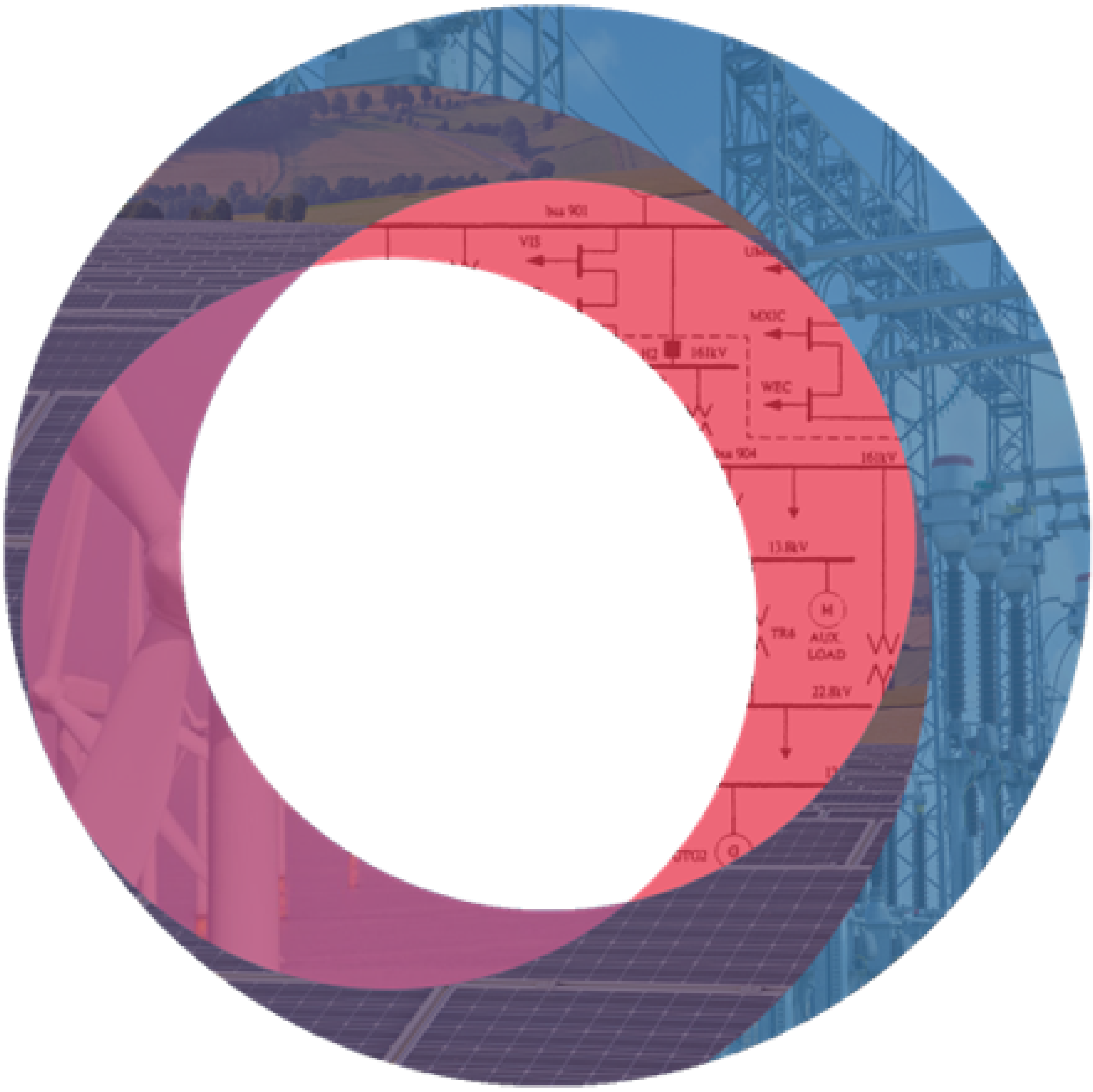
IPSA SOFTWARE

Simply Powerful.



PyIPSA Reference Manual

Version: IPSA v2.10.2

Document Reference: RG003

Date Modified: Apr 19, 2024

IPSA Power, Bainbridge House, 86-90 London Road, Manchester, M1 2PW

Phone: +44 (0)161 233 4800 | Email: support@ipsa-power.com

Contents

PyIPSA is the fastest and easiest interfacing python tool in power systems analysis!

