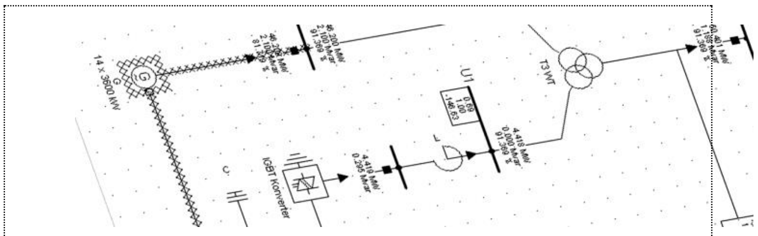


DIgSILENT Interface Documentation

DGS Interface

DIgSILENT GmbH
Heinrich-Hertz-Strasse 9
D-72810 Gomaringen
Tel.: +49 7072 9168 - 0
Fax: +49 7072 9168- 88
http://www.digsilent.de
e-mail: mail@digsilent.de

DGS-Interface

Published by
DIgSILENT GmbH, Germany

DGS-Interface, Doc. Vers. 007

PowerFactory 15

Date: 17.02.2014

Table of Contents

1 Introduction

DIgSILENT PowerFactory provides a standard interface named "DGS" (**DIgS**ILENT) for data exchange with other applications. The DGS *import* interface allows importing of complete network models as well as updating existing models. The DGS *export* interface provides the possibility to export network model data and calculation results. Selective export is supported.

This document describes the data model as well as the DGS data format. In Chapter 2 a short introduction to the PowerFactory data model is presented. Chapter 3 gives an introduction to the DGS data format, by describing the DGS data structure and the supported file formats. Chapters 4 and 5 present the export and import commands of PowerFactory. For experienced users, chapter 6 covers some advanced topics. And finally, the appendix contains a description of frequently used PowerFactory attributes and a sample header configuration. Furthermore, the used DGS examples are listed there.

2 PowerFactory Data Model

The PowerFactory data model distinguishes three different data categories:

• element data	Power System Elements
• type data	
• graphic data	Graphical Representation

2.1 Power System Elements

The first two categories contain the electrical attributes of the power system elements. The *type category* stores manufacturer data, the *element category* the operational and element specific attributes. Generally in the element data, references are given to the type data.

Type attributes of a cable are e.g. the resistance and the reactance per length in Ohm/km, whereas element attributes are for example, a reference to the cable type, the line length in km, the derating factor and the out of service status.

The graphic data category includes the graphic attributes of a certain power system element, e.g. position and size. This data category is not required for any calculation within PowerFactory.

The electrical network is in general represented by terminals and connecting branch elements like lines and transformers. The branch elements are connected to terminals via cubicles, where each connection requires its own cubicle. Connecting two terminals with a cable or line requires therefore two cubicles, see Figure 1.
Each cubicle is stored inside a terminal (as child) and points to the branch element.
The cubicles themselves may contain relays, current transformers (CTs) and voltage transformers (VTs).
The topological information (connectivity) is part of the element data category.

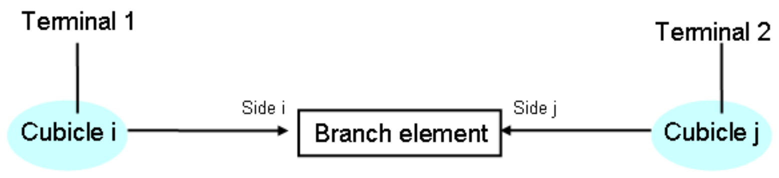


Figure 1: Schematic representation of network topology (Example of a branch element with 2 connections, e.g. line)

Figure 2 below illustrates a transformer which is connected to busbars (terminal of usage busbar) ***via*** disconnectors and circuit breakers modelled in detail.

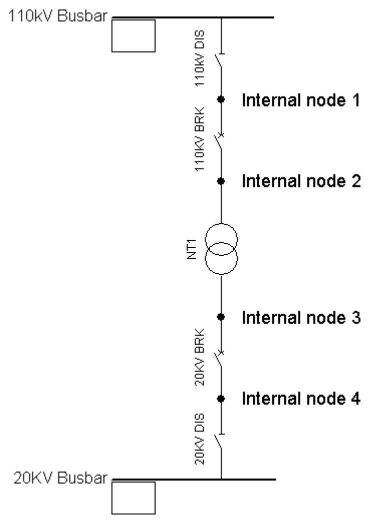


Figure 2 (Switch & Component Model):

consisting of:

- 2 busbars (terminals),
- 4 internal nodes (terminals),
- 4 switches (2 disconnectors, 2 circuit breakers) and one transformer.

The data model includes 10 cubicles, 2 for each switch and transformer. The cubicles are not visible in the figure.

Figure 2: Detailed Transformer Connection

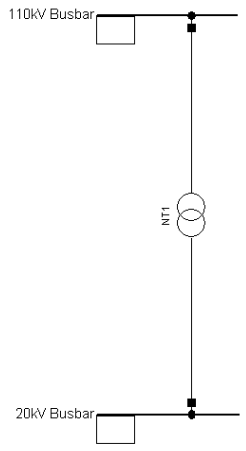


Figure 3 (Node & Branch Model):

The transformer NT1 is directly connected to the busbars via two cubicles. Each cubicle contains switch gears, which is shown as black rectangle. This simplified data model can be used if the switch details are not important.

Figure 3: Simplified Transformer Connection

Examples DGS files for both figures can be found in chapter 6.8 and 6.9.

2.2 Graphical Representation

The graphic is completely separated from the element data. It is organized in diagrams (represented by objects of class IntGrfnet) holding the graphical representations of power system elements (IntGrf). Such a graphic object (IntGrf) stores information about the symbol, position, size and has a reference to the element that is visualized (network element, e.g. line).

A graphical connection object (IntGrfcon) is used to display a connection line between two graphical symbols (branch symbol and node symbol). This object is stored inside the graphic object (IntGrf). The number of connection lines depend on the branch element, e.g. a load has only one connection line, a 2-winding transformer has two, a 3-winding transformer 3 connection lines.

Example: Graphical representation of a 2-winding transformer.

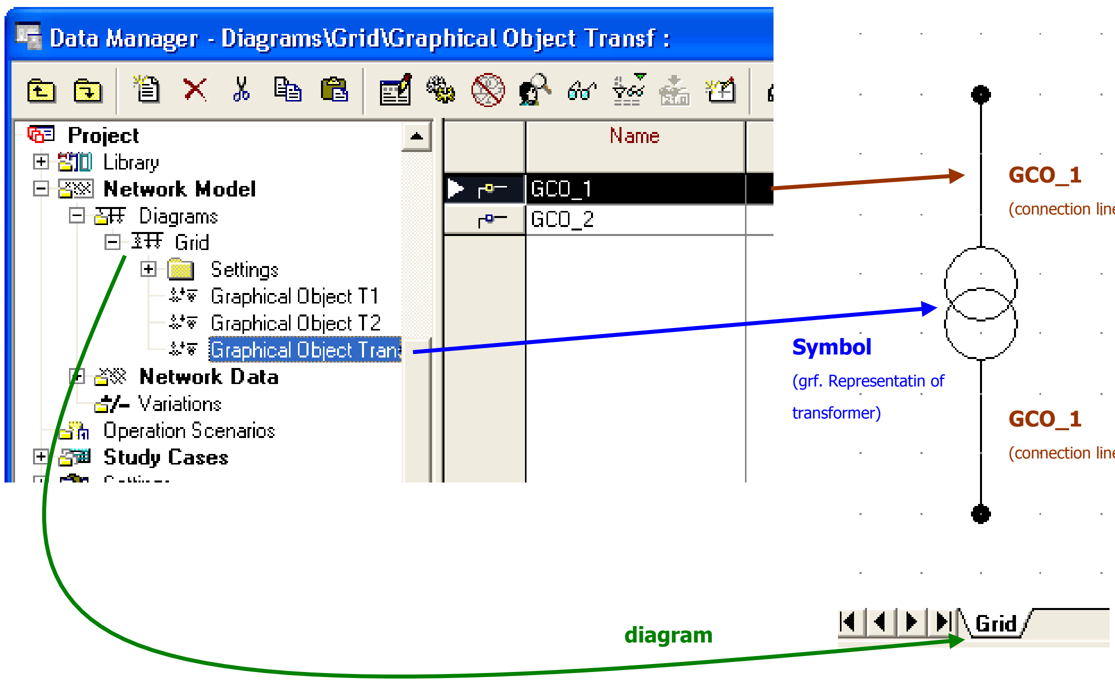


Figure 4: Graphical data objects and visual representation

(IntGrfnet green; IntGrf blue; IntGrfcon brown)

3 DGS Structure

Generally, DGS defines the structure of the data and is therefore not bound to a specific file format or database schema.

The PowerFactory DGS interface supports the following databases:

- Oracle (ODBC client 10 or newer)
- MS-SQL (ODBC driver 2000 or newer)
- ODBC System DSN

Additionally PowerFactory can deal with data in the following file formats:

- ASCII Text (csv like)
- XML
- Microsoft Excel (2003 or newer)
- Microsoft Access (2003 or newer)

The contents of the files are identical; the only difference is the format.

The core principle of DGS is to organize 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.

There are two types of tables:

- the general table and
- the object (data) table.

Each DGS file must contain exactly one "general" table and an arbitrary number of object data tables.

In EBNF notation:

```
    DGS file ::= general_table {, object_table}
```

The detailed structure of the tables is described in the following sections. Generally, for all tables, data values are stored in table rows. It is required that each table contains an identifier column. This column must be named "ID" (data type text) and all table rows must contain a unique value for that column (primary key in the *whole* DGS data source). In case of csv or Excel files the identifier column has to be the first column.

3.1 General Table

The general table is a special table named "General" providing meta information about the DGS file. This table is used as a kind of key-value map and consists of the following three columns:

Name	***Type***	***Description***
ID	**Text**	**Unique identifier for each data row**
Descr	**Text**	**Description text**
Val	**Text**	**Value**

For short:
```
    general_table ::= "ID", "Descr", "Val"
```

Currently, the following settings are supported (value for ID column can be freely chosen but must be unique).

Please note that the "Version" entry is mandatory and must be contained in every general table. Furthermore, all settings are case-sensitive!

Descr	***Description***
Version	DGS Version number. This entry describes the format of the DGS structure and is mandatory. The currently used format is DGS 5.0 Example: 5.0
Source	If given, the data source attribute ("dat_src", displayed on description page) for all newly created PowerFactory objects is set to that text. Please note that the text must not be longer than 3 characters! Example: GIS
PostCommand#	Advanced feature to execute commands after DGS import: It is possible to additionally execute command strings in PowerFactory at the end of the DGS import. These command strings must be given as entries named "PostCommand", followed by a number, e.g. PostCommand1, PostCommand2… After the DGS data import, these commands are executed in alphabetical order, i.e. first PostCommand1, then PostCommand2… Example: To close all graphic boards after DGS import, the following post command can be used (e.g. as PostCommand1): hide/all

Note: Unknown settings are ignored by PowerFactory (no warnings are displayed).

Example of a general table:

ID	Descr	Val
1	Version	5.0
2	Source	DGS
3	PostCommand1	hide/all

3.2 Object Table

The actual data of the PowerFactory objects is stored in object tables. Exactly one table is required for each used PowerFactory class. The name of the table must correspond to the class name (case-sensitive!), e.g. "ElmTerm" for terminals, "ElmTr2" for two-winding transformer.

