grg-pssedata Documentation

Release 0.1.4

Carleton Coffrin

Dec 14, 2020

Contents:

1 Introduction **1**
- 1.1 Overview . . . 1
- 1.2 Installation . . . 1
- 1.3 Testing . . . 1

2 grg-pssedata package **3**
- 2.1 grg_pssedata.io module . . . 3
- 2.2 grg_pssedata.cmd module . . . 4
- 2.3 grg_pssedata.exception module . . . 5
- 2.4 grg_pssedata.struct module . . . 5
- 2.5 Module contents . . . 25

3 Indices and tables **27**

Python Module Index **29**

Index **31**

CHAPTER 1

Introduction

1.1 Overview

grg-pssedata is a minimalist python package to support the reading and writing of PSSE network data files.

The primary entry point of the library is *grg_pssedata.io* module, which contains the methods for data input and output.

1.2 Installation

Simply run:

```
pip install grg-pssedata
```

1.3 Testing

grg-pssedata is designed to be a library that supports other software. It is not immediately useful from the terminal. However, you can test the parsing functionality from the command line with:

```
python -m grg_pssedata.io <path to PSSE case file>
```

If this command is successful, you will see a simplified plain text version of the network data printed to the terminal.

CHAPTER 2

grg-pssedata package

2.1 grg_pssedata.io module

class grg_pssedata.io.**LineRequirements**(*line_index*, *min_values*, *max_values*, *section*)
Bases: tuple

Create new instance of LineRequirements(line_index, min_values, max_values, section)

line_index
Alias for field number 0

max_values
Alias for field number 2

min_values
Alias for field number 1

section
Alias for field number 3

grg_pssedata.io.**build_cli_parser**()

grg_pssedata.io.**expand_commas**(*list*)

grg_pssedata.io.**main**(*args*)

grg_pssedata.io.**parse_line**(*line*, *line_reqs=None*)

grg_pssedata.io.**parse_psse_case_file**(*psse_file_name*)
opens the given path and parses it as pss/e data

Parameters **psse_file_name** (*str*) – path to the a psse data file

Returns a grg_pssedata case

Return type *Case*

grg_pssedata.io.**parse_psse_case_lines**(*lines*)

grg_pssedata.io.**parse_psse_case_str**(*psse_string*)
parses a given string as matpower data

Parameters **mpString** (*str*) – a matpower data file as a string

Returns a grg_pssedata case

Return type *Case*

grg_pssedata.io.**print_err**()
print(value, ..., sep=' ', end='n', file=sys.stdout)

Prints the values to a stream, or to sys.stdout by default. Optional keyword arguments: file: a file-like object (stream); defaults to the current sys.stdout. sep: string inserted between values, default a space. end: string appended after the last value, default a newline.

2.2 grg_pssedata.cmd module

functions for analyzing and transforming psse data files

grg_pssedata.cmd.**build_cmd_parser**()

grg_pssedata.cmd.**compare_component_lists**(*list_1*, *list_2*, *comp_name*, *index_name='index'*)
compares two lists and prints the differences to stdout. Objects in the lists are assumed to have an identification attribute.

Parameters

- **list_1** (*list*) – the first list
- **list_2** (*list*) – the second list
- **comp_name** (*string*) – the name of components being compared
- **index_name** (*string*) – the name of the object identification attribute

Returns (int): returns the number of items that differed in the two lists

grg_pssedata.cmd.**diff**(*case_1*, *case_2*)
Compares two *grg_pssedata.struct.Case* objects and prints the differences to stdout.

Parameters

- **case_1** – the first psse case
- **case_2** – the second psse case

Returns (int): returns the number of items that differed in the two cases

grg_pssedata.cmd.**eq**(*case_1*, *case_2*)

grg_pssedata.cmd.**main**(*args*)
reads a psse case files and processes them based on command line arguments.

Parameters **args** – an argparse data structure

2.3 grg_pssedata.exception module

a collection of all grg_pssedata exception classes

exception grg_pssedata.exception.**PSSEDataException**
Bases: exceptions.Exception

root class for all PSSEData Exceptions

exception grg_pssedata.exception.**PSSEDataParsingError**
Bases: *grg_pssedata.exception.PSSEDataException*

for errors that occur while attempting to parse a pss/e data file

exception grg_pssedata.exception.**PSSEDataValidationError**
Bases: *grg_pssedata.exception.PSSEDataException*

for errors that occur while attempting to validate the correctness of a parsed pss/e data file

exception grg_pssedata.exception.**PSSEDataWarning**
Bases: exceptions.Warning

root class for all PSSEData Warnings

2.4 grg_pssedata.struct module

data structures for encoding pss/e data files

class grg_pssedata.struct.**Area**(*i*, *isw='0'*, *pdes='0.0'*, *ptol='0.0'*, *arnam=''*)
Bases: object

This data structure contains area interchange parameters.

Parameters

- **i** (*int*) – the identifier of the area
- **isw** (*int*) – the identifier of the interchange slack bus
- **pdes** (*float*) – desired net area interchange (MW)
- **ptol** (*float*) – interchange tolerance (MW += out)
- **arnam** (*string*) – area name, 8 characters, must be enclosed in single quotes

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**Branch**(*index*, *i*, *j*, *ckt*, *r*, *x*, *b*, *ratea*, *rateb*, *ratec*, *gi*, *bi*, *gj*, *bj*, *st*, *met*, *len*, *o1*, *f1*, *o2=0*, *f2=1.0*, *o3=0*, *f3=1.0*, *o4=0*, *f4=1.0*)
Bases: object

This data structure contains branch parameters.

