

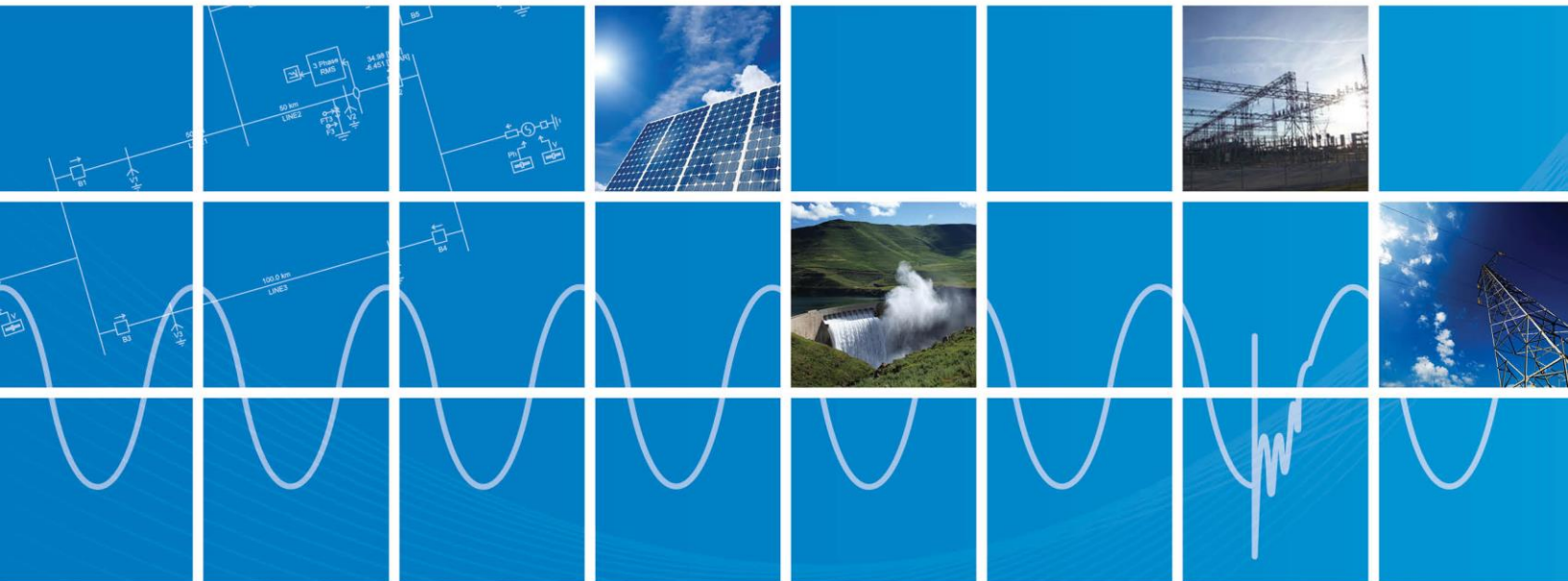


PSCAD™

## Automation Library (PSCAD v4.6.1)

August 20, 2018

Revision 1



Powered by Manitoba Hydro International Ltd.  
211 Commerce Drive  
Winnipeg, Manitoba  
R3P 1A3 Canada  
mhi.ca





# Contents

- 1. GENERAL ..... 1**
- 2. PSCAD PYTHON FUNCTION CALLS IN THE AUTOMATION LIBRARY ..... 1**
  - 2.1 BASICS.....1
  - 2.2 SYSTEM DEPENDENCIES .....1
  - 2.3 AUTOMATION CONTROLLER.....3
  - 2.4 APPLICATION CONTROLLER .....4
  - 2.5 PROJECT CONTROLLER .....5
  - 2.6 CANVAS CONTROLLER.....6
  - 2.7 COMPONENT COMMAND FUNCTIONS.....7
  - 2.8 FILE UTILITY FUNCTIONS .....8
  - 2.9 MICROSOFT WORD UTILITY FUNCTIONS.....9
  - 2.10 PYTHON RECIPES – MICROSOFT EXCEL.....10



## 1. General

This is a reference used to describe the PSCAD Python function calls that are available in the Automation Library. This document will be continuously updated as functions are added.

This document is intended for users of PSCAD v4.6.1.