This guide provides a full reference to all the IPSA objects and their callable functions exposed through Python. This reference guide refers to IPSA version 2.10.2. IPSA version 2.10.2 (64-bit) uses Python version 3.11.

Note the PyIPSA documentation can now be downloaded in an offline version from the Read the Docs flyout menu.

Chapter1

Current Features

The following actions are possible:

- Read and write IPSA network files
- Full access to view and/or modify all the network data - including the analysis parameters
- Create, edit and delete network components
- Add, edit and view extension data
- Perform Load flow studies and get all the results
- Perform Fault Level studies and get all the results
- Perform Harmonic Analysis and get all the results
- Draw components on the diagram

1.1 Changes from IPSA 2.10.1

1.1.1 Converter driven plant functionality

Users in PyIPSA can co-opt the *IscUMachine* object to generate converter driven plants for inverter based generator fault calculations. This uses a parametrisation conforming to ERC G74/2 (more advanced functionality available in IPSA 2.10.1 UI). There is an additional flag in the fault level settings and additional data required in *IscUMachine* for this to work. Now all the methods have been traced correctly and even allowed for advanced mode of CDP modelling. Phase corrections that prioritise reactive power injection are also included and documented.

1.1.2 Groups in PyIPSA

New functions for *IscGroup* have been added: ClearMembers, AddMember, RemoveMember, IsMember, CompareGroups, MergeGroups all allowing for more detailed and flexible modelling of groups now directly with PyIPSA. These are all documented in the *IscGroup* part of the scripting reference.

1.1.3 Component Names in PyIPSA

Component names can now be changed using the SetSValue functionality. Busbars can no longer have the same name which resolves several bugs that were appearing via PyIPSA and will now force users not to do this (as in the user interface).

1.1.4 Documentation fixes

The documentation has been fully reviewed and should be up to date – removing some non-existent functions and adding previously undocumented functions. Additionally the read-the-docs can be accessed in an off line form by downloading a pdf from the readthedocs website, which has been aesthetically updated.

1.1.5 Additional fixes

Multiple new methods added in PyIPSA for access functions (inc. CreateBranch()).

1.1.6 Changes from IPSA 2.10.0

1.1.7 Choppers in PyIPSA

The DCDC Converter object is available through the *IscChopper* class which has full support for the load flow module through PyIPSA. Several field types here have also been corrected from IPSA 2.10.1.

1.1.8 Access to database entries via PyIPSA

In PyIPSA 2.10.1, users can populate the data of their inputted components via the database finally. This is done via the member functions owned by *IscInterface*, such as *OpenDBFromFile()*, *GetDBNames()*, and *PopulateDBEntry()* via each specific network component. Users can also list the entire database entries from their loaded database.

1.1.9 Fixes to drawing functionality

The functions that *CreateBusbarCircular()*, *CreateBreaker()* and *DrawBreaker()* have been built and fixed so that users do not have to rely on the UI to program circular busbars or circuit breakers.

1.1.10 Additional fixes

The ability to access the send and receive ratings of a given branch have now been added back to the *IscBranch* object within PyIPSA. Also we have remapped the *IscTransformer::Winding* entry and the *IscTransformer::VectorGroup* entry together for longevity purposes and corrected the *IscDCMachine::MechPowerMW* bug.

1.2 Ipsa and Python

Python is a high level, general-purpose programming language. It has a simple syntax and programmes written in Python can run on many different platforms. The main features of Python include:

- **Interpreted:** Code is processed by the interpreter at runtime, saving you the task of compiling and linking it.
- **Dynamically typed:** There is no need for variable or argument declarations.
- **Object Orientated:** Supports for user-defined classes and inheritance.
- **Interactive:** Python contains an interactive prompt which is useful for testing short pieces of code.
- **Automatic Memory Management:** Python handles memory management automatically, freeing you from the need to think about allocating and freeing memory in your code.
- **Easy to use and quick to develop code:** Because it is a high-level language with an elegant syntax Python is easy to learn and the built-in data types and features such as lists and dictionaries enable quick code development.
- **Mature:** Python is a mature, stable and well-documented language.
- **Extendable:** New modules can be added in a compiled language such as C++ or C. Python programming interfaces can be incorporated into applications (e.g. IPSA).
- **Interface and Existing Toolboxes:** Many useful modules already exist that can be freely downloaded, for example, to enable interaction with Microsoft Office programmes like Excel. Toolboxes are available that allow the creation of graphical user

interfaces. Libraries like SciPy, NumPy and Matplotlib allow python to be used effectively within the scientific community.