Parameters

- **index** (*int*) – unique branch identifier
- **i** (*int*) – the identifier of the from bus

- **j** (*int*) – the identifier of the to bus, (note a leading minus indicates that the meter is on the from side of the line)
- **ckt** (*str*) – circuit identifier
- **r** (*float*) – the branch resistance (p.u.)
- **x** (*float*) – the branch reactance (p.u.)
- **b** (*float*) – the total branch charging susceptance (p.u.)
- **ratea** (*float*) – base rating (MVA)
- **rateb** (*float*) – shorter rating (MVA)
- **ratec** (*float*) – shortest rating (MVA)
- **gi** (*float*) – line shunt conductance at from end (bus i) (p.u.)
- **bi** (*float*) – line shunt susceptance at from end (bus i) (p.u.)
- **gj** (*float*) – line shunt conductance at to end (bus j) (p.u.)
- **bj** (*float*) – line shunt susceptance at to end (bus j) (p.u.)
- **st** (*int*) – branch status (in service = 1, out of service = 0)
- **met** (*int*) – metered end flag (<= 1 indicates the i-bus, >= 2 indicates the j-bus)
- **len** (*float*) – line length (user selected units)
- **o1** (*int*) – owner one id, 1-9999 (default = the owner of the connecting bus)
- **f1** (*float*) – owner one fraction of total ownership (default = 1.0)
- **o2** (*int*) – owner two id, 1-9999 (default = the owner of the connecting bus)
- **f2** (*float*) – owner two fraction of total ownership (default = 1.0)
- **o3** (*int*) – owner three id, 1-9999 (default = the owner of the connecting bus)
- **f3** (*float*) – owner three fraction of total ownership (default = 1.0)
- **o4** (*int*) – owner four id, 1-9999 (default = the owner of the connecting bus)
- **f4** (*float*) – owner four fraction of total ownership (default = 1.0)

is_breaker()

is_from_metered()

is_switch()

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**Bus**(*i*, *name*, *basekv*, *ide*, *area*, *zone*, *owner*, *vm*, *va*, *nvhi=1.1*, *nvlo=0.9*, *evhi=1.1*, *evlo=0.9*)

Bases: object

This data structure contains bus parameters.

Parameters

- **i** (*int*) – unique bus identifier 1-999997
- **name** (*string*) – bus name, 8 characters, must be enclosed in single quotes

- **basekv** (*float*) – base voltage (kilo volts)
- **ide** (*int*) – bus type, PQ = 1, PV = 2, reference = 3, isolated = 4
- **area** (*int*) – area id, 1-9999 (default = 1)
- **zone** (*int*) – zone id, 1-9999 (default = 1)
- **owner** (*int*) – owner id, 1-9999 (default = 1)
- **vm** (*float*) – voltage magnitude (volts p.u.) (default = 1.0)
- **va** (*float*) – voltage angle (degrees) (default = 0.0)
- **nvhi** (*float*) – voltage magnitude upper bound, normal conditions (volts p.u.) (default = 1.1)
- **nvlo** (*float*) – voltage magnitude lower bound, normal conditions (volts p.u.) (default = 0.9)
- **evhi** (*float*) – voltage magnitude upper bound, emergency conditions (volts p.u.) (default = 1.1)
- **evlo** (*float*) – voltage magnitude lower bound, emergency conditions (volts p.u.) (default = 0.9)

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**Case**(*ic*, *sbase*, *rev*, *xfrrat*, *nxfrat*, *basfrq*, *record1*, *record2*, *buses*, *loads*, *fixed_shunts*, *generators*, *branches*, *transformers*, *areas*, *tt_dc_lines*, *vsc_dc_lines*, *transformer_corrections*, *mt_dc_lines*, *line_groupings*, *zones*, *transfers*, *owners*, *facts*, *switched_shunts*, *gnes*, *induction_machines*)

Bases: object

This data structure contains lists of all the key components in a pss/e power network. At this time, only pss/e case version 33 is supported.

Parameters

- **ic** (*int*) – case type, 0 for base, 1 for incremental
- **sbase** (*float*) – the system MVA base value (MVA)
- **rev** (*int*) – file type version number
- **xfrrat** (*float*) – transformer rating units
- **nxfrat** (*float*) – units of branch ratings
- **basfrq** (*float*) – base frequency (Hertz)
- **record1** (*string*) – system description part 1, up to 60 characters
- **record2** (*string*) – system description part 2, up to 60 characters
- **buses** (*list of Bus*) – buses
- **loads** (*list of Load*) – loads
- **fixed_shunts** (*list of TBD*) – fixed shunts
- **generators** (*list of Generator*) – generators

- **branches** (*list of Branch*) – branches
- **transformers** (*list of TwoWindingTransformer and ThreeWindingTransformer*) – two and three winding transformers
- **areas** (*list of Area*) – areas
- **tt_dc_lines** (*list of TBD*) – two-terminal dc lines
- **vsc_dc_lines** (*list of TBD*) – vsc dc lines
- **transformer_corrections** (*list of TBD*) – transformer correction tables
- **mt_dc_lines** (*list of TBD*) – multi-terminal dc lines
- **line_groupings** (*list of TBD*) – line groupings
- **zones** (*list of Zone*) – zones
- **transfers** (*list of TBD*) – inter-area transfers
- **owners** (*list of Owner*) – owners
- **facts** (*list of TBD*) – FACTS devices
- **switched_shunts** (*list of SwitchedShunt*) – switched shunt devices
- **gnes** (*list of TBD*) – general network elements
- **induction_machines** (*list of TBD*) – induction machines

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification.

class grg_pssedata.struct.**FACTSDevice**(*index*, *name*, *i*, *j*, *mode*, *pdes*, *qdes*, *vset*, *shmx*, *trmx*, *vtmn*, *vtmx*, *vsmx*, *imx*, *linx*, *rmpct*, *owner*, *set1*, *set2*, *vsref*, *remot='0'*, *mname=''*)

Bases: object

This data structure contains FACTS device parameters.

Parameters

- **index** (*int*) – unique FACTS device identifier
- **name** (*str*) – name of device
- **i** (*int*) – sending end bus number
- **j** (*int*) – terminal end bus number
- **mode** (*int*) – control mode (status)
- **pdes** (*float*) – desired active power flow at j
- **qdes** (*float*) – desired reactive power flow at j
- **vset** (*float*) – voltage setpoint at i
- **shmx** (*float*) – max shunt current at i
- **trmx** (*float*) – max active power transfer
- **vtmn** (*float*) – min voltage at j
- **vtmx** (*float*) – max voltage at j

- **vsmx** (*float*) – max series voltage
- **imx** (*float*) – max series current
- **linx** (*float*) – reactance of dummy series element
- **rmpct** (*float*) – percent of Mvar required to hold voltage at bus controlled by shunt element
- **owner** (*int*) – owner number
- **set1** (*float*) – resistance of constant impedance, or min of constant series voltage magnitude, or real component of constant series voltage
- **set2** (*float*) – reactance of constant impedance, or max of constant series voltage magnitude, or imaginary component of constant series voltage
- **vsref** (*int*) – series voltage reference code
- **remot** (*int*) – bus number of bus regulated by shunt element
- **mname** (*str*) – name of the IPFC master FACTS Device

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**FixedShunt**(*index*, *i*, *id*, *status*, *gl*, *bl*)
Bases: object