## 2. PSCAD Python Function Calls in the Automation Library

### 2.1 Basics

PSCAD is a highly structure environment. By design the control and function of the software is organized in a series of access object that are organized in a hierarchy. Each object provides functionality at an increasing level of detail. To manipulate the details at lower levels, the strategy is to access individual controllers that are specifically tailored for that level of detail. Using abstraction as a natural part of the language, these controllers can be manipulated with relative ease.

This document outlines the controller function starting with the highest level types and working its way down the lowest and most detailed types. Each controller has relatively few methods so that they may be easy to use.

### 2.2 System Dependencies

#### Standard Operating System parameters and functions

This module provides a portable way of using operating system dependent functionality. (<https://docs.python.org/2/library/os.html>)

```
import os
```

#### Standard System-specific parameters and functions

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. (<https://docs.python.org/2/library/sys.html>)

```
import sys
```

#### Standard Logging parameters and functions

This module defines functions and classes which implement a flexible event logging system for applications and libraries. (<https://docs.python.org/2/library/logging.html>)

```
import logging
```

#### Define a path to the PSCAD Automation library

This will allow this script to import classes and functions from the Automation library.

```
sys.path.append(r"C:\Program Files (x86)\AutomatedTestSuite")
```



### **Import the Controller functions**

From the Automation library, this is the controller that is used to launch PSCAD.

```
import automation.controller
```

### **Import the win32com.client functions**

The modules in this package allow for dynamic usage of COM clients by Python scripts.

```
import win32com.client
```

### **Import the shutil functions**

The shutil module offers a number of high-level operations on files and collections of files. In particular, functions are provided which support file copying and removal. (<https://docs.python.org/2/library/shutil.html>)

```
import shutil
```

### **Import Dispatch function/generate the cache list of available COM commands**

The Dispatch function will allow you to open any program installed on the windows operating system that has a COM interface. Such programs include MS Excel, Word, and Outlook.

```
from win32com.client.gencache import EnsureDispatch as Dispatch
```

### **Import custom Microsoft Word utility**

The Word utility is a special collecting of functions that can be used to interact with Microsoft Word.

```
from automation.utilities.word import Word
```

### **Import custom File utility**

The File utility is a special collection of functions that can be used to easily manipulate files.

```
from automation.utilities.file import File
```

### **Import custom Mail utility**

The Mail utility allows you to send emails using Outlook or other web based emails.

```
from automation.utilities.mail import Mail
```



## 2.3 Automation Controller

### Note

Versions of installed PSCAD and FORTRAN compilers can be found in a log file:  
C:\Users\Public\Documents\Manitoba HVDC Research Centre\ATS\ProductList.xml

The Automation Controller is used to launch the application or perform other high level functions. To access the functionality get the controller object through an access method. The left hand side (LHS) reference will provide access to the controller methods.

```
controller = automation.controller.Controller()
```

Application command functions are the top level of commands that operate on the core functions. There are only a few commands in this set as it is used primarily for starting, loading and terminating the application. These commands are embedded in the automation controller module itself.

### Launch

The application can be launched using the automation controller. In this case we have instructed the application to silence all dialogue boxes. The object returned is then used to provide application control from that point forward.

```
pscad = controller.launch("PSCAD 4.6 (x64)", options={'silence': True})
```

### Arguments:

Product = product identity string  
Options = command line options



## 2.4 Application Controller

Once an application controller is established

### Quit

The application can be shut down using the quit command.

```
pscad.quit()
```

### Set the compiler

You can set the compiler to any installed FORTRAN compiler.

```
pscad.settings(cl_use_advanced='true', fortran_version='GFortran 4.6.2')
```

*Arguments: TODO*

### Load

```
pscad.load(["C:\test\project.pscx"])
```

### Access Project Controller

Get a handle to any project, for example "test".

```
project = project("test")
```

### Simulation sets

#### Run all simulation sets

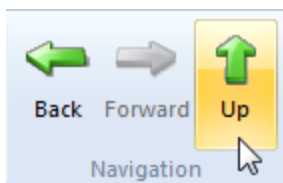
You can run all of the simulation sets by calling this one command.

```
pscad.run_all_simulation_sets()
```

### Navigation

#### Navigate up

This command will mimic the navigate up command.



```
pscad.navigate_up()
```



## 2.5 Project Controller

### Project Focus

Put project in focus, this is like selecting a project.

```
project.focus()
```

### Project Run

This command is used to run a project.

```
project.run()
```

### Layers enable/disable

You can specify a layer and enable/disable it.

```
project.set_layer('Harmonic_Impedance', 'enabled')
```

### Get Canvas

Get a handle to any canvas, for example "Main".