The DGS interface is generic; tables can be added or deleted as required. The tables may occur in any sequence.

The columns represent the attributes of the PowerFactory class. One column must be the "ID" column. The remaining columns correspond to the PowerFactory attributes of the certain class. Each column must exactly be named as the attribute in the PowerFactory class. The table contains only the attributes required for the import or export task and is therefore generic with regards to the attributes.

Determined by the class name, the data of the PowerFactory objects are stored in the corresponding tables where each row holds the values for exactly one object in PowerFactory. There is a 1:1 correspondence of PowerFactory objects to table rows.

Example:
In PowerFactory, terminals are objects of class "ElmTerm". Amongst others, this class has the attributes:

loc_name:	name of the object, e.g. Busbar1
iUsage:	node type, busbar=0, junction=1, internal node=2
uknom:	nominal voltage in kV

A DGS file or database that stores these values must at least contain 4 columns (ID column + 3 attributes) and might look as follows (Table contains data five terminals):

ID	loc_name	iUsage	uknom
1	Busbar1	0	110
2	Busbar2	0	110
3	TerminalA	1	66
4	TerminalB	1	66
5	TerminalC	1	66

3.2.1 Object References

The handling of object references demands special attention. While plain data types as text and numeric values are directly stored in the table entries, special conventions are required for references to objects – either to objects existing in PowerFactory or to those stored in other rows within same DGS data file.

Before looking at an example, we need some pre-information at this stage. We already know what a cubicle is. In PowerFactory the table with cubicles is known as "StaCubic". The table with the general load is called "ElmLod" (see section 6.7.7)

DGS 5.0 supports the following ways of referencing an object:

a) <u>ID reference:</u> As the value in the ID column is primary key (unique within the whole file), it can be used to reference another object defined in the same DGS file. For attributes that point to other objects, the value stored in corresponding table entry in DGS is simply the DGS identifier (ID) of that object.

   Example: The following DGS excerpt shows a cubicle ("StaCubic") that is connected to a load (ElmLod). (The attribute "obj_id" of the cubicle points to the load object by using the DGS identifier.)

```
Table "ElmLod":         ID  loc_name     …
                        1   LoadA


Table "StaCubic":       ID  loc_name          obj_id  …
                        2   LoadCubicle       1
                        …
```

b) <u>Foreign-key:</u> Referencing objects that already exist in PowerFactory prior to the DGS import is realised using the value of the foreign-key attribute of that object (attribute "for_name" in PowerFactory). To indicate that a value represents a foreign-key reference, it must be preceded with double hashes ("##" without quotes).

   Please note, for import, only objects already existing in PowerFactory database can be referenced via foreign-key. It is not possible to create an object within a DGS file and refer to it via foreign key within the same file.

   Example: Same as a) but assume: that an import is executed into an existing project (update) and that the load already exists and has the foreign-key "Load117" (attribute "for_name" of load object)

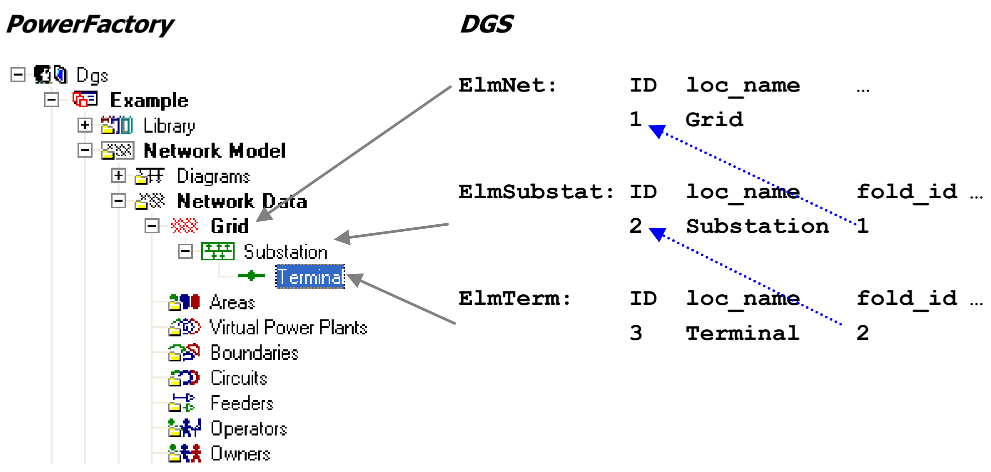
```
Table "ElmLoad":     --- (none, object already exists in PowerFactory)

Table "StaCubic":    ID  loc_name        obj_id
                     1   LoadCubicle     ##Load117
```

3.2.2 Object Hierarchy

In PowerFactory, all objects within a project are organized in a hierarchical structure. As this hierarchy is essential for the PowerFactory data model, the hierarchy must also be mapped to the DGS structure. This is done via the attribute "fold_id" (=parent) for all PowerFactory objects.

In DGS, for all objects that are not directly stored in a project folder (IntPrjfolder), the parent (attribute "fold_id") must be given. The parent can either be referenced using a DGS identifier or a foreign-key.

Example:



```
ElmNet:      ID  loc_name    …
             1   Grid

ElmSubstat:  ID  loc_name    fold_id …
             2   Substation  1

ElmTerm:     ID  loc_name    fold_id …
             3   Terminal    2
```

Figure 5: Hierarchy of power system elements in PowerFactory

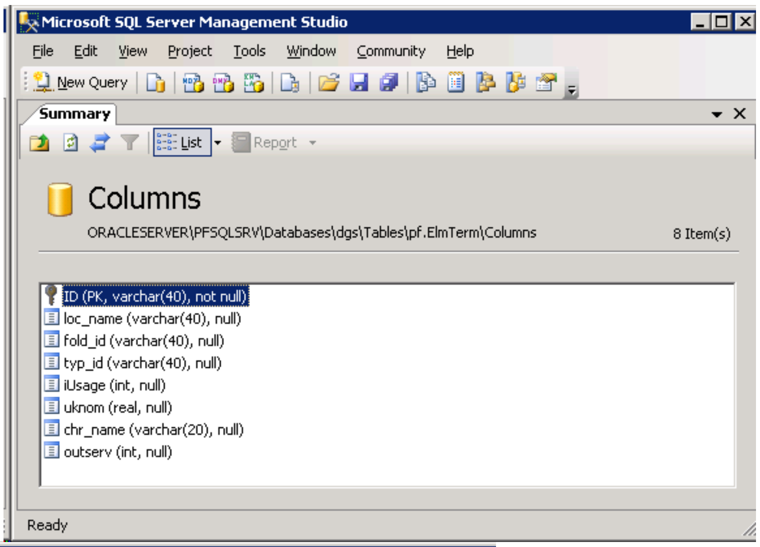
3.3 Databases

PowerFactory can directly access DGS data stored on a database server. Such a server can either be an Oracle or a Microsoft SQL server or any data source accessible via ODBC system DSN.

For all databases, the DGS tables are mapped 1:1 to database tables where all table and column names are case-sensitive. The data type for each column is chosen according to the PowerFactory attribute type. The following types are supported:

- INTEGER
- REAL (for float and double values)
- VARCHAR (for text, size is set according to the length of the PowerFactory attribute)

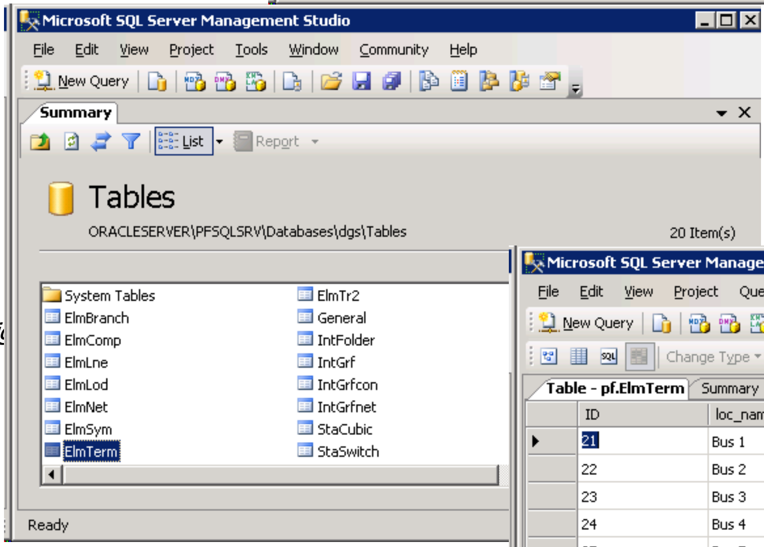
All ID columns are expected to be PRIMARY KEY.


Microsoft SQL Server Management Studio
File Edit View Project Tools Window Community Help
New Query
Summary
List
Report
Ready

Columns

ORACLESERVER\PFSQLSRV\Databases\dgs\Tables\pf.ElmTerm\Columns 8 Item(s)

ID (PK, varchar(40), not null)
loc_name (varchar(40), null)
fold_id (varchar(40), null)
typ_id (varchar(40), null)
iUsage (int, null)
uknom (real, null)
chr_name (varchar(20), null)
outserv (int, null)

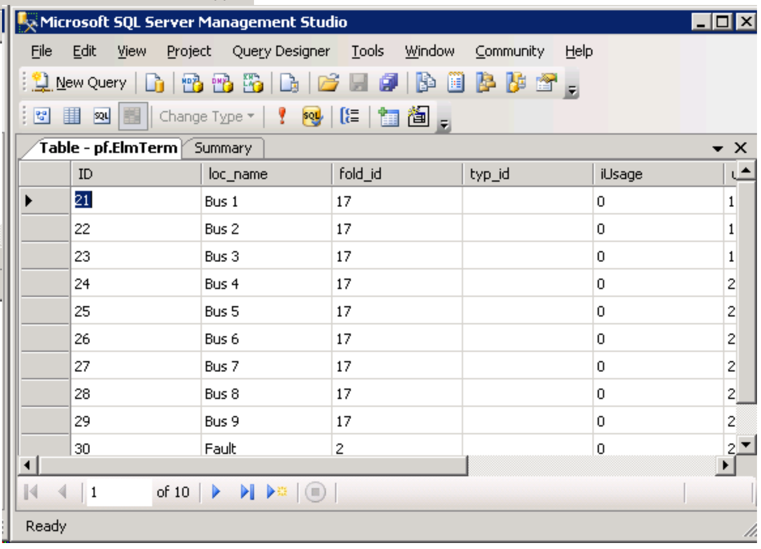

Microsoft SQL Server Management Studio
File Edit View Project Tools Window Community Help
New Query
Summary
List
Report
Ready

Tables

ORACLESERVER\PFSQLSRV\Databases\dgs\Tables 20 Item(s)

System Tables	ElmTr2
ElmBranch	General
ElmComp	IntFolder
ElmLne	IntGrf
ElmLod	IntGrfcon
ElmNet	IntGrfnet
ElmSym	StaCubic
ElmTerm	StaSwitch

Fi


Microsoft SQL Server Management Studio
File Edit View Project Query Designer Tools Window Community Help
New Query
Change Type
Table - pf.ElmTerm
Summary
1 of 10
Ready

ID	loc_name	fold_id	typ_id	iUsage	
21	Bus 1	17		0	1
22	Bus 2	17		0	1
23	Bus 3	17		0	1
24	Bus 4	17		0	2
25	Bus 5	17		0	2
26	Bus 6	17		0	2
27	Bus 7	17		0	2
28	Bus 8	17		0	2
29	Bus 9	17		0	2
30	Fault	2		0	2

3.4 File Formats

The DGS interface supports various file formats. These formats are described in the following chapters.