This data structure contains fixed shunt parameters

Parameters

- **index** (*int*) – unique fixed shunt identifier
- **i** (*int*) – the identifier of the bus that this fixed shunt is connected to
- **id** (*string*) – fixed shunt identifier (not unique)
- **status** (*int*) – fixed shunt status (in service = 1, out of service = 0)
- **gl** (*float*) – the conductance to ground in MW at one per unit voltage (default = 0.0)
- **bl** (*float*) – the susceptance to ground in MVar at one per unit voltage (default = 0.0)

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**Generator**(*index*, *i*, *id*, *pg*, *qg*, *qt*, *qb*, *vs*, *ireg*, *mbase*, *zr*, *zx*, *rt*, *xt*, *gtap*, *stat*, *rmpct*, *pt*, *pb*, *o1=1*, *f1=0*, *o2=0*, *f2=0*, *o3=1.0*, *f3=1.0*, *o4=1.0*, *f4=1.0*, *wmod=0*, *wpf=1.0*)
Bases: object

This data structure contains generator parameters.

Parameters

- **index** (*int*) – unique generator identifier
- **i** (*int*) – the identifier of the bus that this generator is connected to
- **id** (*string*) – machine identifier (not unique)

- **pg** (*float*) – active power output (MW)
- **qg** (*float*) – reactive power output (MVAr)
- **qt** (*float*) – reactive power output upper bound (MVAr)
- **qb** (*float*) – reactive power output lower bound (MVAr)
- **vs** (*float*) – voltage magnitude setpoint (volts p.u.)
- **ireg** (*int*) – Remote controlled bus index (must be type 1), zero to control own voltage, and must be zero for gen at swing bus
- **mbase** (*float*) – machine mva base (MVA)
- **zr** (*float*) – machine resistance, pu on MBASE
- **zx** (*float*) – machine reactance, pu on MBASE
- **rt** (*float*) – step up transformer resistance, p.u. on MBASE
- **xt** (*float*) – step up transformer reactance, p.u. on MBASE
- **gtap** (*float*) – step up transformer off nominal turns ratio
- **stat** (*int*) – generator status (in service = 1, out of service = 0)
- **rmpct** (*float*) – percent of total VARS required to hold voltage at bus IREG to come from bus I - for remote buses controlled by several generators
- **pt** (*float*) – active power output upper bound (MW)
- **pb** (*float*) – active power output lower bound (MW)
- **o1** (*int*) – owner one id, 1-9999 (default = the owner of the connecting bus)
- **f1** (*float*) – owner one fraction of total ownership (default = 1.0)
- **o2** (*int*) – owner two id, 1-9999 (default = the owner of the connecting bus)
- **f2** (*float*) – owner two fraction of total ownership (default = 1.0)
- **o3** (*int*) – owner three id, 1-9999 (default = the owner of the connecting bus)
- **f3** (*float*) – owner three fraction of total ownership (default = 1.0)
- **o4** (*int*) – owner four id, 1-9999 (default = the owner of the connecting bus)
- **f4** (*float*) – owner four fraction of total ownership (default = 1.0)
- **wmod** (*int*) – wind machine control mode, not-wind = 0, Q limits = 1, P limits as Q limits = 2, power factor = 4
- **wpf** (*float*) – wind power factor (used when wmod is 2 or 3)

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**InductionMachine**(*index*, *i*, *id*, *stat*, *scode*, *dcode*, *area*, *zone*, *owner*, *tcode*, *bcode*, *mbase*, *ratekv*, *pcode*, *pset*, *h*, *a*, *b*, *d*, *e*, *ra*, *xa*, *xm*, *r1*, *x1*, *r2*, *x2*, *x3*, *e1*, *se1*, *e2*, *se2*, *ia1*, *ia2*, *xamult*)

Bases: object

This data structure contains Induction Machine parameters

Parameters

- **index** (*int*) – unique interarea transfer identifier
- **i** (*int*) – bus number
- **id** (*str*) – machine identifier
- **stat** (*int*) – status
- **scode** (*int*) – standard code
- **dcode** (*int*) – design code
- **area** (*int*) – area
- **zone** (*int*) –
- **owner** (*int*) – owner
- **tcode** (*int*) – type of mechanical load torque
- **bcode** (*int*) – base power code
- **mbase** (*float*) – base power
- **ratekv** (*float*) – rated voltage
- **pcode** (*int*) – scheduled power code
- **pset** (*float*) – scheduled active power
- **h** (*float*) – machine inertia
- **a** (*float*) – constant describing variation of torque with speed
- **b** (*float*) – constant describing variation of torque with speed
- **d** (*float*) – constant describing variation of torque with speed
- **e** (*float*) – constant describing variation of torque with speed
- **ra** (*float*) – armature resistance
- **xa** (*float*) – armature leakage reactance
- **xm** (*float*) – unsaturated magnetizing reactance
- **r1** (*float*) – resistance of first rotor winding
- **x1** (*float*) – reactance of first rotor winding
- **r2** (*float*) – resistance of second rotor winding
- **x2** (*float*) – reactance of second rotor winding
- **x3** (*float*) – reactance of third rotor winding
- **e1** (*float*) – first terminal voltage point
- **se1** (*float*) – saturation factor at terminal voltage e1
- **e2** (*float*) – second terminal voltage point
- **se2** (*float*) – saturation factor at terminal voltage e2
- **ia1** (*float*) – stator current
- **ia2** (*float*) – stator current
- **xamult** (*float*) – multiplier for saturated value

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**InterareaTransfer**(*index*, *arfrom*, *arto*, *trid*, *ptran*)
Bases: object

This data structure contains Interarea Transfer parameters

Parameters

- **index** (*int*) – unique interarea transfer identifier
- **arfrom** (*int*) – from area identifier
- **arto** (*int*) – to area identifier
- **trid** (*str*) – interarea transfer identifier
- **ptran** (*float*) – MW of transfer

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**Load**(*index*, *i*, *id*, *status*, *area*, *zone*, *pl*, *ql*, *ip*, *iq*, *yp*, *yq*, *owner*, *scale*, *intrpt='0'*)
Bases: object

This data structure contains load parameters.