- **Free:** Python is available under an open source license and is free to both download and include in an application.

Python is useful to us in the power industry because as the computers are advancing in power, the industry is demanding more complex, accurate and computationally intensive models. For example power systems based analysis and economic analysis based on power systems models. IPSA 2 contains application programming interfaces to Python making Python a good choice to automate analysis using power systems analysis software.

This guide provides a full reference to all the IPSA objects and their callable functions exposed through Python. This reference guide refers to IPSA version 2.10.0.

IPSA version 2.10.0 32-bit uses Python version 2.7, while the 64-bit one makes use of Python 3.8.

1.3 Coding Requirements

1.3.1 Importing IPSA

All IPSA scripts should import the IPSA interface (*IscInterface*) using the import command near the start of the script.

Starting from IPSA 2.3.2 there are two ways of launching IPSA 2, either from within IPSA 2 itself or from a separate Python process. The following code demonstrates how scripts should be written when launched from within IPSA 2. Refer to section 2.2.2 for details of running IPSA 2 from a separate Python process.

```
# Initialise Scripting interface into IPSA+
import ipsa
# Get IPSA scripting instance
ipsascript = ipsa.GetScriptInterface()

print("Welcome to IPSA")
```

It is important to ensure that there is only one *import ipsa* statement in the full extent of the code. Calling *import ipsa* multiple times in the same Python session may result in unexpected errors.

1.3.2 IPSA unique identifiers and names

IPSA assigns all components and graphical objects a unique integer number called a UID, which is used for referencing the individual component. These UIDs can be seen in the IPSA i2f files and can also be used by scripts. Component classes include functions to obtain the UID and to perform operations, such as filtering results, using them.

The component UIDs provide the best method of referring to individual components in a script. The UID of an individual component will never change during the execution of the script. Since it is an integer it is also passed efficiently between different functions in the script. The component UIDs are normally obtained from functions such as *GetBusbarUID()*.

Some functions return the IPSA object itself, such as *GetBusbar()*, which are required when the script needs to read or write component data. Due to the way in which the Python IPSA interface works these objects are not guaranteed to refer to the same component. They exist only in the Python script and may be deleted or overwritten by IPSA. This typically occurs when the script calls a *Get* type function and the internal IPSA data maps are deleted and recreated. It is recommended that the objects returned by functions such as *GetBusbar()* are used immediately.

It is important to note that the UIDs are only unique across a single network. Different network files will use the same UIDs and therefore the UIDs must never be used to refer to components in multiple networks.

Some functions also accept and return Python names for IPSA network components, for example, the *GetName()* functions. These names are also unique and are in the following format:

```
Busbar1                       # busbar name
Busbar1.Load                  # radial component
Busbar1.Busbar2.Branch        # branch component
```

1.3.3 Debugging Scripts

When IPSA encounters an error in a script a traceback message is usually produced in the form shown below. This message is printed in the IPSA progress window and provides details of the error.

```
[Nov 6 12 22:53:23] Traceback(most recent call last):
File "C:/Program Files/IpsaPower/Ipsa 2.2/scripts/PyTester.py", line 18, in <module>
gens = ipsa_network.GetIndMachines()
AttributeError: 'NoneType' object has no attribute 'GetIndMachines'
```

This provides details of the line number and file name where the error was found, in this case line 18 in file PyTester.py. The error is reported as an *AttributeError*, in the example above the *ipsa_network* variable does not have a function, or attribute, called *GetIndMachines()*. More advanced debugging is also provided as described in the following section.

Debugging with an IDE

IPSA 2 scripts can be debugged using an Integrated Development Environment such as Wing© available from **wingware.com**. This allows developers to step through code line by line and examine variables as the script is run.

It is recommended that more complex scripts are developed using PyIPSA. This allows the users' script to be started from the IDE and code can then be stepped through as required.

Once the code has been debugged it can then be quickly converted to run from normal IPSA by changing the original IscInterface loading function.