3.4.1 ASCII

A DGS ASCII file is basically a CSV format (comma separated values), extended by support for multiple tables (within one file).

Throughout the file,

- the semicolon ";" denotes the separator character,
- double quotes are used as escape character (") and
- all lines starting with a start ('*') are considered to be comments and are ignored in automatic processing.

To identify the different tables within the file, each table must begin with a special header line. This header must give the name of the table and the definitions for the columns. It must start with two dollar characters '$$' followed by the name of the table. After that, the column definitions must follow:

```
table_header ::= $$TableName {; ColumnDefinition}
```

Each column definition must consist of a column name (name of PowerFactory attribute), followed by its data type in parenthesis. For data types, the following identifiers are support:

- "i", integer
- "r", float
- "d", double
- "a", string, followed by the length, e.g. "a:15" for a string with a length of 15 characters
- "p", pointer, i.e. references to other objects/records (only DGS >= 5.0)

Example: A table header for terminals (class "ElmTerm") could have the following form:

```
$$ElmTerm;ID(a:40);loc_name(a:40);fold_id(p);typ_id(p);iUsage(i);uknom(r)
```

After the header line, the data rows are following. The number of given values must exactly match the number of columns defined in the header line of that table. If a text value contains a separator or starts/ends with a blank it must be escaped using the double quote (CSV conform encoding).

Example: see chapters 6.8 and 6.9.

3.4.2 XML

DGS data can also be stored in XML format. A DGS set consists of 2 parts: the DGS data and a XML schema definition. The definition is usually stored as a separated file referenced by the data file. The schema definition is obligatory as the DGS format itself is generic.

Basically, the schema definition represents the table headers. For each column, a separate XML element is defined. An entry in the DGS table is represented by a XML element containing DGS columns as child elements.

The data is stored as instances of those elements. Each DGS data column is stored as an element instance containing child instances for the attribute values.

Example: a DGS table for ElmTerm given as

ID	***loc name***	***iUsage***	***Uknom***
1	**Busbar1**	**0**	**110**
2	**Busbar2**	**0**	**110**

is described by the following schema (xsd)

```
<xsd:element maxOccurs="unbounded" minOccurs="0" name="ElmTerm">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="1" minOccurs="1" name="ID" type="xsd:string">
      </xsd:element>
      <xsd:element maxOccurs="1" minOccurs="1" name="loc_name">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="40"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element maxOccurs="1" minOccurs="1" name="iUsage" type="xsd:integer">
      </xsd:element>
      <xsd:element maxOccurs="1" minOccurs="1" name="uknom" type="xsd:float">
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

and its data would be represented as (xml)

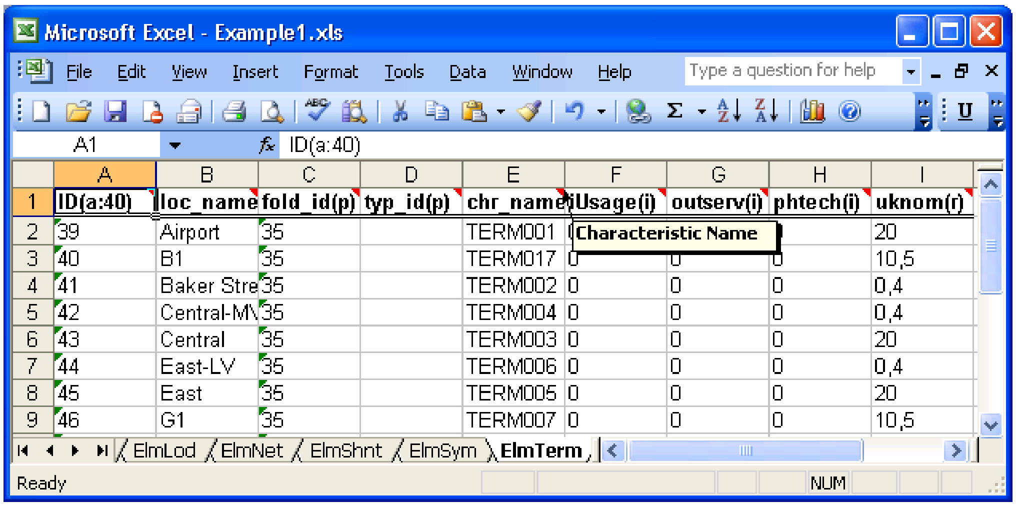
```
<ElmTerm>
  <ID>1</ID>
  <loc_name>Busbar1</loc_name>
  <iUsage>0</iUsage>
  <uknom>110</uknom>
</ElmTerm>

<ElmTerm>
  <ID>2</ID>
  <loc_name>Busbar2</loc_name>
  <iUsage>0</iUsage>
  <uknom>110</uknom>
</ElmTerm>
```

3.4.3 Microsoft Excel

Microsoft Excel is a sheet based file format. DGS makes optimal use of this fact by putting each table into its own sheet. Each sheet is named according to the table and contains just the data of that table. The sequence of the sheets is not of importance.

The sheets are filled with the table data. The first row (row '1') must contain the column definitions (header): The first column (column 'A') is always an ID column. Then, for all following columns, the name of the PowerFactory attribute and its type must be specified, e.g. "loc_name(a:40)". The type encoding is same as for ASCII files. An empty cell in the definition indicates the end of the row.

Each column definition row might be followed by one or more data rows. An empty row indicates the end of the table.


Microsoft Excel - Example1.xls
File Edit View Insert Format Tools Data Window Help
Type a question for help
A1
ID(a:40)
Characteristic Name
ElmLod ElmNet ElmShnt ElmSym ElmTerm
Ready
NUM

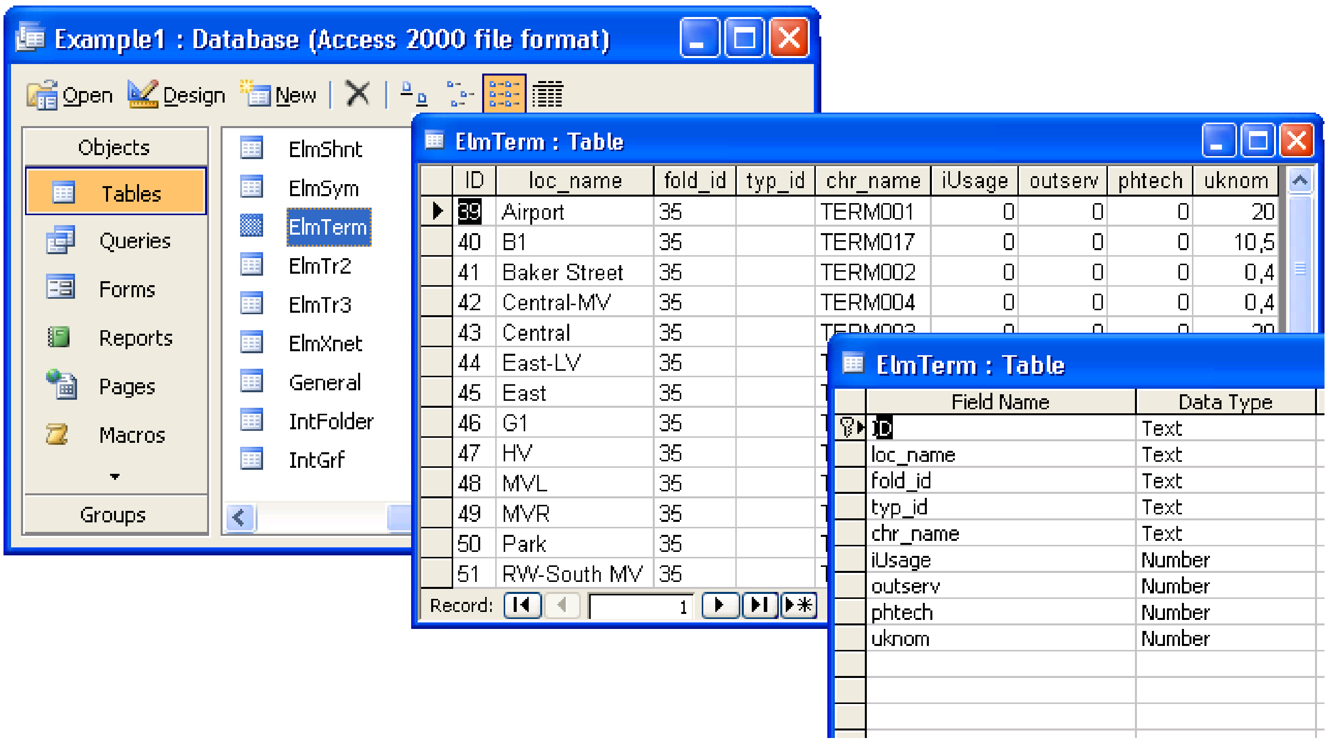
	A	B	C	D	E	F	G	H	I
1	ID(a:40)	loc_name	fold_id(p)	typ_id(p)	chr_name	iUsage(i)	outserv(i)	phtech(i)	uknom(r)
2	39	Airport	35		TERM001				20
3	40	B1	35		TERM017	0	0	0	10,5
4	41	Baker Stre	35		TERM002	0	0	0	0,4
5	42	Central-MV	35		TERM004	0	0	0	0,4
6	43	Central	35		TERM003	0	0	0	20
7	44	East-LV	35		TERM006	0	0	0	0,4
8	45	East	35		TERM005	0	0	0	20
9	46	G1	35		TERM007	0	0	0	10,5

Figure 7: DGS in Excel format

3.4.4 Microsoft Access

Microsoft Access is a file based database. The DGS tables are simply mapped to database tables one by one. This means, for each DGS table, an own database table is used. The name of the table must be identical to the name of the DGS table.

The table schema is determined by the column definitions of the DGS table. The data type is not incorporated into the column name but reflected by the data type of the table column.



ID	loc_name	fold_id	typ_id	chr_name	iUsage	outserv	phtech	uknom
39	Airport	35		TERM001	0	0	0	20
40	B1	35		TERM017	0	0	0	10,5
41	Baker Street	35		TERM002	0	0	0	0,4
42	Central-MV	35		TERM004	0	0	0	0,4
43	Central	35		TERM003	0	0	0	20
44	East-LV	35		T				
45	East	35						
46	G1	35		T				
47	HV	35		T				
48	MVL	35		T				
49	MVR	35		T				
50	Park	35		T				
51	RW-South MV	35		T				

Field Name	Data Type
ID	Text
loc_name	Text
fold_id	Text
typ_id	Text
chr_name	Text
iUsage	Number
outserv	Number
phtech	Number
uknom	Number

Figure 8: DGS in MS Access format

4 Import

DGS data can be imported into PowerFactory by using the built-in import command. The command object is of class "ComImport" and can be accessed via main menu: File → Import… → DGS Format

Generally, DGS data can be imported into

a) <u>a new project</u>:

> This means, a new project will be created and all DGS data will be imported into this project. There will be one object created for each DGS data row. In this case, the DGS data must be complete as all objects will be created from scratch. The data should include the topological information, type information, graphical information and the network element data. Further, it is not possible to use foreign-key references. Typically, this kind of import is used for complete data transfers from other systems, e.g. a Geographic Information System (GIS). Attributes not given in the DGS data keep their PowerFactory default value.

b) <u>an existing project</u>:

> In this case, the DGS data will be imported into an already existing project. This is different from the first option as the data can be very selective and must not be complete. Normally most objects are already existent and only some of them will get an update. The DGS can contain only a few attributes and foreign-key references are used to access the existing objects. A foreign-key value can either be used for an attribute pointing to an object and or for an update of an object itself. In the second case, the entry in the ID column must correspond to the foreign-key of the object (see example below).
>
> This type of import can be used to update operational data (switch states, active and reactive power consumption of loads, generator output, transformer tap changer positions…) e.g. from a SCADA system.
>
> Example 1: Changing the switch status of an existing breaker
>
> Assume that the breaker has a value "Breaker223" as foreign-key (attribute "for_name"). The status can then be set to open via the following DGS table:

```
Table: ElmCoup      ID              on_off
                    ##Breaker223    0
```

> Please note that not using a foreign-key as ID in the table above would result in a newly created breaker.

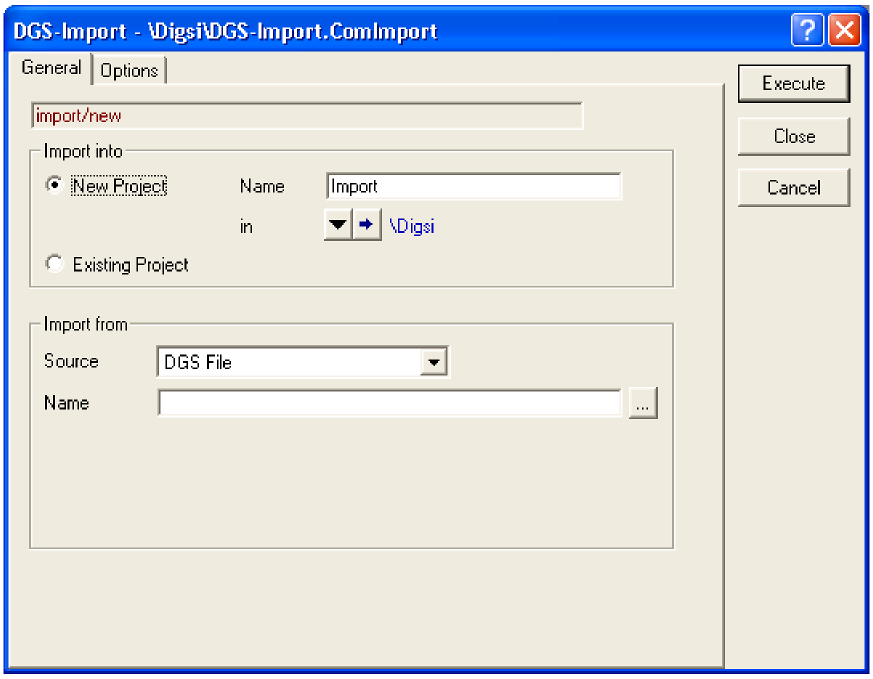
Example 2: Creating a new breaker and referring to an existing breaker type

The foreign-key mechanism is also used to reference existing objects, e.g. types.

```
Table: ElmCoup      ID  loc_name    typ_id
                    1   Breaker1    ##BrkType1
```

Generally in both cases, only given attributes will be changed. All attributes, not listed in the DGS table, will remain unchanged. (Newly created objects start with default values for all attributes.)

The way of import can be selected in the dialog of the import command (as depicted below):


DGS-Import - \Digsi\DGS-Import.ComImport

General | Options

import/new

Import into

[✓ New Project]   Name: Import

in: \Digsi

Existing Project

Import from

Source: DGS File

Name:

Figure 9: DGS import dialog

In this dialog, a DGS file has to be selected. The file format must either be set to ASCII, XML, MS Excel or MS Access. Depending on whether Microsoft Office is installed on the system, the file format MS Excel or MS Access might not be available.

5 Export

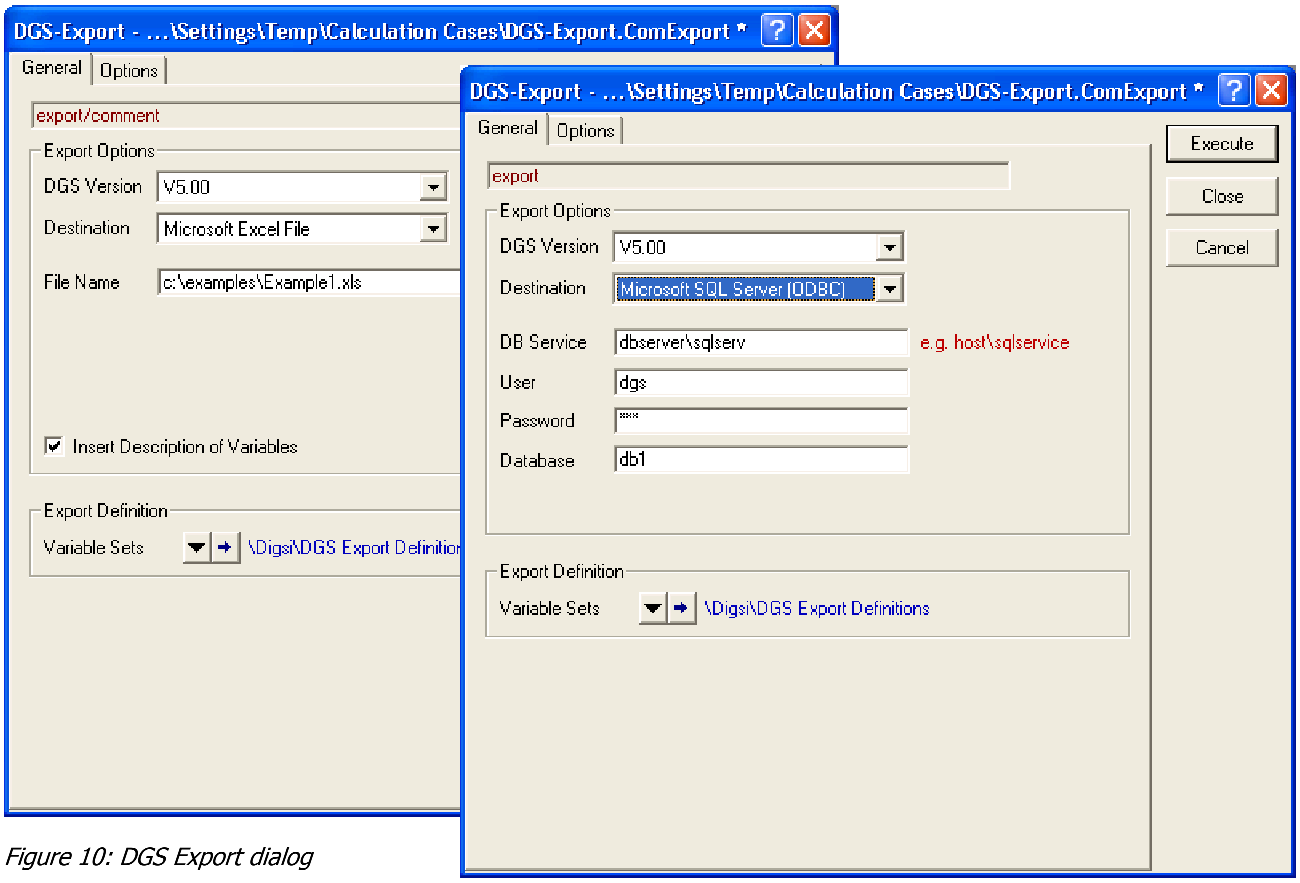
PowerFactory has a built-in export command to export an activated project to a DGS file. This means, the objects are exported in their current state (e.g. resulting state depending on active scenario and variations). Furthermore, the export can be fully configured to only export the attributes that are of interest. Inactive objects of variation management are generally ignored

The following data can be exported:

- element data,
- type data,
- graphical data and
- result data (e.g. load flow results)

The command object is of class "ComExport" and can be accessed via the menu: File → Export... → DGS Format

The configuration is done in the command edit dialog (as depicted below):



DGS-Export - ...\Settings\Temp\Calculation Cases\DGS-Export.ComExport *

export/comment

Export Options
DGS Version: V5.00
Destination: Microsoft Excel File
File Name: c:\examples\Example1.xls
☑ Insert Description of Variables

Export Definition
Variable Sets: \Digsi\DGS Export Definition

DGS-Export - ...\Settings\Temp\Calculation Cases\DGS-Export.ComExport *

export

Export Options
DGS Version: V5.00
Destination: Microsoft SQL Server (ODBC)
DB Service: dbserver\sqlserv e.g. host\sqlservice
User: dgs
Password: ***
Database: db1

Export Definition
Variable Sets: \Digsi\DGS Export Definitions

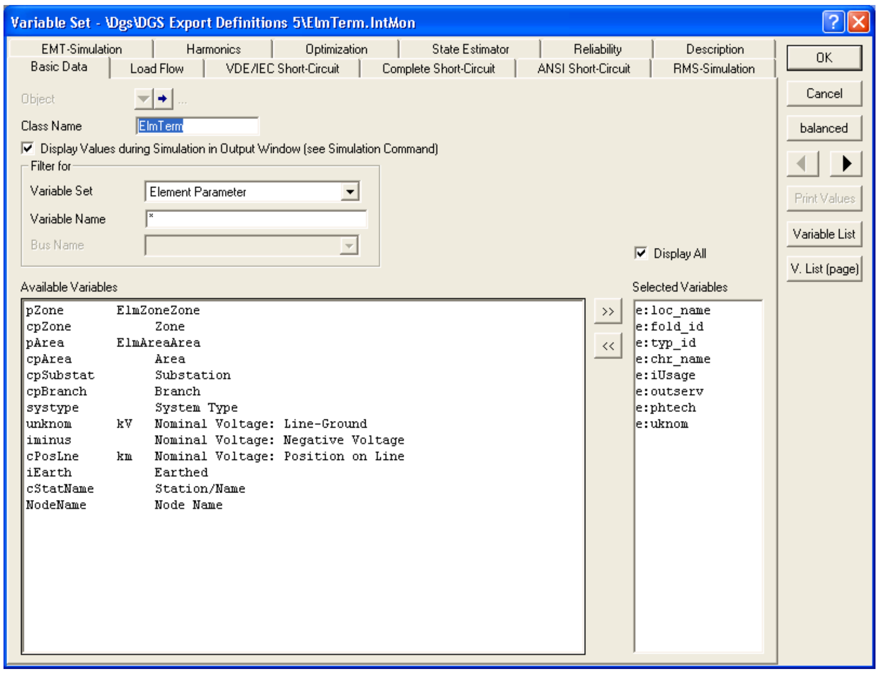
Figure 10: DGS Export dialog

The following options are available:

DGS Version	Version of the DGS structure. It is highly recommended to use 5.0 for PowerFactory V14.0.
Format	Output format. Either file as ASCII, XML, MS Excel or MS Access file (for Excel or Access, Microsoft Office must be installed on the computer) or database as Oracle, MS SQL Server and ODBC DSN.
Insert Description of Variables	If checked, in ASCII, XML and MS Excel a description of the columns is included (not available for Access or database server)
Variable Sets	Via this option, the attribute export is configured.

It is required to select a folder that contains exactly one monitor variable object (IntMon) for each class that is to be exported. The "Class Name" of each monitor variable object must be set to the PowerFactory class for which it provides a definition. The attribute definition for the class is represented by the selected variables in the monitor object.

Example for class ElmTerm:


Variable Set - \Dgs\DGS Export Definitions 5\ElmTerm.IntMon

Object

Class Name: ElmTerm

[✓] Display Values during Simulation in Output Window (see Simulation Command)

Filter for

Variable Set: Element Parameter

Variable Name: *

Bus Name

[✓] Display All

Available Variables