Parameters

- **index** (*int*) – unique load identifier
- **i** (*int*) – the identifier of the bus that this load is connected to
- **id** (*string*) – load identifier (not unique)
- **status** (*int*) – load status (in service = 1, out of service = 0)
- **area** (*int*) – area id, 1-9999 (default = the area of the connecting bus)
- **zone** (*int*) – zone id, 1-9999 (default = the zone of the connecting bus)
- **pl** (*float*) – active power load (MW) (default = 0.0)
- **ql** (*float*) – reactive power output (MVAr) (default = 0.0)
- **ip** (*float*) – real current load (MW per unit voltage) (default = 0.0)
- **iq** (*float*) – imaginary current load (MVAr per unit voltage) (default = 0.0)
- **yp** (*float*) – real admittance load (MW per unit voltage) (default = 0.0)
- **yq** (*float*) – imaginary admittance load (MVAr per unit voltage) (default = 0.0)
- **owner** (*int*) – owner id, 1-9999 (default = the owner of the connecting bus)
- **scale** (*int*) – scaling flag (scalable = 1, fixed = 0) (default = 1)
- **intrpt** (*int*) – interruptible flag, (interruptible = 1, non-interruptible = 0) (optional, default = 0)

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**MultiSectionLineGrouping**(*index*, *i*, *j*, *id*, *met*, **dumi*)
Bases: object

This data structure contains Multi-Section Line Grouping parameters

Parameters

- **index** (*int*) – unique transformer impedence correction identifier
- **i** (*int*) – from bus number
- **j** (*int*) – to bus number
- **id** (*str*) – grouping identifier
- **met** (*int*) – metered end flag
- **dumi** (*int*) – bus numbers of dummy buses

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**MultiTerminalDCLine**(*index*, *params*, *nconv*, *ndcbs*, *ndcln*)
Bases: object

This data structure contains Multi-Terminal DC Line parameters

Parameters

- **index** (*int*) – unique interarea transfer identifier
- **params** (MultiTerminalDCLineParameters) – first line of MultiTerminal DC Line entry
- **nconv** (*list(*MultiTerminalDCLineConverter*)*) – next nconv lines of MutliTerminal DC Line entries
- **ndcbs** (*list(*MultiTerminalDCLineDCBus*)*) – next ndcbs lines of MutliTerminal DC Line entries
- **ndcln** (*list(*MultiTerminalDCLineDCLink*)*) – next ndcln lines of MutliTerminal DC Line entries

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**MultiTerminalDCLineConverter**(*ib*, *n*, *angmx*, *angmn*, *rc*, *xc*, *ebas*, *tr*, *tap*, *tpmx*, *tpmn*, *tstp*, *setvl*, *dcpf*, *marg*, *cnvcod*)
Bases: object

This data structure contains MultiTerminal DC Line Converter parameters

Parameters

- **ib** (*int*) – ac converter bus number
- **n** (*int*) – number of bridges in series
- **angmx** (*float*) – nominal max alpha or gamma angle
- **angmn** (*float*) – nominal min alpha or gamma angle
- **rc** (*float*) – commutating resistance per bridge
- **xc** (*float*) – commutating reactance per bridge
- **ebas** (*float*) – primary base ac voltage
- **tr** (*float*) – actual transformer ratio
- **tap** (*float*) – tap setting
- **tpmx** (*float*) – max tap setting
- **tpmn** (*float*) – min tap setting
- **tstp** (*float*) – tap step
- **setvl** (*float*) – converter setpoint
- **dcpf** (*float*) – converter participation factor
- **marg** (*float*) – rectifier margin
- **cnvcod** (*int*) – converter code

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**MultiTerminalDCLineDCBus**(*idc*, *ib*, *area*, *zone*, *dcname*, *idc2*, *rgrnd*, *owner*)

Bases: object

This data structure contains MultiTerminal DC Line DC Bus parameters

Parameters

- **idc** (*int*) – dc bus number
- **ib** (*int*) – ac converter bus number
- **area** (*int*) – area number
- **zone** (*int*) – zone number
- **dcname** (*str*) – identifier
- **idc2** (*int*) – second dc bus to which idc is connected
- **rgrnd** (*float*) – resistance to ground
- **owner** (*int*) – owner number

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**MultiTerminalDCLineDCLink**(*idc*, *jdc*, *dcckt*, *met*, *rdc*, *ldc*)
Bases: object

This data structure contains MultiTerminal DC Line DC Link parameters

Parameters

- **idc** (*int*) – from bus dc number
- **jdc** (*int*) – to bus dc number
- **dcckt** (*str*) – circuit identifier
- **met** (*int*) – metered end flag
- **rdc** (*float*) – dc link resistance
- **ldc** (*float*) – dc link inductance

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**MultiTerminalDCLineParameters**(*name*, *nconv*, *ndcbs*, *ndcln*, *mdc*, *vconv*, *vcmod*, *vconvn*)
Bases: object

This data structure contains MultiTerminal DC Line first-line parameters

Parameters

- **name** (*str*) – name of dc line
- **nconv** (*int*) – number of ac converters
- **ndcbs** (*int*) – number of dc buses
- **ndcln** (*int*) – number of dc links
- **mdc** (*int*) – control mode
- **vconv** (*int*) – bus number of positive pole controlling ac converter station
- **vcmod** (*float*) – mode switch dc voltage
- **vconvn** (*int*) – bus number of negative pole controlling ac converter station

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**Owner**(*i*, *ownname*)
Bases: object

This data structure contains owner parameters.

Parameters

- **i** (*int*) – the identifier of the owner
- **owname** (*string*) – owner name

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**SwitchedShunt**(*index*, *i*, *modsw*, *adjm*, *stat*, *vswhi*, *vswlo*, *swrem*, *rmpct*, *rmidnt*, *binit*, *n1*, *b1*, *n2=None*, *b2=None*, *n3=None*, *b3=None*, *n4=None*, *b4=None*, *n5=None*, *b5=None*, *n6=None*, *b6=None*, *n7=None*, *b7=None*, *n8=None*, *b8=None*)