1.3.4 Coding Methods

Execution Speed

Complex scripts may have long execution times and some additional functions have been provided to reduce this time. These are summarised below:

- *SetLoadPower()* - changes the MW and MVAr of a load in the analysis engine only
- *SetLoadStatus()* - changes the MW and MVAr of a load in the analysis engine only
- *SetGeneratorPower()* - changes the MW and MVAr of a generator in the analysis engine only
- *SetGeneratorStatus()* - changes the MW and MVAr of a generator in the analysis engine only
- *SetBranchStatus()* - changes the MW and MVAr of a load in the analysis engine only
- *DoLoadFlow()* - This function includes an option to perform a load flow calculation based on the data currently in the engine
- *SetEngineMessageSuppresion()* - prevents the user interface displaying analysis engine messages
- *AllowStackBarUpdates()* - prevents the user interface from redrawing the stack bar

Memory Requirements

Memory issues have been encountered when running a significant number of studies. For example, running many load flow studies on a network will slowly use up all the maximum allowable memory for the Python process, approximately 1.3Gb. This is understood to be a result of the Python garbage collection not releasing memory back to the operating system. To avoid this issue, it is possible to run scripted IPSA as a set of separate processes. Please contact **support@ipsa-power.com** for further details.

Changing Data

Most of the objects are accessed via scripting, such as components, diagrams, analysis functions etc, have an associated set of data fields which the script can get and set, for example the nominal busbar voltage, the branch status or the load flow convergence tolerance. The majority of operations with components require the use of field values to access various data fields. There are four data types in common use, integers, strings, boolean variables and float numbers. There are therefore four functions to set and four functions to get these different data types from a component. Note that some functions may use lists of these types. The general get and set functions are as follows:

Get Functions	Set Functions	Python Data Type
GetBValue	*SetBValue*	Boolean
GetDValue	*SetDValue*	Float
GetIValue	*SetIValue*	Integer
GetSValue	*SetSValue*	String

Field indexes must be used to get and set specific items for a component. These indexes are defined for each component class and listed in the relevant sections. Field indexes are usually required in the following format, separated by dots:

- Starting with the IPSA module name
- Followed by the class name
- Ending with the field name

The following example illustrates this:

```
SetDValue(ipsa.IscBusbar.NomVoltkV, 33.0)   # Set the nominal busbar voltage
                                            # to 33kV
GetDValue(ipsa.IscBusbar.NomVoltkV)         # Get the nominal bus voltage
```

The sample code below provides some simple examples.

```python
# Initialise Scripting interface into IPSA 2
import ipsa
ipsascript = ipsa.IscInterface()

# load or create a new network
ipsascript.ReadFile('Refinery.i2f')
# return an IscNetwork instance representing the new network
ipsa_network = ipsascript.GetNetwork()

# Set data example
busbar = ipsa_network.GetBusbar('SUB 2')
# set the bus voltage
busbar.SetDValue(ipsa.IscBusbar.NomVoltkV, 11.0)

# get the nominal voltage at SUB 2
dSub2Voltage = busbar.GetDValue(ipsa.IscBusbar.NomVoltkV)
print("The voltage at SUB 2 is", dSub2Voltage, "kV")
```

Adding and Editing Components

In order to achieve optimum efficiency in terms of speed and memory usage, there are some simple recommendations regarding the execution order of statements. A common example is creating multiple components and editing the associated data. Due to the way IPSA refreshes its internal data the most efficient way to achieve this is to create all the new components first and then set the data.

IPSA creates internal data maps to store the component data accessed via scripting. These data maps must be rebuilt after components are added or deleted from the network. Changing component data does not require these maps to be rebuilt, but IPSA will automatically rebuild the maps if components have been added or deleted.

Therefore the most efficient way to add and edit components is to add all components first, then edit the component data. This will ensure that the data maps are only rebuilt once when a component is accessed to change its data. The *Get* functions have a *bFetchFromSystem* flag, setting this to *True* will force IPSA to rebuild its internal maps. Setting it to *False* will prevent these maps from being rebuilt unless required, i.e. they may still be rebuilt if components have been added or deleted.

For clarity no error checking is included in this example. For robust code, it is recommended that the return values of the various functions are checked to confirm they have executed correctly. For example, if IPSA fails to create one of the busbars then the following calls to set the voltages for that busbar will fail.

```python
# Initialise Scripting interface into IPSA
import ipsa

```

(continues on next page)

(continued from previous page)

```python
# create a new network
ipsascript = ipsa.IscInterface()
ipsascript.CreateNewNetwork(100.0, 50.0, True, True, 1.0, 1)

# return an IscNetwork instance representing the new network
ipsa_network = ipsascript.GetNetwork()

# list of busbars and associated voltages to create
busbar_list = ["Grid", "Substation", "Primary", "Secondary", "Customer"]
busbar_voltages = [132.0, 33.0, 11.0, 11.0, 0.415]
# create an empty list to store bus UIDs in
busbar_uids = []

# create all busbar objects and save UIDs
for bus in busbar_list:
    uid = ipsa_network.CreateBusbar(bus)
    busbar_uids.append(uid)

# add busbar voltages, need to access busbars using UIDs
for index in range(len(busbar_uids)):
    busbar = ipsa_network.GetBusbar(busbar_uids[index])
    busbar.SetDValue(ipsa.IscBusbar.NomVoltkV, busbar_voltages[index])
```