```
pZone       ElmZoneZone
cpZone           Zone
pArea       ElmAreaArea
cpArea           Area
cpSubstat        Substation
cpBranch         Branch
systype          System Type
unknom      kV   Nominal Voltage: Line-Ground
iminus           Nominal Voltage: Negative Voltage
cPosLne     km   Nominal Voltage: Position on Line
iEarth           Earthed
cStatName        Station/Name
NodeName         Node Name
```

Selected Variables

```
e:loc_name
e:fold_id
e:typ_id
e:chr_name
e:iUsage
e:outserv
e:phtech
e:uknom
```

Figure 11: Monitor object used for export attribute definition

Please note, the "Options" tab in the export command is currently not relevant for DGS 5.0.

6 Advanced Topics

6.1 Deletion of Existing Objects

DGS provides the possibility to delete already existing objects during import. Therefore, a special DGS object table is used where all objects that are to be removed are listed. The removal is executed as last step during import.

As only already existing objects can be deleted, *all objects in that table are referenced by foreign key*. The table is called "Deletion" and holds no data columns except the "ID" one.

A "Deletion" table might look as follows:

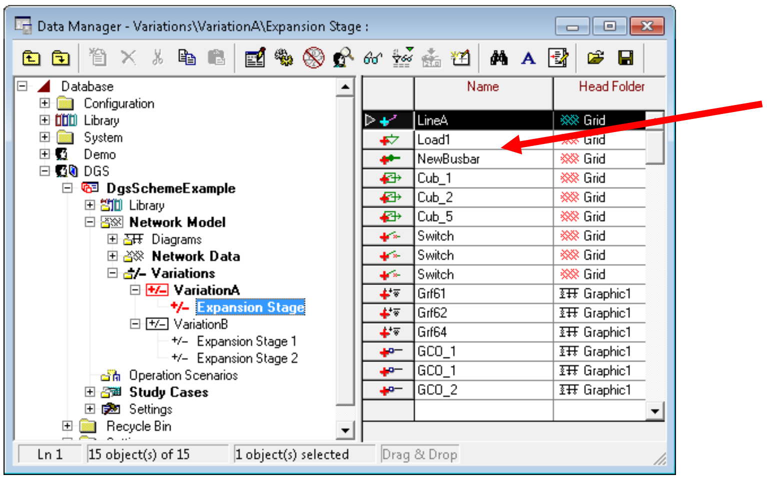
```
ID
##FKeyOfLoad1
##FKeyOfTerminalA
…
```

Please note, the deletion table is only supported for imports executed into an existing project.

6.2 Creation of Variations

Besides recording object creation and modification during import into a recording variation, experienced users can directly create variations in DGS.

In PowerFactory, variations are represented by IntScheme objects, having IntSstage objects as children. An IntSstage object becomes active at a certain study time and applies modifications to a network. The supported types of modifications are object *addition*, object *deletion* and attribute *modification*. The variation objects are directly stored inside its IntSstage object as depicted below. Please refer to chapter "7.3.4 Variations and Expansion Stages" of the "PowerFactory User's Manual" for a more detailed description.



Name	Head Folder
LineA	Grid
Load1	Grid
NewBusbar	Grid
Cub_1	Grid
Cub_2	Grid
Cub_5	Grid
Switch	Grid
Switch	Grid
Switch	Grid
Grf61	Graphic1
Grf62	Graphic1
Grf64	Graphic1
GCO_1	Graphic1
GCO_1	Graphic1
GCO_2	Graphic1

Ln 1 | 15 object(s) of 15 | 1 object(s) selected | Drag & Drop

Variation objects:
- add,
- modify and
- delete

Creation of IntScheme and IntSstage objects is quite straightforward. The IntScheme table consists basically of name and a parent column (fold_id). If the parent column is omitted, the object will be created in its default location, the "Variations" project folder.

Example of an IntScheme table:

ID	loc name	fold id
10	Variation A	{empty}
11	Variation B	{empty}

For contained IntSstage objects, it is important to properly set the parent IntScheme object and the activation time. The activation time must be given as number of seconds since 01.01.1970 UTC. Please note that an IntScheme object can contain multiple IntSstage objects.

Example of an IntSstage table:

ID	loc name	fold id	tAcTime
12	Expansion Stage	10	1286275114
13	Expansion Stage 1	11	1286275114
14	Expansion Stage 2	11	1286276328

The creation of modification objects is more complex and needs a detailed understanding of how PowerFactory deals with those objects: Each modification object (add, modify or delete) refers to a "root" object being the target of the modifications. On activation of an IntStage, the data of all contained add and change objects will be copied to their target objects, overwriting all attributes. For delete objects, the target object is deleted. These modifications of the target objects are done in memory only and are reverted on deactivation of the variation.

6.2.1 Delete Objects

Variation delete objects are modelled by a dedicated class called "IntSdel". For each object that shall be deleted, an IntSdel object has to be created in the expansion stage. A delete object is a simple reference object that points to the object that is to be deleted. An IntSdel table always consists of the columns "ID", "fold_id" and "root_id", where "fold_id" is a reference to the parent expansion stage and "root_id" references the target object that is to be deleted.

Example: Having a load (table ElmLod)

ID	loc name	fold id	...
100	Load A	{grid}	

A deletion of that load within an expansion stage would require the following record (table IntSdel):

ID	loc name	fold id	root id
200	Load A	{stage}	100

6.2.2 Add Objects

The adding of an object in a variation requires the existence of two objects in PowerFactory:

1. a hidden "root" object and

2. a variation add object inside an expansion stage.

The root object is a kind of placeholder located in the network. This root object is marked hidden as long as there is no add object active (for that object). All attributes of such a hidden object stay on their default values, except the attributes name ("loc_name"), parent ("fold_id") and foreign-key ("for_name").

The add object stored inside an expansion stage is responsible for making the hidden object active. It contains all the attribute data (except loc_name, fold_id and for_name). On activation of the expansion stage, all attribute data of the add object (except loc_name, fold_id and for_name) is copied to the hidden root object and the hidden object becomes active.

Hidden objects and variation add objects are always instances of the same class. For creation of hidden objects, a special table in DGS is used. The creation of add objects is done via normal class tables. For add objects, it is important to always set the reference to the hidden root object (root_id) and setting the variation status flag to a value of "4" (attribute iSchemeStatus).

The special hidden object table is named "HiddenObject" and consists of the following columns:

ID	DGS identifier as for any other object
loc_name	Name of the object. The name can only be set in the hidden root object and cannot be modified in a variation.
fold_id	Parent reference.
for_name	Foreign-key

<u>Example</u>: Adding a terminal in a variation could be done as follows:

The hidden object located for example in a grid (table "HiddenObject"):

ID	***loc_name***	***fold_id***	***for_name***
100	**Terminal1**	**{grid}**	**FKeyTerm1**

And a corresponding add object (IntSstage ID)

ID	***loc_name***	***fold_id***	***root_id***	***iSchemeStatus***	***...***
200	**{empty}**	**{stage}**	100	**4**	

6.2.3 Modification Objects

A modification object is a complete object instance residing in an expansion stage and pointing to a target object. It must be of same class as the target object and its variation status flag must be set to "2" indicating that it is a modification object.

On activation, all attribute values of the modification object are copied to the target object overwriting the existing data. Please note that attributes "loc_name", "fold_id" and "for_name" are never copied.

Example for modification of a terminal (table ElmTerm):

ID	*loc_name*	*fold_id*	*root_id*	*iSchemeStatus*	*iUsage*	*uknom*
100	Terminal1	{grid}	{empty}	{empty}	0	110
101	{empty}	{stage}	100	2	1	110

6.2.4 Notes

6.2.4.1 Attribute iSchemeStatus

The attribute iSchemeStatus is used for identifying add and modification objects in variations. For non-variation objects, the value "0" is used which can be omitted as it is the default value. The following values are supported:

0	Default; normal object
1	Hidden object, used for additions as inactive (hidden) placeholders. This value will automatically be set for objects in the "HiddenObject" table.
2	Modification object
4	Add object

6.2.4.2 Graphical Connections (IntGrfcon)

Special attention must be drawn to graphical connection objects that are added or modified in a variation. As the variations are inactive during creation, there are two attributes that cannot be automatically set and must explicitly be given in the DGS data: "iGrfNr" and "iDatConNr".

The "iGrfNr" is the graphical connection number specifying the side of the graphical symbol to which the connection line belongs to. This value is "1" based.

The "iDatConNr" called the internal connection number indicates the side of the edit object (cubicle) to which the connection belongs to. This value is "0" based.

Generally, proper values for these attributes can be derived from the corresponding cubicle (StaCubic) on element object topology:

- iGrfNr = StaCubic:obj_bus + 1
- iDatConNr = StaCubic:obj_bus

Please note that these attributes must only be set for graphical connection in variations (add and modification objects). Their value will be automatically detected for non-variation objects and should be left blank there.

6.3 Import sequence of attributes

For dependent attributes in PowerFactory objects, the sequence in which these attributes are set is important. For example, a load object provides several input possibilities for its operating point (active and reactive power, current and power factor...). Whether a special input option is available depends on the setting of the ‘input mode’ attribute. Therefore, it is important first to set the ‘input mode’ before setting values as P and Q, I and cos(phi)...

As from PowerFactory V14.0.525.4, V14.1.3 and V14.2, the order in which attributes are processed by the DGS converter can explicitly be defined. This is done by adding an integer sequence number to the attribute column (header column of DGS table), separated by the character ‘@’.

For example, a table for ‘ElmLod’ might look as follows:

ID@1	loc_name@2	fold_id@3	mode_inp@4	ilini@5	coslini@6
3	Load1	{grid}	IC	1.5	0.89

Remarks:

- Sequence numbers are completely optional. But, if used, they must be given for all columns of the table.
- For each table, it can individually be decided whether sequence numbers are given or not.
- Sequence numbers must be positive integer values of any range. They must be unique within a table definition. Gaps in the number range are allowed, e.g. @1, @5, @4 is valid although ‘2’ and ‘3’ are missing.
- As an alternative symbol the ‘#’ sign can be used to separate the sequence number. This might be required for DB servers that consider the ‘@’ sign as a reserved character. E.g. “loc_name@1” and “loc_name#1” are processed identically.

6.4 Multiple data sets

Working with multiple DGS data sets, e.g. different snapshots of operational data, is quite easy when using a file based format. In this case, each data set (snapshot) can be stored into a different file being identified via the file name.

Unfortunately, this approach is not applicable to databases as this would require different databases / table spaces for each data set. Therefore, the DGS database was extended to directly support multiple sets within one database. This is done by adding an extra data set ID column ("SID") to each DGS table. The data sets itself are declared in a separate "DataSet" table where additional description can be added.

Please note this feature is available for databases only! (Oracle, MS SQL, MS Access)

6.4.1 Database schema

This additional "DataSet" table consists of the following columns:

Label	(Text) Unique user identifier for the data set. This label is to be set in the DGS ex-/ and import commands. E.g. "MySet1", "MySet".. or "20120101", "20120102"...
Descr	(Text) Free description field. Currently not used by PowerFactory.
SID	(Integer) Internal data set identifier. Each "user" label corresponds to such an internal SID. This SID is used to identify the related records in the DGS tables.

Each DGS table, e.g. "ElmTerm", is extended by a SID column. This column is basically used as a filter for accessing data of a specific set only.

SID	(Integer) Internal data set identifier. Only values listed in the DataSet table are allowed.

Example for breaker positions (2 different data sets within same database):

Table *DataSet*

Label	***Descr***	***SID***
2013-01-01	Just some text	1
2013-01-02		2

Table *ElmCoup*

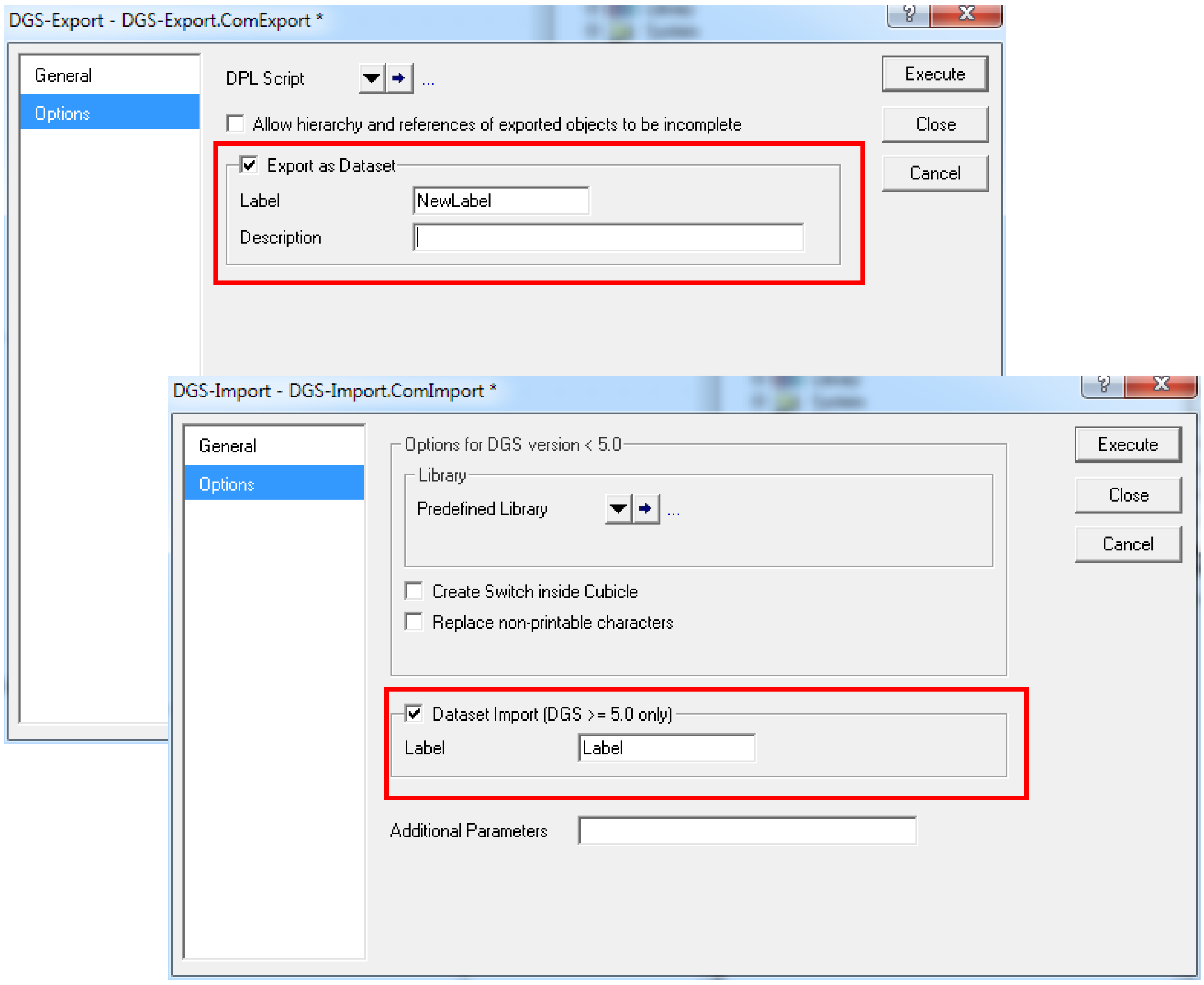
SID	***ID***	***loc name***	***on off***
1	100	##Switch1	1
1	101	##Switch2	1
1	102	##Switch3	0
2	100	##Switch1	0