Bases: object

This data structure contains switch shunt parameters

Parameters

- **index** (*int*) – unique switched shunt identifier
- **i** (*int*) – the identifier of the bus that this switched shunt is connected to
- **modsw** (*int*) – control mode (locked = 0, discrete = 1, continuous = 2, ... 6) (default = 1)
- **adjm** (*int*) – adjustment method (input order = 0, admittance order = 1) (default = 0)
- **stat** (*int*) – switched shunt status (in service = 1, out of service = 0)
- **vswhi** (*float*) – control parameter upper bound (default = 1.0)
- **vswlo** (*float*) – control parameter lower bound (default = 1.0)
- **swrem** (*int*) – bus id to monitor reactive power (default = 0)
- **rmpct** (*float*) – percentage of reactive power this shunt should contribute (default = 100.0)
- **rmidnt** (*string*) – the name of the dc line or facts device where the reactive output should be controlled
- **binit** (*float*) – initial shunt susceptance (MVAr per unit voltage) (default = 0.0)
- **n1** (*int*) – the number of steps in block 1 (default = 0)
- **b1** (*float*) – the susceptance increment of block 1 (default = 0.0)
- **n2** (*int*) – the number of steps in block 2 (default = 0)
- **b2** (*float*) – the susceptance increment of block 2 (default = 0.0)
- **n3** (*int*) – the number of steps in block 3 (default = 0)
- **b3** (*float*) – the susceptance increment of block 3 (default = 0.0)
- **n4** (*int*) – the number of steps in block 4 (default = 0)
- **b4** (*float*) – the susceptance increment of block 4 (default = 0.0)
- **n5** (*int*) – the number of steps in block 5 (default = 0)
- **b5** (*float*) – the susceptance increment of block 5 (default = 0.0)
- **n6** (*int*) – the number of steps in block 6 (default = 0)
- **b6** (*float*) – the susceptance increment of block 6 (default = 0.0)
- **n7** (*int*) – the number of steps in block 7 (default = 0)
- **b7** (*float*) – the susceptance increment of block 7 (default = 0.0)
- **n8** (*int*) – the number of steps in block 8 (default = 0)
- **b8** (*float*) – the susceptance increment of block 8 (default = 0.0)

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**ThreeWindingTransformer**(*index*, *p1*, *p2*, *w1*, *w2*, *w3*)
Bases: object

This data structure contains three winding transformer parameters.

Parameters

- **index** (*int*) – unique transformer identifier
- **p1** (TransformerParametersFirstLine) – first line of parameters
- **p2** (TransformerParametersSecondLine) – second line of parameters
- **w1** (TransformerWinding) – first winding data
- **w2** (TransformerWinding) – second winding data
- **w3** (TransformerWinding) – third winding data

is_three_winding()

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TransformerImpedanceCorrection**(*index*, *i*, *t1='0.0'*, *f1='0.0'*, *t2='0.0'*, *f2='0.0'*, *t3='0.0'*, *f3='0.0'*, *t4='0.0'*, *f4='0.0'*, *t5='0.0'*, *f5='0.0'*, *t6='0.0'*, *f6='0.0'*, *t7='0.0'*, *f7='0.0'*, *t8='0.0'*, *f8='0.0'*, *t9='0.0'*, *f9='0.0'*, *t10='0.0'*, *f10='0.0'*, *t11='0.0'*, *f11='0.0'*)
Bases: object

This data structure contains Transformer Impedence Correction Table parameters

Parameters

- **index** (*int*) – unique transformer impedence correction identifier
- **i** (*int*) – transformer impedence correction table number
- **ti** (*float*) – off-nominal turns ratio or phase shift angle
- **fi** (*float*) – scaling factor

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TransformerParametersFirstLine**(*i*, *j*, *k*, *ckt*, *cw*, *cz*, *cm*, *mag1*, *mag2*, *nmetr*, *name*, *stat*, *o1*, *f1*, *o2*, *f2*, *o3*, *f3*, *o4*, *f4*, *vecgrp=' '*)

Bases: object

This data structure contains transformer parameters that are common to two and three winding transformers.

Parameters

- **i** (*int*) – the identifier of the primary bus
- **j** (*int*) – the identifier of the secondary bus
- **k** (*int*) – the identifier of the tertiary bus (0 if a two winding transformer)
- **ckt** (*string*) – circuit identifier
- **cw** (*int*) – turn ratio units (1 = off-nominal pu winding bus base voltage, 2 = voltage kc, 3 = off-nominal pu nominal winding voltage)
- **cz** (*int*) – winding impedance units (1 = pu on system mva base, 2 = pu on specified mva base, 3 =)
- **cm** (*int*) – mag units (1 = pu on system mva, 2 =)
- **mag1** (*float*) – ground conductance on the primary bus
- **mag2** (*float*) – ground susceptance on the primary bus
- **nmetr** (*int*) – the nonmetered end of the transformer the primary, secondary, and tertiary buses are specified by 1,2,3 respectively
- **name** (*string*) – name of the transformer
- **stat** (*int*) – transformer status (0 = out of service, 1 = in service, 2 = winding 2 out, 3 = winding 3 out, 4 = winding 1 out)
- **o1** (*int*) – owner one id, 1-9999 (default = the owner of the connecting bus)
- **f1** (*float*) – owner one fraction of total ownership (default = 1.0)
- **o2** (*int*) – owner two id, 1-9999 (default = the owner of the connecting bus)
- **f2** (*float*) – owner two fraction of total ownership (default = 1.0)
- **o3** (*int*) – owner three id, 1-9999 (default = the owner of the connecting bus)
- **f3** (*float*) – owner three fraction of total ownership (default = 1.0)
- **o4** (*int*) – owner four id, 1-9999 (default = the owner of the connecting bus)
- **f4** (*float*) – owner four fraction of total ownership (default = 1.0)
- **vecgrp** (*string*) – vector group identifier (default = ' ')

to_psse()

Returns: a pss/e encoding of this data structure as a string

validate(*transformer_id*)

Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TransformerParametersSecondLine**(*r12*, *x12*, *sbase12*, *r23*, *x23*, *sbase23*, *r31*, *x31*, *sbase31*, *vmstar*, *anstar*)

Bases: object

This data structure contains transformer parameters for the second line of three winding transformers.

Parameters

- **r12** (*float*) – resistance between terminal i and j (default 0.0)
- **x12** (*float*) – reactance between terminal i and j
- **sbase12** (*float*) – the MVA base between terminal i and j
- **r23** (*float*) – resistance between terminal j and k (default 0.0)
- **x23** (*float*) – reactance between terminal j and k
- **sbase23** (*float*) – the MVA base between terminal j and k
- **r31** (*float*) – resistance between terminal k and i (default 0.0)
- **x31** (*float*) – reactance between terminal k and i
- **sbase31** (*float*) – the MVA base between terminal k and i
- **vmstar** (*float*) – the voltage magnitude of the start point (default 1.0)
- **anstar** (*float*) – the voltage angle of the start point (default 0.0)

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate(*transformer_id*)
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TransformerParametersSecondLineShort**(*r12*, *x12*, *sbase12*)

Bases: object

This data structure contains transformer parameters for the second line of two winding transformers.

