```

http(s) denotes the protocol, the.server.name is the computer name (or DNS) of the server computer and psmsws/PSMSService.asmx is the name of the *StationWare* application.

The connection options are as follows:

**Service Endpoint** The Service Endpoint denotes the *StationWare* server 'name' as described above

**Username/Password** Username and Password have to be valid user account in *StationWare*. A *StationWare* user account has nothing to do with the *PowerFactory* user account.

The very same *StationWare* account can be used by two different *PowerFactory* users. The privileges of the *StationWare* account actually restrict the functionality. For device import the user requires read-access rights. For exporting additionally write-access rights are required.

### The Browser Dialog

As mentioned in the Concept description (see Section 24.13.3: Device) the *StationWare* device ID is stored as Foreign Key in the e.g. *ElmRelay* object dialog (Description page) as shown in Figure 24.13.23.

Figure 24.13.23: *ElmRelay* dialog

A more convenient way is to use the Browser dialog shown in Figure 24.13.24. The dialog allows to browse through the *StationWare* location hierarchy and select a device. The hierarchy data is cached to minimise network accesses. Due this caching it's possible that there may exist newly created locations or devices which are not displayed in the browser dialog. The **Refresh** button empties the cache and enforces *PowerFactory* to re-fetch the correct data from the server.

### The ComStationware Object

In *PowerFactory* almost everything is an object: relays are *ElmRelay* objects, users are *IntUser* objects, and grids are *ElmNet* objects, ...

What may be on the first sight confusing is the fact that actions are objects as well: for a short-circuit calculation a *ComShc* object is created. The calculation can be performed with several options e.g. 3-Phase, single phase, or 3 Phase to Neutral.

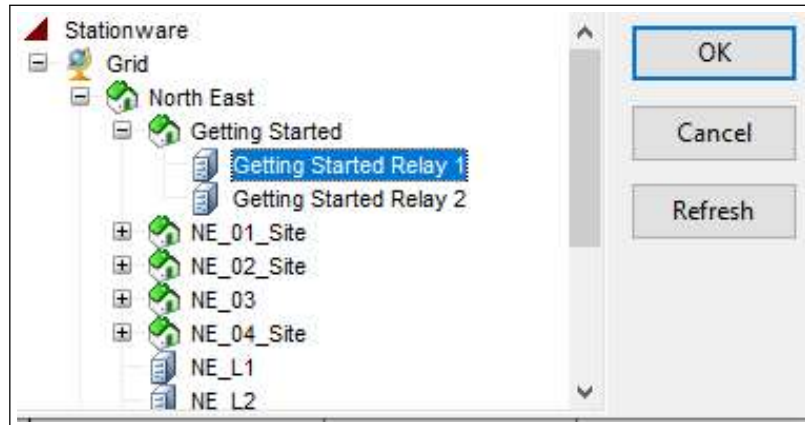


Figure 24.13.24: Browser dialog

You can even specify the fault location. All these calculation options are stored in the *ComShc* object. Every action object has an **Execute** button which starts the action. In fact there is a large number of parametrised actions like load flow calculation (*ComLdf*), simulation (*ComSim*), there is even a *ComExit* object that shuts down *PowerFactory*. All objects which can 'do' something have the Com prefix.

Since the *StationWare* interface is actually 'doing' something (it does import data, it does export data) it is implemented as a *ComStationware* object.

The *ComStationware* object is used both for the import and the export. It is located in the project's study case according to *PowerFactory* convention.

By default the study case of a new project contains no *ComStationWare* object. It is automatically created when it is first needed, as well as the *ComShc* object is instantiated at the time when the first short-circuit calculation is performed.

### Import/Export Options

The *ComStationware* dialog provides import/export options as follows:

**Transfer Mode** select Import/Export from *StationWare* as Transfer Mode

**Transfer Data** select Import/Export Data from *StationWare* (Attributes, Settings, Results of last calculation)

**Check only Plausibility** if the Check only Plausibility flag is enabled the import is only simulated but not really executed.

**Lifecycle Phase/Time stamp** A list of available lifecycle phases is shown.

- *PowerFactory* selects the current setting with *PowerFactory* phase as source setting.
- If Applied is selected the current Applied setting is transferred. If additionally a Timestamp value is entered the setting that was applied at this time is transferred which may either be Applied or Historic.

The Timestamp format is in ISO format: e.g. 2005-02-28 22:27:16

The time part may be omitted. Then 00:00:00 AM is assumed.

**All Devices** If All Devices is enabled, all calculation-relevant devices are imported/exported.

**Device Selection** Unless All Devices is enabled, the Device Selection provides a more subtle way to specify which devices are to be transferred.

The Device Selection is automatically set if the Device Context Menu mechanism (Section [24.13.6](#): The Device Context Menu) is used.

**All Settings Groups/Group Index** This parameter specifies how multiple settings groups (MSG) are handled.

The import/export transfer is started by pressing **Execute**.

## 24.14 API (Application Programming Interface)

For a detailed description on the API, a reference document is available via the main menu *Help* → *Additional Packages* → *Programming Interface (API)*