```
2      101      ##Switch2             1
2      102      ##Switch3             1
```

Note: The contents of column "ID" needs only to be unique within records of same "SID".

6.4.2 User interface

For both export and import the data set is configured on the "Options" page. For export current data as a new set on a database, the "Export as Dataset" checkbox must be selected and a label for the new set must be given. The import is configured correspondingly.


DGS-Export - DGS-Export.ComExport *

General | Options

DPL Script

☐ Allow hierarchy and references of exported objects to be incomplete

☑ Export as Dataset

Label: NewLabel

Description:

Execute | Close | Cancel

DGS-Import - DGS-Import.ComImport *

General | Options

Options for DGS version < 5.0

Library

Predefined Library

☐ Create Switch inside Cubicle

☐ Replace non-printable characters

☑ Dataset Import (DGS >= 5.0 only)

Label: Label

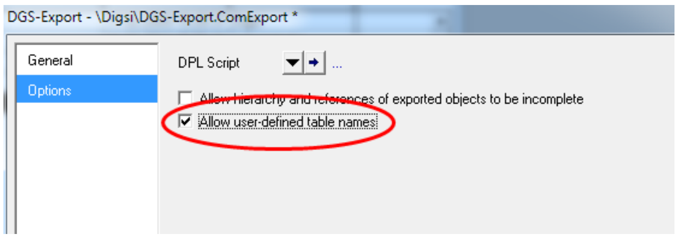
Additional Parameters:

Execute | Close | Cancel

6.5 User-defined table names (export only)

By default the DGS table names correspond to the PowerFactory class name. For special application cases it might be desired to use different, user-defined names. This is supported by the DGS converter and needs to be enabled as follows:

In the export command (ComExport) there is an option (checkbox on the "Options" page) that allows switching between free and standard table names. By default, this option is disabled.


DGS-Export - \Digsi\DGS-Export.ComExport *

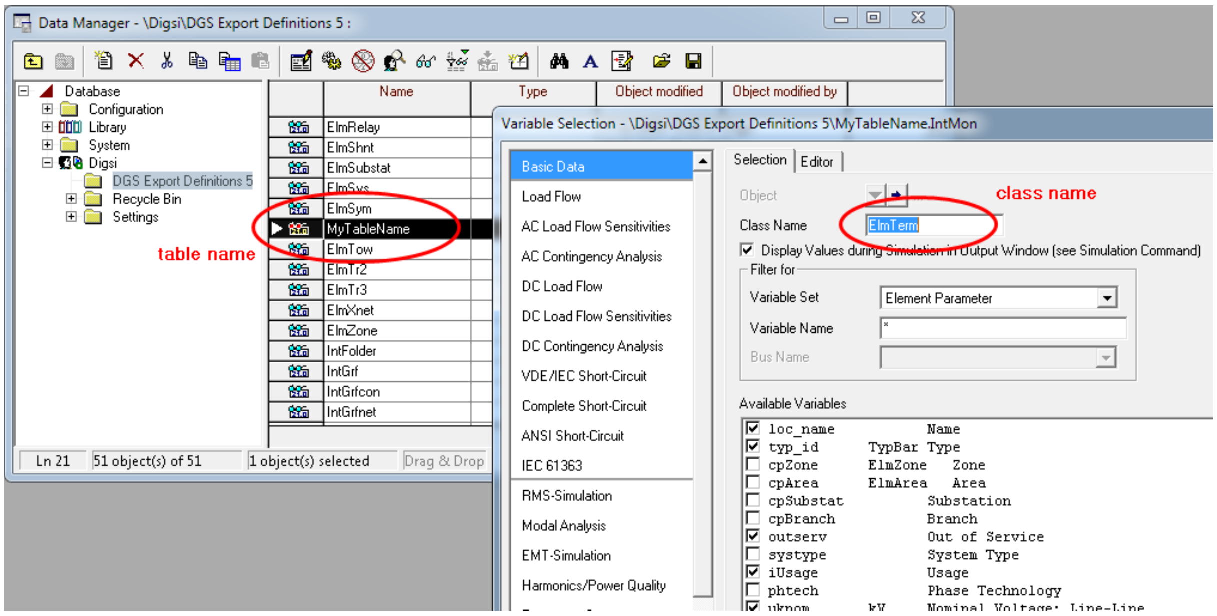
DPL Script

☐ Allow hierarchy and references of exported objects to be incomplete

☑ Allow user-defined table names

When this option is enabled the names of the exported tables can simply be changed by renaming the corresponding variable selection objects ("IntMon") in the export definition: the name of the definition object is now used as table name. (Please be aware of the fact that the class name (in the object) must not be changed and must still be a valid PowerFactory class.)

The picture below demonstrates how the table for "ElmTerm" objects is configured to be exported as "MyTableName".


Data Manager - \Digsi\DGS Export Definitions 5 :

Name	Type	Object modified	Object modified by
ElmRelay			
ElmShnt			
ElmSubstat			
ElmSys			
ElmSym			
MyTableName			
ElmTow			
ElmTr2			
ElmTr3			
ElmXnet			
ElmZone			
IntFolder			
IntGrf			
IntGrfcon			
IntGrfnet			

Ln 21 | 51 object(s) of 51 | 1 object(s) selected | Drag & Drop

Variable Selection - \Digsi\DGS Export Definitions 5\MyTableName.IntMon

Class Name: ElmTerm

☑ Display Values during Simulation in Output Window (see Simulation Command)

Filter for

Variable Set: Element Parameter

Variable Name: *

Bus Name:

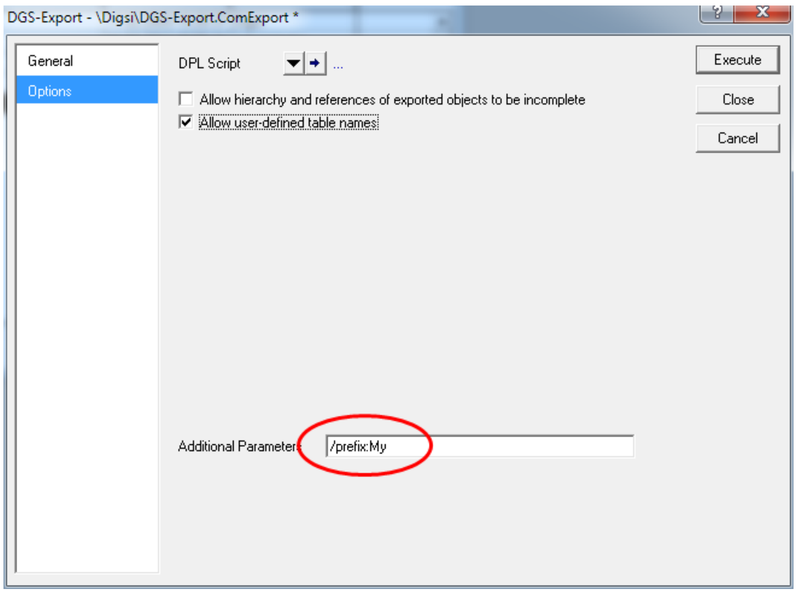
Available Variables