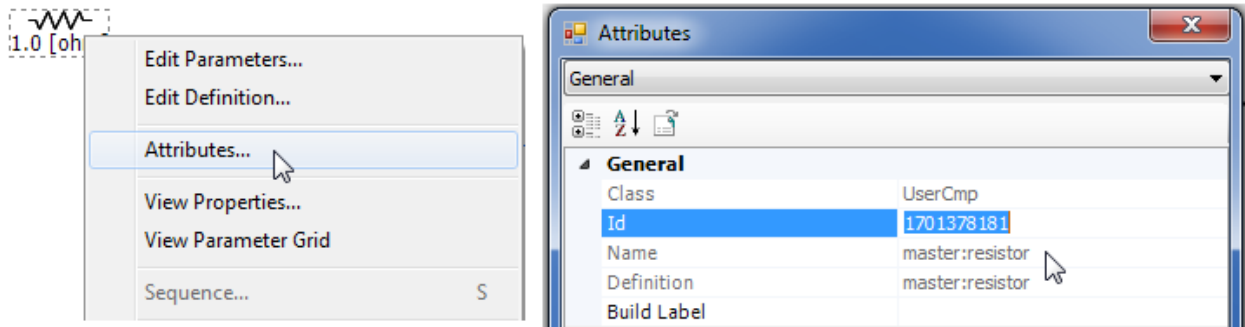
```
main
```



## 2.6 Canvas Controller

### Get Component

Get a handle to any user component by using the component ID. You get the ID of any component by right clicking and reading its attributes



```
user_cmp(1701378181)
```

### Get Transmission Line

Get a handle to any transmission line by using the component ID, exactly like getting User Components. In the example script we are getting a transmission line with the ID = 1935965525.

```
tline(1935965525)
```

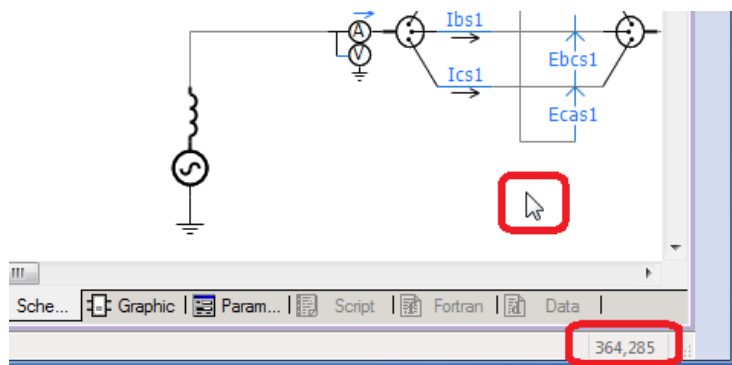
### Get Cable

Get a handle to any cable by using the component ID, exactly like getting User Components. In the example script we are getting a transmission line with the ID = 1935965525.

```
cable(1935965525)
```

### Select canvas components

Create a selection of components; this mimics a box selection using a mouse. You must first get the canvas coordinates that define a box region. The coordinates are displayed and change when you move your mouse around the canvas region.



```
select_components(x1=1425,y1=634,x2=2394,y2=1240)
```





### Copy selection as Metafile

This command tells PSCAD to copy the current selection of components as a Metafile and send the image to the Windows clipboard.

```
copy_as_metafile()
```

### Copy selection as Bitmap

This command tells PSCAD to copy the current selection of components as a Bitmap and send the image to the Windows clipboard.

```
copy_as_bitmap()
```