Parameters

- **r12** (*float*) – resistance between terminal i and j (default 0.0)
- **x12** (*float*) – reactance between terminal i and j
- **sbase12** (*float*) – the MVA base between terminal i and j

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate(*transformer_id*)
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TransformerWinding**(*index*, *windv*, *nomv*, *ang*, *rata*, *ratb*, *ratc*, *cod*, *cont*, *rma*, *rmi*, *vma*, *vmi*, *ntp*, *tab*, *cr*, *cx*, *cnxa=0.0*)

Bases: object

This data structure contains transformer winding parameters.

Parameters

- **index** (*int*) – transformer winding identifier (1,2,3)
- **windv** (*float*) – off-nominal turn ratio (p.u.) (default = 1.0)
- **nomv** (*float*) – base voltage (kilo volts)
- **ang** (*float*) – angle shift (degrees)

- **rata** (*float*) – base rating (MVA)
- **ratb** (*float*) – shorter rating (MVA)
- **ratc** (*float*) – shortest rating (MVA)
- **cod** (*int*) – transformer control mode
- **cont** (*int*) – remote bus index for transformer voltage control ()
- **rma** (*float*) – off-nominal turn ratio upper bound
- **rmi** (*float*) – off-nominal turn ratio lower bound
- **vma** (*float*) – controller band upper limit
- **vmi** (*float*) – controller band lower limit
- **ntp** (*int*) – number of tap positions available 2-9999 (default = 33)
- **tab** (*int*) – the identifier of the transformer impedance correction table
- **cr** (*float*) – load drop compensation resistance (p.u.)
- **cx** (*float*) – load drop compensation reactance (p.u.)
- **cnxa** (*float*) – winding connection angle (degrees) (default = 0.0)

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate(*transformer_id*)
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TransformerWindingShort**(*index*, *windv*, *nomv*)
Bases: object

This data structure contains the shortend transformer winding parameters for the secondary side of a two winding transformer

Parameters

- **index** (*int*) – transformer winding identifier (1,2,3)
- **windv** (*float*) – off-nominal turn ratio (p.u.) (default = 1.0)
- **nomv** (*float*) – base voltage (kilo volts)

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate(*transformer_id*)
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TwoTerminalDCLine**(*index*, *params*, *rectifier*, *inverter*)
Bases: object

This data structure contains Two-Terminal DC Line parameters.

Parameters

- **index** (*int*) – unique two-terminal dc line identifier
- **params** (TwoTerminalDCLineParameters) – first line of two-terminal dc line data entry
- **rectifier** (TwoTerminalDCLineRectifier) – second line of two-terminal dc line data entry

- **inverter** (TwoTerminalDCLineInverter) – third line of two-terminal dc line data entry

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TwoTerminalDCLineInverter**(*ipi*, *nbi*, *anmxi*, *anmni*, *rci*, *xci*, *ebasi*, *tri*, *tapi*, *tmxi*, *tmni*, *stpi*, *ici*, *ifi*, *iti*, *idi*, *xcapi*)

Bases: object

This data structure contains Two-Terminal DC Line parameters for the Inverter (third line)

Parameters

- **ipi** (*int*) – bus number
- **nbi** (*int*) – number of bridges
- **anmxi** (*float*) – nominal max firing angle
- **anmni** (*float*) – minimum steady-state firing angle
- **rci** (*float*) – transformer resistance per bridge
- **xci** (*float*) – transformer reactance per bridge
- **ebasi** (*float*) – primary base ac voltage
- **tri** (*float*) – transformer ratio
- **tapi** (*float*) – tap setting
- **tmxi** (*float*) – max tap setting
- **tmni** (*float*) – min tap setting
- **stpi** (*float*) – tap step
- **ici** (*int*) – firing angle measuring bus number
- **ifi** (*int*) – winding 1 side from bus number
- **iti** (*int*) – winding 2 side to bus number
- **idi** (*str*) – circuit identifier
- **xcapi** (*float*) – capacitor resistance per bridge

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TwoTerminalDCLineParameters**(*name*, *mdc*, *rdc*, *setvl*, *vschd*, *vcmod*, *rcomp*, *delti*, *meter*, *dcvmin*, *cccitmx*, *cccacc*)

Bases: object

This data structure contains Two-Terminal DC Line parameters for the first line of the data entry

Parameters

- **name** (*str*) – name of the dc line

- **mdc** (*int*) – control mode (status)
- **rdc** (*float*) – dc line resistance
- **setvl** (*float*) – current or power demand
- **vschd** (*float*) – scheduled dc voltage
- **vcmod** (*float*) – switch dc voltage
- **rcomp** (*float*) – compounding resistance
- **delti** (*float*) – margin of desired dc power or current
- **meter** (*str*) – metered end code
- **dcvmin** (*float*) – minimum compounded dc voltage
- **cccitmx** (*int*) – iteration limit for solution proceedure
- **cccacc** (*float*) – acceleration factor for solution proceedure

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TwoTerminalDCLineRectifier**(*ipr*, *nbr*, *anmxr*, *anmnr*, *rcr*, *xcr*, *ebasr*, *trr*, *tapr*, *tmxr*, *tmnr*, *stpr*, *icr*, *ifr*, *itr*, *idr*, *xcapr*)

Bases: object

This data structure contains Two-Terminal DC Line parameters for the Rectifier (second line)

Parameters

- **ipr** (*int*) – bus number
- **nbr** (*int*) – number of bridges
- **anmxr** (*float*) – nominal max firing angle
- **anmnr** (*float*) – minimum steady-state firing angle
- **rcr** (*float*) – transformer resistance per bridge
- **xcr** (*float*) – transformer reactance per bridge
- **ebasr** (*float*) – primary base ac voltage
- **trr** (*float*) – transformer ratio
- **tapr** (*float*) – tap setting
- **tmxr** (*float*) – max tap setting
- **tmnr** (*float*) – min tap setting
- **stpr** (*float*) – tap step
- **icr** (*int*) – firing angle measuring bus number
- **ifr** (*int*) – winding 1 side from bus number
- **itr** (*int*) – winding 2 side to bus number
- **idr** (*str*) – circuit identifier
- **xcapr** (*float*) – capacitor resistance per bridge

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**TwoWindingTransformer**(*index*, *p1*, *p2*, *w1*, *w2*)
Bases: object

This data structure contains two winding transformer parameters.