```
☑ loc_name            Name
☑ typ_id     TypBar Type
☐ cpZone     ElmZone   Zone
☐ cpArea     ElmArea   Area
☐ cpSubstat           Substation
☐ cpBranch            Branch
☑ outserv             Out of Service
☐ systype             System Type
☑ iUsage              Usage
☐ phtech              Phase Technology
☑ uknom      kV       Nominal Voltage: Line-Line
```

Beside this table based renaming, it is possible to define a global pre- or postfix for all table names by using the arguments

/prefix:XXX or /postfix:XXX

in the additional parameters field of the export command.


DGS-Export - \Digsi\DGS-Export.ComExport *

[ ] Allow hierarchy and references of exported objects to be incomplete

[✓] Allow user-defined table names

Additional Parameters: /prefix:My

<u>Please note</u>: User-defined table names are supported by the export only. It is **not possible to re-import** those tables again into PowerFactory. For import, the table name must always correspond to the PowerFactory class name!

Appendix

6.6 Common Attributes

The following table describes frequently used PowerFactory attributes:

Attribute	***Data Type***	***Description***
loc_name	Text (max. 40 characters) (a:40)	Name; attribute is available for all PowerFactory objects. Naming rules: In PowerFactory the name must be unique within folders. The following characters cannot be part of a name: * ? = " , \ ~ |
fold_id	Object (p)	Parent object; attribute is available for all PowerFactory objects.
chr_name	Text (max. 20 characters) (a:20)	Characteristic name; attribute is available for all power system elements. This attribute can be used to store a foreign database ID within a PowerFactory object, e.g. a GIS ID. This attribute has only descriptive meaning and cannot be used for object references.
typ_id	Object (p)	Reference to an object type. In PowerFactory, the type specific data is stored in a separate type object.
for_name	Text (max. 20 characters) (a:20)	Foreign-key. If not empty, this name must be unique for all objects within a PowerFactory project. It can be used to reference existing elements from DGS.

6.7 Example of DGS Table Headers

The following chapters provide table definitions for the most frequently used PowerFactory classes. The selected attributes form a basic subset of available attributes that is useful for different tasks and is a good starting point for individual adaptations.

All headers are given in ASCII format as this is the most easiest format to display.

6.7.1 Terminal Data (ElmTerm)

```
$$ElmTerm;ID(a:40);loc_name(a:40);fold_id(p);typ_id(p);chr_name(a:20);iUsage(i);uknom(r)
************************************************************************************
*  Terminal
*
*  ID:          Unique identifier for DGS file
*  loc_name:    Name
*  fold_id:     In Folder
*  typ_id:      Type of the terminal (TypBar)
*  chr_name:    Characteristic Name
*  iUsage:      Usage: 0=Busbar: 1=Junction Node: 2=Internal Node
*  uknom:       Nominal Voltage: Line-Line in kV
************************************************************************************
```

The terminal type can be empty.

6.7.2 Cubicle Data (StaCubic)

```
$$StaCubic;ID(a:40);loc_name(a:40);fold_id(p);obj_bus(i);obj_id(p);chr_name(a:20);
************************************************************************************
*  Cubicle
*
*  ID:          Unique identifier for DGS file
*  loc_name:    Name
*  fold_id:     In Folder
*  obj_bus:     Bus Index (side of connection)
*  obj_id:      Connected with in Elm*
*  chr_name:    Characteristic Name
************************************************************************************
```

Each cubicle belongs to exactly one terminal. In PowerFactory the cubicle is stored within the terminal.

In additional tables CTs and VTs can be added to cubicles.

6.7.3 Line/cable Data (element, ElmLne)

```
$$ElmLne;ID(a:40);loc_name(a:40);fold_id(p);typ_id(p);chr_name(a:20); dline(r);fline(r)
************************************************************************************
*  Line
*
*  ID:          Unique identifier for DGS file
*  loc_name:    Name
*  fold_id:     In Folder
*  typ_id:        Type of Line (TypLne,TypTow,TypGeo)
*  chr_name:      Characteristic Name
*  dline:          Parameters: Length of Line in km
*  fline:         Parameters: Derating Factor
*************************************************************************************
```

6.7.4 Line/cable Data (type, TypLne)

```
$$TypLne;ID(a:40);loc_name(a:40);fold_id(p);chr_name(a:20);Ithr(r);aohl_(a:3);cline(r);cline0(r);nlnph(i);nneutral(i);rline(r);rline0(r);rtemp(r);sline(r);uline(r);xline(r);xline0(r)
*************************************************************************************
*  Line Type
*
*  ID:            Unique identifier for DGS file
*  loc_name:      Name
*  fold_id:       In Folder
*  chr_name:      Characteristic Name
*  Ithr:          Rated Short-Time (1s) Current (Conductor) in kA
*  aohl_:         Cable / OHL
*  cline:         Parameters per Length 1,2-Sequence: Capacitance C' in uF/km
*  cline0:        Parameters per Length Zero Sequence: Capacitance C0' in uF/km
*  nlnph:         Phases:1:2:3
*  nneutral:      No. of Neutrals:0:1
*  rline:         Parameters per Length 1,2-Sequence: Resistance R' in Ohm/km
*  rline0:        Parameters per Length Zero Sequence: Resistance R0' in Ohm/km
*  rtemp:         Max. End Temperature in degC
*  sline:         Rated Current in kA
*  uline:         Rated Voltage in kV
*  xline:         Parameters per Length 1,2-Sequence: Reactance X' in Ohm/km
*  xline0:        Parameters per Length Zero Sequence: Reactance X0' in Ohm/km
*************************************************************************************
```

6.7.5 Line section (element, ElmLnesec)

```
$$ElmLnesec;ID(a:40);loc_name(a:40);fold_id(p);typ_id(p);chr_name(a:20);Inom(r);Unom(r);dline(r);fline(r);index(r)
*************************************************************************************
*  Line Sub-Section
*
*  ID:            Unique identifier for DGS file
*  loc_name:      Name
*  fold_id:       In Folder
*  typ_id:        Type in TypLne,TypTow*
*  chr_name:      Characteristic Name
*  Inom:          Resulting Values: Rated Current in kA
*  Unom:          Nominal Voltage in kV
*  dline:         Topology: Length in km
*  fline:         Topology: Derating Factor
*  index:          Index of the section
*************************************************************************************
```

6.7.6 Switch Data (element, ElmCoup)

```
$$ElmCoup;ID(a:40);loc_name(a:40);fold_id(p);typ_id(p);chr_name(a:20);aUsage(a:4);on_off(i)
********************************************************************************
*  Breaker/Switch
*
*  ID:          Unique identifier for DGS file
*  loc_name:    Name
*  fold_id:     In Folder
*  typ_id:      Type (TypSwitch)
*  chr_name:    Characteristic Name
*  aUsage:      Switch Type cbk=Circuit-Breaker, dct=Disconnector, sdc=Load-Break-Disconnector, swt=Load-Switch
*  on_off:      State, 1=Closed, 0=Open
********************************************************************************
```

The switch type can be empty.

6.7.7 Load Data (element, ElmLod)

```
$$ElmLod;ID(a:40);loc_name(a:40);fold_id(p);typ_id(p);chr_name(a:20);plini(r);qlini(r);scale0(r)
********************************************************************************
*  General Load
*
*  ID:            Unique identifier for DGS file
*  loc_name:      Name
*  fold_id: I     n Folder
*  typ_id:        Type (TypLod,TypLodind)
*  chr_name:      Characteristic Name
*  plini:         Operating Point: Active Power in MW
*  qlini:         Operating Point: Reactive Power in Mvar
*  scale0:        Operating Point: Scaling Factor
********************************************************************************
```

6.7.8 Load Data (type, TypLod)

```
$$TypLod;ID(a:40);loc_name(a:40);fold_id(p);kpu(r);kqu(r);systp(i)
********************************************************************************
*  General Load Type
*
*  ID:            Unique identifier for DGS file
*  loc_name:      Name
*  fold_id:       In Folder
*  kpu:           Voltage Dependence P: Exponent e_cP
*  kqu:           Voltage Dependence Q: Exponent e_cQ
*  systp:         System Type:AC:DC
********************************************************************************
```

6.7.9 External Grid Data (ElmXnet)

```
$$ElmXnet;ID(a:40);loc_name(a:40);fold_id(p);chr_name(a:20);bustp(a:2);cgnd(i);iintgnd(i);ikssmin(r);r0tx0(r);r0tx0min(r);rntxn(r
);rntxnmin(r);snss(r);snssmin(r)
********************************************************************************
*  External Grid
*
*  ID:           Unique identifier for DGS file
*  loc_name:     Name
*  fold_id:      In Folder
*  chr_name:     Characteristic Name
*  bustp:        Bus Type:PQ:PV:SL
*  cgnd:         Internal Grounding Impedance: Star Point:Connected:Not connected
*  iintgnd:       External Star Point
*  ikssmin:      Min. Values: Short-Circuit Current Ik"min in kA
*  r0tx0:        Max. Values Impedance Ratio: R0/X0 max.
*  r0tx0min:     Min. Values Impedance Ratio: R0/X0 min.
*  rntxn: Max.   Values: R/X Ratio (max.)
*  rntxnmin:     Min. Values: R/X Ratio (min.)
*  snss: Max.    Values: Short-Circuit Power Sk"max in MVA
*  snssmin:      Min. Values: Short-Circuit Power Sk"min in MVA
********************************************************************************
```

Hint: An external grid type does not exist.

6.7.10 2-winding Transformer Data (element, ElmTr2)

```
$$ElmTr2;ID(a:40);loc_name(a:40);fold_id(p);typ_id(p);chr_name(a:20);sernum(a:20);constr(i);cgnd_h(i);cgnd_l(i);i_auto(i);nnta
p(i);ntrcn(i);outserv(i);ratfac(r)
********************************************************************************
*  2-Winding Transformer
*
*  ID:           Unique identifier for DGS file
*  loc_name:     Name
*  fold_id:      In Folder
*  typ_id:       Type in TypTr2
*  chr_name:     Characteristic Name
*  sernum:       Serial Number
*  constr:       Year of Construction
*  cgnd_h:        Internal Grounding Impedance, HV Side: Star Point:Connected:Not connected
*  cgnd_l:       Internal Grounding Impedance, LV Side: Star Point:Connected:Not connected
*  i_auto:       Auto Transformer
*  nntap:        Tap: Tap Position
*  ntrcn:        Tap: Automatic Tap Changing
*  outserv:      Out of Service
*  ratfac:       Rating Factor
********************************************************************************
```

6.7.11 2-winding transformer data (type, TypTr2)