### Clear Selection

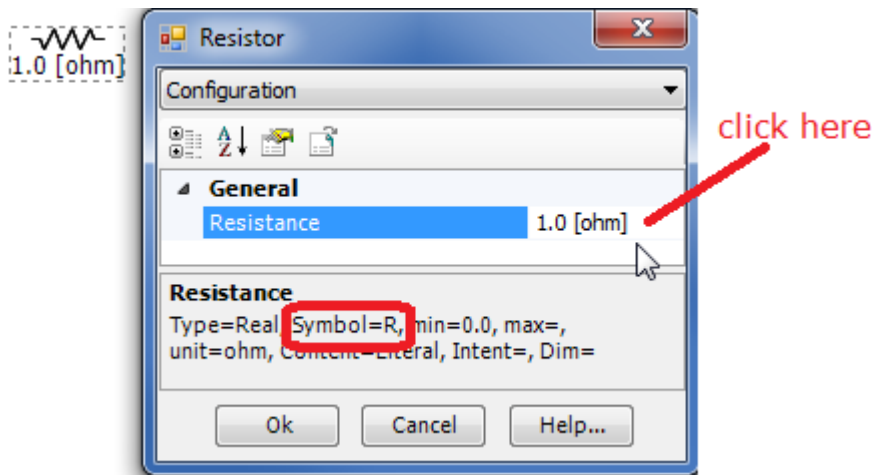
This command clears any selected components. This mimics the action of a single click on the canvas.

```
clear_selection()
```

## 2.7 Component command functions

### Set the parameters of any component

You can set the parameters of a user component by first determining what the parameter variable is. Simply open the parameters of the component and click on the field you want to change; the variable name will appear below. You can modify more than 1 parameter at the same time by separating them with a comma in the set\_parameters call.



```
set_parameters(R=0.5)
```

### Navigate into page module

This function will navigate into a page module.

```
navigate_in()
```



## 2.8 File utility functions

### New folder

Create a new folder at a specified path.

```
os.mkdir(r"C:\testing\output_folder")
```

### Move Files

Move files with specific file extensions from a source folder to a destination folder. In the example snippet, we are moving all files of type .out and .inf.

```
File.move_files(r"C:\testing\project.gf46", r"C:\testing\output_folder", ".out", ".inf")
```

### Convert files from .out to .csv

This custom function will convert a given PSCAD.out file to a comma separated variable .csv.

In the example we are taking a file called Harm.out located in some folder and creating a new csv file called Harm.csv

```
File.convert_out_to_csv(src_folder, "Harm.out", "Harm.csv")
```

## 2.9 Microsoft Word utility functions

### Open Microsoft Word

This function starts Word with a default state with no document.

```
Word()
```

### New document

This function will add a new document to Word.

```
new_document()
```

### Add text

This function will add the specified text and allow you to change font size and specify whether or not the text is bold.

```
textParagraph("some text", 20, True)
```

### Add page break

This function will add a page break, essentially starting a new page.

```
addPageBreak()
```

### Paste from Windows clipboard

This function will paste anything from the Windows Clipboard into Word.

```
pasteImage()
```



## 2.10 Python Recipes – Microsoft Excel

### Open Microsoft Excel

This function starts Excel with a default state with no sheet

```
Dispatch("Excel.Application")
```

### Make Excel visible

This function starts Excel with a default state with no sheet.

```
Visible = True
```

### Open a file

This function will load a file into Excel. A full path must be specified.

```
Workbooks.Open(r'C:\test\Harm.csv')
```

### Get a specific workbook

This function will retrieve a workbook. All workbooks are indexed as you add them. This example gets the first one.

```
Workbooks(1)
```

### Get a specific worksheet

This function will retrieve a worksheet. All worksheets are indexed as you add them. This example gets the first one.

```
Sheets(1)
```

### Get a specific column

This function will retrieve a column. All columns are indexed as you add them. This example gets the first one.

```
Columns(1)
```

### Select all rows of a column

This function will select all rows of a specific column.

```
Select()
```

### Add a chart to a specific workbook

This function will use a workbook object and add a chart to it.

```
workbook.Charts.Add()
```



### **Get a specific chart**

This function will retrieve a chart. All charts are indexed as you add them. This example gets the first one.

```
chart = workbook.Charts(1)
```

This is how you change chart types in Excel. A list of types can be found here:

<https://msdn.microsoft.com/en-us/library/office/ff837417.aspx>

```
chart.ChartType = win32com.client.constants.xlXYScatter
```

### **Activate a specific worksheet**

This is how you select or bring a specific worksheet into the main view of Excel.

```
workbook.Sheets("worksheet name").Activate()
```



## DOCUMENT TRACKING

Rev.	Description	Date
0	Initial	
1	Updated to new branding guidelines	20/Aug/2018

Copyright © 2018 Manitoba Hydro International. All Rights Reserved.