# grg-psse2grg Documentation

***Release 0.0.3***

**Carleton Coffrin**

**Aug 18, 2019**

# Contents:

CHAPTER 1

# Introduction

## 1.1 Overview

grg-psse2grg is a python package for translating PSSE and GRG network data files.

The primary entry point of the library is *grg_psse2grg.io* module, which contains the methods for bi-directional translation.

## 1.2 Installation

Simply run:

```
pip install grg-psse2grg
```

## 1.3 Testing

grg-psse2grg 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_psse2grg.io <path to PSSE or GRG case file>
```

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

CHAPTER 2

# grg-psse2grg package

## 2.1 grg_psse2grg.io module

grg_psse2grg.io.**build_cli_parser**()

grg_psse2grg.io.**build_psse_case**(*grg_data*, *starting_point_map_id*, *switch_assignment_map_id*)

grg_psse2grg.io.**main**(*args*)
reads a psse or grg case files and processes them based on command line arguments.

**Args:** args: an argparse data structure

grg_psse2grg.io.**parse_grg_case_file**(*grg_file_name*)
opens the given path and parses it as json data

**Args:** grg_file_name(str): path to the a json data file

**Returns:** Dict: a dictionary case

grg_psse2grg.io.**parse_psse_case_file**(*psse_file_name*)
opens the given path and parses it as pss/e data

**Args:** psse_file_name(str): path to the a psse data file

**Returns:** Case: a grg_pssedata case

grg_psse2grg.io.**parse_psse_case_lines**(*lines*)

grg_psse2grg.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.

grg_psse2grg.io.**psse_name**(*data*, *default_name*, *name_key='name'*, *length=8*)

grg_psse2grg.io.**test_idempotent**(*input_data_file*, *name*)

## 2.2 grg_psse2grg.exception module

a collection of all grg_psse2grg exception classes

**exception** grg_psse2grg.exception.**PSSE2GRGWarning**
Bases: exceptions.Warning

root class for all PSSE2GRG Warnings

## 2.3 grg_psse2grg.struct module

extensions to data structures for encoding psse data files to support grg data encoding

**class** grg_psse2grg.struct.**Area**(*i*, *isw=0*, *pdes=0.0*, *ptol=0.0*, *arnam=''*)
Bases: grg_pssedata.struct.Area

**to_grg_area**()
Returns: a grg area data dictionary

**class** grg_psse2grg.struct.**Branch**(*index*, *i*, *j*, *ckt*, *r*, *x*, *b*, *ratea*, *rateb*, *ratec*, *gi*, *bi*, *gj*, *bj*, *st*, *met*, *len*, *o1*, *f1*, *o2*, *f2*, *o3*, *f3*, *o4*, *f4*)
Bases: grg_pssedata.struct.Branch

**get_grg_operations**(*lookup*)

**get_grg_status**()
Returns: a grg data status assignment as a dictionary

**to_grg_line**(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: a grg data line name and data as a dictionary

**class** grg_psse2grg.struct.**Bus**(*i*, *name*, *basekv*, *ide*, *area*, *zone*, *owner*, *vm*, *va*, *nvhi*, *nvlo*, *evhi*, *evlo*)
Bases: grg_pssedata.struct.Bus

**get_grg_bus_setpoint**(*lookup*)
Returns: a grg data voltage set point as a dictionary

**get_grg_status**()
Returns: a grg data status assignment as a dictionary

**to_grg_bus**(*lookup*, *omit_subtype=False*)
Returns: a grg data bus name and data as a dictionary

**class** grg_psse2grg.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: grg_pssedata.struct.Case

**to_grg**(*network_id*, *omit_subtype=False*, *skip_validation=False*)
Returns: an encoding of this data structure as a grg data dictionary

**class** grg_psse2grg.struct.**FixedShunt**(*index*, *i*, *id*, *status*, *gl*, *bl*)
Bases: grg_pssedata.struct.FixedShunt

**get_grg_status**()
Returns: a grg data status assignment as a dictionary

**to_grg_shunt**(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: a grg data shunt name and data as a dictionary

**class** grg_psse2grg.struct.**Generator**(*index*, *i*, *id*, *pg*, *qg*, *qt*, *qb*, *vs*, *ireg*, *mbase*, *zr*, *zx*, *rt*, *xt*, *gtap*, *stat*, *rmpct*, *pt*, *pb*, *o1*, *f1*, *o2*, *f2*, *o3*, *f3*, *o4*, *f4*, *wmod*, *wpf*)
Bases: grg_pssedata.struct.Generator

**get_grg_cost_model**(*lookup*, *base_mva*)
Returns: a grg data encoding of this data structure as a dictionary

**get_grg_setpoint**(*lookup*, *base_mva*)
Returns: a grg data power output set point as a dictionary

**get_grg_status**()
Returns: a grg data status assignment as a dictionary

**is_synchronous_condenser**()

**to_grg_generator**(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: a grg data gen name and data as a dictionary

**class** grg_psse2grg.struct.**Load**(*index*, *i*, *id*, *status*, *area*, *zone*, *pl*, *ql*, *ip*, *iq*, *yp*, *yq*, *owner*, *scale*, *intrpt=0*)
Bases: grg_pssedata.struct.Load

**get_grg_load_setpoint**(*lookup*, *base_mva*)

**get_grg_status**()
Returns: a grg data status assignment as a dictionary

**to_grg_load**(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: a grg data load name and data as a dictionary

**class** grg_psse2grg.struct.**Owner**(*i*, *ownname*)
Bases: grg_pssedata.struct.Owner

**to_grg_owner**()

**class** grg_psse2grg.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: grg_pssedata.struct.SwitchedShunt

**get_grg_shunt_setpoint**(*lookup*, *base_mva*)

**get_grg_status**()
Returns: a grg data status assignment as a dictionary

**to_grg_shunt**(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: a grg data shunt name and data as a dictionary

**class** grg_psse2grg.struct.**ThreeWindingTransformer**(*index*, *p1*, *p2*, *w1*, *w2*, *w3*)
Bases: grg_pssedata.struct.ThreeWindingTransformer

**get_grg_status**()
Returns: a grg data status assignment as a dictionary

**to_grg_three_winding_transformer**(*bus_lookup*, *base_mva*, *omit_subtype=False*)

**class** grg_psse2grg.struct.**TwoWindingTransformer**(*index*, *p1*, *p2*, *w1*, *w2*)
Bases: grg_pssedata.struct.TwoWindingTransformer

**get_grg_status**()
Returns: a grg data status assignment as a dictionary

**get_grg_tap_changer_setpoint**(*lookup*)

**to_grg_two_winding_transformer**(*lookup*, *base_mva*, *omit_subtype=False*)
Returns: grg transformer data as a dictionary

**class** grg_psse2grg.struct.**Zone**(*i*, *zoname*)
Bases: grg_pssedata.struct.Zone

**to_grg_zone**()
Returns: a grg zone data dictionary

## 2.4 Module contents

a package for converting psse data files to grg data files

CHAPTER 3

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## g

# Index

## S



## T



## Z