```
$$TypTr2;ID(a:40);loc_name(a:40);fold_id(p);chr_name(a:20);curmg(r);dutap(r);frnom(r);manuf(a:20);nntap0(i);nt2ag(i);ntpmn(i);
ntpmx(i);pcutr(r);pfe(r);phitr(r);strn(r);tap_side(i);tr2cn_h(a:2);tr2cn_l(a:2);uk0tr(r);uktr(r);ur0tr(r);utrn_h(r);utrn_l(r);zx0hl_n(r)
********************************************************************************
*  2-Winding Transformer Type
*
*  ID:          Unique identifier for DGS file
*  loc_name:    Name
*  fold_id:     In Folder
*  chr_name:    Characteristic Name
*  curmg:       Magnetizing Impedance: No Load Current in %
*  dutap:       Tap Changer: Additional Voltage per Tap in %
*  frnom:       Nominal Frequency in Hz
*  manuf:       Manufacturer
*  nntap0:      Tap Changer: Neutral Position
*  nt2ag:       Vector Group: Phase Shift in *30deg
*  ntpmn:       Tap Changer: Minimum Position
*  ntpmx:       Tap Changer: Maximum Position
*  pcutr:       Positive Sequence Impedance: Copper Losses in kW
*  pfe:         Magnetizing Impedance: No Load Losses in kW
*  phitr:       Tap Changer: Phase of du in deg
*  strn:         Rated Power in MVA
*  tap_side:    Tap Changer: at Side:HV:LV
*  tr2cn_h:     Vector Group: HV-Side:Y :YN:Z :ZN:D
*  tr2cn_l:     Vector Group: LV-Side:Y :YN:Z :ZN:D
*  uk0tr:       Zero Sequ. Impedance, Short-Circuit Voltage: Absolute uk0 in %
*  uktr:        Positive Sequence Impedance: Short-Circuit Voltage uk in %
*  ur0tr:       Zero Sequ. Impedance, Short-Circuit Voltage: Resistive Part ukr0 in %
*  utrn_h:      Rated Voltage: HV-Side in kV
*  utrn_l:      Rated Voltage: LV-Side in kV
*  zx0hl_n:     Zero Sequence Magnetizing Impedance: Mag. Impedance/uk0
********************************************************************************
```

6.7.12 Shunts (ElmShnt)

```
$$ElmShnt;ID(a:40);loc_name(a:40);fold_id(p);chr_name(a:20);ctech(i);fres(r);greaf0(r);iswitch(i);ncapa(i);ncapx(i);outserv(i);qtot
n(r);shtype(i);ushnm(r)
********************************************************************************
*  Shunt/Filter
*
*  ID:          Unique identifier for DGS file
*  loc_name:     Name
*  fold_id:     In Folder
*  chr_name:    Characteristic Name
*  ctech:       Technology:3PH-'D':3PH-'Y':3PH-'YN':2PH-'Y':2PH-'YN':1PH PH-PH:1PH PH-N:1PH PH-E
*  fres:        Design Parameter (per Step): Resonance Frequency in Hz
*  greaf0:      Design Parameter (per Step): Quality Factor (at fr)
*  iswitch:     Controller: Switchable
*  ncapa:        Controller: Act.No. of Step
*  ncapx:        Controller: Max. No. of Steps
*  outserv:      Out of Service
*  qtotn:        Design Parameter (per Step): Rated Reactive Power, L-C in Mvar
*  shtype:       Shunt Type
*  ushnm:        Nominal Voltage in kV
********************************************************************************
```

Additional tables for network elements and types are available, see examples stored in the "DGS" subfolder of the PowerFactory installation directory.

6.7.13 Single line diagrams (IntGrfnet)

```
$$IntGrfnet;ID(a:40);loc_name(a:40);fold_id(p);grid_on(i);ortho_on(i);snap_on(i);pDataFolder(p)
********************************************************************************
*  Graphic
*
*  ID:            Unique identifier for DGS file
*  loc_name:      Name
*  fold_id:       In Folder
*  grid_on:       Drawing Tools: Grid
*  ortho_on:      Drawing Tools: Ortho Type:
*  snap_on:       Drawing Tools: Snap
*  pDataFolder:   Current Grid Data in ElmNet,BlkDef,ElmStat,IntFolder,ElmSubstat,ElmSite,ElmBranch
********************************************************************************
```

6.7.14 Graphical representation of power system element (IntGrf)

```
$$IntGrf;ID(a:40);loc_name(a:40);fold_id(p);iCol(i);iRot(i);pDataObj(p);rCenterX(r);rCenterY(r);rSizeX(r);rSizeY(r);sSymNam(a:4
0)
********************************************************************************
*  Graphical Net Object
*
*  ID:            Unique identifier for DGS file
*  loc_name:      Name
*  fold_id:       In Folder
*  iCol:          Colour
*  iRot:          Rotation
*  pDataObj:      Net Data Object in Elm*,Sta*,Blk*,RelFuse,IntGrfnet
*  rCenterX:      X-Position
*  rCenterY:      Y-Position
*  rSizeX:        X-Size
*  rSizeY:        Y-Size
*  sSymNam:       Symbol Name
********************************************************************************
```

6.7.15 Graphical connection lines (IntGrfcon)

```
$$IntGrfcon;ID(a:40);loc_name(a:40);fold_id(p);rX:0(r);rX:1(r);rX:2(r);rY:0(r);rY:1(r);rY:2(r)
************************************************************************************
*  GCO
*
*  ID:            Unique identifier for DGS file
*  loc_name:      Name
*  fold_id:       In Folder
*  rX:            X in mm
*  rY:            Y in mm
************************************************************************************
```

6.8 Example DGS - Figure 2: Detailed Transformer Connection

```
$$General;ID(a:40);Descr(a:40);Val(a:40)
**********************************************************************************
*  General Information
*
*  ID:              Unique identifier for DGS file
*  Descr:           Setting
*  Val:             Value
**********************************************************************************
  1;Version;5.0


$$ElmCoup;ID(a:40);loc_name(a:40);fold_id(p);aUsage(a:4);on_off(i)
**********************************************************************************
*  Breaker/Switch
*
*  ID:              Unique identifier for DGS file
*  loc_name:        Name
*  fold_id:         In Folder
*  aUsage:          Switch Type, cbk=Circuit-Breaker, dct=Disconnector, sdc=Load-Break-Disconnector, swt=Load-Switch
*  on_off:          Closed (1=closed, 0=open)
**********************************************************************************
  2;110KV BRK;6;cbk;0
  3;110kV DIS;6;dct;0
  4;20KV BRK;6;cbk;0
  5;20KV DIS;6;dct;0


$$ElmNet;ID(a:40);loc_name(a:40);fold_id(p);frnom(r)
**********************************************************************************
*  Grid
*
*  ID:              Unique identifier for DGS file
*  loc_name:        Name
*  fold_id:         In Folder
*  frnom:     Nominal Frequency in Hz
**********************************************************************************
  6;MS;;50


$$ElmTerm;ID(a:40);loc_name(a:40);fold_id(p);iUsage(i);uknom(r)
**********************************************************************************
*  Terminal
*
*  ID:              Unique identifier for DGS file
*  loc_name:        Name
*  fold_id:          In Folder
*  iUsage:          Usage:0=Busbar:1=Junction Node:2=Internal Node
*  uknom:           Nominal Voltage: Line-Line in kV
```

```
**********************************************************************************
  7;110kV Busbar;6;0;110
  8;20KV Busbar;6;0;20
  9;Internal node 1;6;2;110
  10;Internal node 2;6;2;110
  11;Internal node 3;6;2;20
  12;Internal node 4;6;2;20


$$ElmTr2;ID(a:40);loc_name(a:40);fold_id(p);chr_name(a:20);cgnd_h(i);cgnd_l(i);i_auto(i);nntap(i);ntrcn(i);ratfac(r)
**********************************************************************************
*  2-Winding Transformer
*
*  ID:          Unique identifier for DGS file
*  loc_name:    Name
*  fold_id:     In Folder
*  chr_name:    Characteristic Name
*  cgnd_h:      Internal Grounding Impedance, HV Side: Star Point:Connected:Not connected
*  cgnd_l:      Internal Grounding Impedance, LV Side: Star Point:Connected:Not connected
*  i_auto:      Auto Transformer
*  nntap:       Tap: Tap Position
*  ntrcn:       Tap: Automatic Tap Changing
*  ratfac:      Rating Factor
**********************************************************************************
  13;NT1;6;;0;0;0;4;0;1


$$StaCubic;ID(a:40);loc_name(a:40);fold_id(p);chr_name(a:20);obj_id(p);obj_bus(i)
**********************************************************************************
*  Cubicle
*
*  ID:          Unique identifier for DGS file
*  loc_name:    Name
*  fold_id:     In Folder
*  chr_name:    Characteristic Name
*  obj_id:      Connected with in Elm*
*  obj_bus:     Bus Index
**********************************************************************************
  14;Cub_1;7;;3;1
  15;Cub_1;8;;5;0
  16;Cub_1;9;;3;0
  17;Cub_2;9;;2;1
  18;Cub_1;10;;2;0
  19;Cub_2;10;;13;0
  20;Cub_1;11;;4;1
  21;Cub_2;11;;13;1
  22;Cub_1;12;;5;1
  23;Cub_2;12;;4;0
```

6.9 Example DGS - Figure 3: Simplified Transformer Connection

```
$$General;ID(a:40);Descr(a:40);Val(a:40)
********************************************************************************
*  General Information
*
*  ID:             Unique identifier for DGS file
*  Descr:          Setting
*  Val:            Value
********************************************************************************
  1;Version;5.0


$$ElmNet;ID(a:40);loc_name(a:40);fold_id(p);frnom(r)
********************************************************************************
*  Grid
*
*  ID:               Unique identifier for DGS file
*  loc_name:         Name
*  fold_id:          In Folder
*  frnom:            Nominal Frequency in Hz
********************************************************************************
  2;MS;;50


$$ElmTerm;ID(a:40);loc_name(a:40);fold_id(p);iUsage(i);uknom(r)
********************************************************************************
*  Terminal
*
*  ID:               Unique identifier for DGS file
*  loc_name:         Name
*  fold_id:          In Folder
*  iUsage:           Usage:0=Busbar:1=Junction Node:2=Internal Node
*  uknom:            Nominal Voltage: Line-Line in kV
********************************************************************************
  3;110kV Busbar;2;0;110
  4;20kV Busbar;2;0;20


$$ElmTr2;ID(a:40);loc_name(a:40);fold_id(p);chr_name(a:20);cgnd_h(i);cgnd_l(i);i_auto(i);nntap(i);ntrcn(i);ratfac(r)
********************************************************************************
*  2-Winding Transformer
*
*  ID:               Unique identifier for DGS file
*  loc_name:         Name
*  fold_id:          In Folder
*  chr_name:         Characteristic Name
*  cgnd_h:           Internal Grounding Impedance, HV Side: Star Point:Connected:Not connected
*  cgnd_l:           Internal Grounding Impedance, LV Side: Star Point:Connected:Not connected
*  i_auto:           Auto Transformer
*  nntap:              Tap: Tap Position
*  ntrcn:              Tap: Automatic Tap Changing
*  ratfac:             Rating Factor
********************************************************************************
  5;NT1;2;;0;0;0;4;0;1


$$StaCubic;ID(a:40);loc_name(a:40);fold_id(p);chr_name(a:20);obj_id(p);obj_bus(i)
********************************************************************************
*  Cubicle
*
*  ID:                 Unique identifier for DGS file
*  loc_name:           Name
*  fold_id:             In Folder
*  chr_name:           Characteristic Name
*  obj_id:             Connected with in Elm*
*  obj_bus:            Bus Index
********************************************************************************
  6;Cub_1;3;;5;0
  7;Cub_1;4;;5;1


$$StaSwitch;ID(a:40);loc_name(a:40);fold_id(p);iUse(i);on_off(i)
********************************************************************************
*  Switch
*
*  ID:               Unique identifier for DGS file
*  loc_name:         Name
*  fold_id:          in Cubicle
*  iUse:             Type of Usage
*  on_off:           Closed
********************************************************************************
  8;Switch;6;;1;1
  9;Switch;7;;1;1
```