Parameters

- **index** (*int*) – unique transformer identifier
- **p1** (TransformerParametersFirstLine) – first line of parameters
- **p2** (TransformerParametersSecondLineShort) – second line of parameters
- **w1** (TransformerWinding) – first winding data
- **w2** (TransformerWindingShort) – second winding data

is_three_winding()

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**VSCDCLine**(*index*, *params*, *c1*, *c2*)
Bases: object

This data structure contains VSC DC Line parameters.

Parameters

- **index** (*int*) – unique vsc dc line identifier
- **params** (VSCDCLineParameters) – first line of VSC DC line data entry
- **c1** (VSCDCLineConverter) – second line of VSC DC Line data entry (converter 1)
- **c2** (VSCDCLineConverter) – third line of VSC DC Line data entry (converter 2)

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**VSCDCLineConverter**(*ibus*, *type*, *mode*, *dcset*, *acset*, *aloss*, *bloss*, *minloss*, *smax*, *imax*, *pwf*, *maxq*, *minq*, *remot='0'*, *rmpct='100.0'*)
Bases: object

This data structure contains VSC DC Line Converter parameters

Parameters

- **ibus** (*int*) – bus number
- **type** (*int*) – type of converter dc control
- **mode** (*int*) – ac control mode
- **dcset** (*float*) – dc setpoint

- **acset** (*float*) – ac setpoint
- **aloss** (*float*) – coefficient to linear equation for converter losses
- **bloss** (*float*) – coefficient to linear equation for converter losses
- **minloss** (*float*) – minimum losses
- **smax** (*float*) – MVA rating
- **imax** (*float*) – ac current rating
- **pwf** (*float*) – power weighting factor
- **maxq** (*float*) – reactive power upper limit
- **minq** (*float*) – reactive power lower limit
- **remot** (*int*) – bus number of remote regulating bus
- **rmpct** (*float*) – percent of Mvar required to hold bus voltage

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**VSCDCLineParameters**(*name*, *mdc*, *rdc*, *o1='0'*, *f1='1'*, *o2='0'*, *f2='1'*, *o3='0'*, *f3='1'*, *o4='0'*, *f4='1'*)
Bases: object

This data structure contains VSC DC line first-line parameters

Parameters

- **name** (*str*) – name of VSC DC Line
- **mdc** (*int*) – control mode (status)
- **rdc** (*float*) – dc line resistance
- **oi** (*int*) – owner number
- **fi** (*float*) – fraction of total ownership assigned to oi

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

class grg_pssedata.struct.**Zone**(*i*, *zoname*)
Bases: object

This data structure contains zone parameters.

Parameters

- **i** (*int*) – the identifier of the zone
- **zoname** (*string*) – zone name

to_psse()
Returns: a pss/e encoding of this data structure as a string

validate()
Checks that this data structure conforms to the pss/e data specification

grg_pssedata.struct.**quote_string**(*s*)
adds PSSE single quotes to a string

grg_pssedata.struct.**unquote_string**(*s*)
strips single quotes from a PSSE string

2.5 Module contents

a package for reading and writing of pss/e data files

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

grg_pssedata, 25
grg_pssedata.cmd, 4
grg_pssedata.exception, 5
grg_pssedata.io, 3
grg_pssedata.struct, 5

Index

A

Area (*class in grg_pssedata.struct*), 5

B

Branch (*class in grg_pssedata.struct*), 5
build_cli_parser() (*in module grg_pssedata.io*), 3
build_cmd_parser() (*in module grg_pssedata.cmd*), 4
Bus (*class in grg_pssedata.struct*), 6

C

Case (*class in grg_pssedata.struct*), 7
compare_component_lists() (*in module grg_pssedata.cmd*), 4

D

diff() (*in module grg_pssedata.cmd*), 4

E

eq() (*in module grg_pssedata.cmd*), 4
expand_commas() (*in module grg_pssedata.io*), 3

F

FACTSDevice (*class in grg_pssedata.struct*), 8
FixedShunt (*class in grg_pssedata.struct*), 9

G

Generator (*class in grg_pssedata.struct*), 9
grg_pssedata (*module*), 25
grg_pssedata.cmd (*module*), 4
grg_pssedata.exception (*module*), 5
grg_pssedata.io (*module*), 3
grg_pssedata.struct (*module*), 5

I

InductionMachine (*class in grg_pssedata.struct*), 10
InterareaTransfer (*class in grg_pssedata.struct*), 12
is_breaker() (*grg_pssedata.struct.Branch method*), 6
is_from_metered() (*grg_pssedata.struct.Branch method*), 6
is_switch() (*grg_pssedata.struct.Branch method*), 6
is_three_winding() (*grg_pssedata.struct.ThreeWindingTransformer method*), 17
is_three_winding() (*grg_pssedata.struct.TwoWindingTransformer method*), 23

L

line_index (*grg_pssedata.io.LineRequirements attribute*), 3
LineRequirements (*class in grg_pssedata.io*), 3
Load (*class in grg_pssedata.struct*), 12

M

main() (*in module grg_pssedata.cmd*), 4
main() (*in module grg_pssedata.io*), 3
max_values (*grg_pssedata.io.LineRequirements attribute*), 3
min_values (*grg_pssedata.io.LineRequirements attribute*), 3
MultiSectionLineGrouping (*class in grg_pssedata.struct*), 13
MultiTerminalDCLine (*class in grg_pssedata.struct*), 13
MultiTerminalDCLineConverter (*class in grg_pssedata.struct*), 13
MultiTerminalDCLineDCBus (*class in grg_pssedata.struct*), 14
MultiTerminalDCLineDCLink (*class in grg_pssedata.struct*), 14
MultiTerminalDCLineParameters (*class in grg_pssedata.struct*), 15

O

Owner (*class in grg_pssedata.struct*), 15

P

parse_line() (*in module grg_pssedata.io*), 3
parse_psse_case_file() (*in module grg_pssedata.io*), 3
parse_psse_case_lines() (*in module grg_pssedata.io*), 3
parse_psse_case_str() (*in module grg_pssedata.io*), 3
print_err() (*in module grg_pssedata.io*), 4
PSSEDataException, 5
PSSEDataParsingError, 5
PSSEDataValidationError, 5
PSSEDataWarning, 5

Q

quote_string() (*in module grg_pssedata.struct*), 24

S

section (*grg_pssedata.io.LineRequirements attribute*), 3
SwitchedShunt (*class in grg_pssedata.struct*), 16

T

ThreeWindingTransformer (*class in grg_pssedata.struct*), 17
to_psse() (*grg_pssedata.struct.Area method*), 5
to_psse() (*grg_pssedata.struct.Branch method*), 6
to_psse() (*grg_pssedata.struct.Bus method*), 7
to_psse() (*grg_pssedata.struct.Case method*), 8
to_psse() (*grg_pssedata.struct.FACTSDevice method*), 9
to_psse() (*grg_pssedata.struct.FixedShunt method*), 9
to_psse() (*grg_pssedata.struct.Generator method*), 10
to_psse() (*grg_pssedata.struct.InductionMachine method*), 11
to_psse() (*grg_pssedata.struct.InterareaTransfer method*), 12
to_psse() (*grg_pssedata.struct.Load method*), 12
to_psse() (*grg_pssedata.struct.MultiSectionLineGrouping method*), 13
to_psse() (*grg_pssedata.struct.MultiTerminalDCLine method*), 13
to_psse() (*grg_pssedata.struct.MultiTerminalDCLineConverter method*), 14
to_psse() (*grg_pssedata.struct.MultiTerminalDCLineDCBus method*), 14
to_psse() (*grg_pssedata.struct.MultiTerminalDCLineDCLink method*), 15
to_psse() (*grg_pssedata.struct.MultiTerminalDCLineParameters method*), 15
to_psse() (*grg_pssedata.struct.Owner method*), 15
to_psse() (*grg_pssedata.struct.SwitchedShunt method*), 16
to_psse() (*grg_pssedata.struct.ThreeWindingTransformer method*), 17
to_psse() (*grg_pssedata.struct.TransformerImpedanceCorrection method*), 17
to_psse() (*grg_pssedata.struct.TransformerParametersFirstLine method*), 18
to_psse() (*grg_pssedata.struct.TransformerParametersSecondLine method*), 19
to_psse() (*grg_pssedata.struct.TransformerParametersSecondLineShor method*), 19
to_psse() (*grg_pssedata.struct.TransformerWinding method*), 20
to_psse() (*grg_pssedata.struct.TransformerWindingShort method*), 20
to_psse() (*grg_pssedata.struct.TwoTerminalDCLine method*), 21
to_psse() (*grg_pssedata.struct.TwoTerminalDCLineInverter method*), 21
to_psse() (*grg_pssedata.struct.TwoTerminalDCLineParameters method*), 22
to_psse() (*grg_pssedata.struct.TwoTerminalDCLineRectifier method*), 22
to_psse() (*grg_pssedata.struct.TwoWindingTransformer method*), 23
to_psse() (*grg_pssedata.struct.VSCDCLine method*), 23
to_psse() (*grg_pssedata.struct.VSCDCLineConverter method*), 24
to_psse() (*grg_pssedata.struct.VSCDCLineParameters method*), 24
to_psse() (*grg_pssedata.struct.Zone method*), 24
TransformerImpedanceCorrection (*class in grg_pssedata.struct*), 17
TransformerParametersFirstLine (*class in grg_pssedata.struct*), 17
TransformerParametersSecondLine (*class in grg_pssedata.struct*), 18
TransformerParametersSecondLineShort (*class in grg_pssedata.struct*), 19
TransformerWinding (*class in grg_pssedata.struct*), 19
TransformerWindingShort (*class in grg_pssedata.struct*), 20
TwoTerminalDCLine (*class in grg_pssedata.struct*), 20
TwoTerminalDCLineInverter (*class in grg_pssedata.struct*), 21
TwoTerminalDCLineParameters (*class in grg_pssedata.struct*), 21
TwoTerminalDCLineRectifier (*class in grg_pssedata.struct*), 22
TwoWindingTransformer (*class in*

grg_pssedata.struct), 23

U

unquote_string() (*in module grg_pssedata.struct*), 25

V

validate() (*grg_pssedata.struct.Area method*), 5
validate() (*grg_pssedata.struct.Branch method*), 6
validate() (*grg_pssedata.struct.Bus method*), 7
validate() (*grg_pssedata.struct.Case method*), 8
validate() (*grg_pssedata.struct.FACTSDevice method*), 9
validate() (*grg_pssedata.struct.FixedShunt method*), 9
validate() (*grg_pssedata.struct.Generator method*), 10
validate() (*grg_pssedata.struct.InductionMachine method*), 12
validate() (*grg_pssedata.struct.InterareaTransfer method*), 12
validate() (*grg_pssedata.struct.Load method*), 13
validate() (*grg_pssedata.struct.MultiSectionLineGrouping method*), 13
validate() (*grg_pssedata.struct.MultiTerminalDCLine method*), 13
validate() (*grg_pssedata.struct.MultiTerminalDCLineConverter method*), 14
validate() (*grg_pssedata.struct.MultiTerminalDCLineDCBus method*), 14
validate() (*grg_pssedata.struct.MultiTerminalDCLineDCLink method*), 15
validate() (*grg_pssedata.struct.MultiTerminalDCLineParameters method*), 15
validate() (*grg_pssedata.struct.Owner method*), 16
validate() (*grg_pssedata.struct.SwitchedShunt method*), 17
validate() (*grg_pssedata.struct.ThreeWindingTransformer method*), 17
validate() (*grg_pssedata.struct.TransformerImpedanceCorrection method*), 17
validate() (*grg_pssedata.struct.TransformerParametersFirstLine method*), 18
validate() (*grg_pssedata.struct.TransformerParametersSecondLine method*), 19
validate() (*grg_pssedata.struct.TransformerParametersSecondLineShort method*), 19
validate() (*grg_pssedata.struct.TransformerWinding method*), 20
validate() (*grg_pssedata.struct.TransformerWindingShort method*), 20
validate() (*grg_pssedata.struct.TwoTerminalDCLine method*), 21
validate() (*grg_pssedata.struct.TwoTerminalDCLineInverter method*), 21
validate() (*grg_pssedata.struct.TwoTerminalDCLineParameters method*), 22
validate() (*grg_pssedata.struct.TwoTerminalDCLineRectifier method*), 23
validate() (*grg_pssedata.struct.TwoWindingTransformer method*), 23
validate() (*grg_pssedata.struct.VSCDCLine method*), 23
validate() (*grg_pssedata.struct.VSCDCLineConverter method*), 24
validate() (*grg_pssedata.struct.VSCDCLineParameters method*), 24
validate() (*grg_pssedata.struct.Zone method*), 24
VSCDCLine (*class in grg_pssedata.struct*), 23
VSCDCLineConverter (*class in grg_pssedata.struct*), 23
VSCDCLineParameters (*class in grg_pssedata.struct*), 24

Z

Zone (*class in grg_pssedata.struct*), 24