Setting Analysis Engine Data

Virtually all the functions presented in this manual operate on the main IPSA data model and therefore any changes can be saved within the network. There are a few functions which do not affect the main IPSA data model but change the data loaded into the calculation engine instead. These changes do not get reflected in the saved network or the network that a user would see in the User Interface. These functions allow simple changes to be made to improve calculation speed when undertaking large numbers of studies. For additional details see the IscAnalysis classes.

1.4 IscInterface

The *IscInterface* class is the main interface class used to access all other IPSA objects and functions. It **must** be created before any other references to IPSA objects. To create an instance from Python the following commands are required when running IPSA with the User Interface:

```python
# Run inside the IPSA User Interface
import ipsa
ipsascript = ipsa.GetScriptInterface()
```

The *GetScriptInterface()* returns an *IscInterface* instance which can then be used to access all other IPSA objects. The following sections provide the syntax for all other *IscInterface* functions.

Alternatively, the following code returns the *IscInterface* object when IPSA is running without a User Interface. In IPSA, the *GetInterface()* function should work as a conduit between both functions:

```
# Run with No User Interface
import ipsa
ipsascript = ipsa.IscInterface()
```

The functions *IscInterface* and *GetScriptInterface* must only be called once for each running process. Unexpected errors will occur if multiple calls to the above functions are made!

1.4.1 Debugging Options

To aid the development of scripted applications a number of debugging functions have been provided. These functions allow logging and timing of the analysis routines by providing detailed information on the analysis settings and data loaded into the analysis engines. The example below shows the output generated from a *DoLoadFlow()* function on a small test network.

```
IlfSetParameters: (100, 100, 0.01, 1,250,250,250,250,0, 0)
IlfSetRunOpts: (0, 1, 1, 1, 1, 0, 1)
IlfAddBusbarWithName: ([1]: <b>Busbar1</b> 0, 1, 0)
IlfAddBusbarWithName: ([2]: <b>Busbar2</b> 0, 1, 2e-005)
IlfAddUniversalMachine: (2, 0, 2, 0, 0)
IlfAddGridInfeed: (1, 0, 1, -2, 4.00037e-005, 0.1, 0.1, 0.1, 0, 0, 0, 0)
IlfAddBranch: (1, 2, 3, 0.0001, 0.001, 0)
IlfSetSlkBus: (1, 1)
IlfDoCalc: (4)
IlfGetBusResults: (1, 1, 0, -0.000529898, -0.00267593)
IlfGetBusResults: (2, 1, 1.99973e-005, 0.000265069, 0)
IlfGetGridInfeedResults: (1, -2.00026, -0.00263594)
IlfGetUMachResults: (1, 2, 0)
IlfGetLineResults: (1, -1.99973, 3.99892e-005, -1.99973, -0)
```

1.4.2 Database Functionality

Starting with version 2.10.1, the database functionality is now accessible within PyIPSA. The user simply has to open a database and populate an item with a database entry using the string as a reference. There is even added functionality to support item names returned to the user as well.

1.4.3 IscInterface Class

class** ipsa.**IscInterface

The main interface class used to access all other IPSA objects and functions.

***ReadFile**(strName: **str**)*

Opens an IPSA i2f file strName and returns an IscNetwork instance for that file.

Parameters

strName (*str*) – The IPSA i2f file that is going to be opened.

Returns

The IscNetwork instance for the strName file

Return type

IscNetwork

***ReadIpsa1File**(strName: **str**)*

Imports an IPSA 1 (*.iif) file strName and returns an IscNetwork instance for that file.

Parameters

strName (*str*) – The IPSA i2f file that is going to be imported.

Returns

The IscNetwork instance for the strName file

Return type

IscNetwork

GetNetwork()

Returns an IscNetwork instance for the current IPSA network.

Returns

The IscNetwork instance of the IPSA network.

Return type

IscNetwork

CloseNetwork() → **bool**

Closes the current network. Returns False if the network can't be closed, e.g. if there is unsaved data.

Returns

Boolean denoting whether the network is closed.

Return type

bool

GetDiagram(*network*, *strName: str*)

Returns an IscDiagram instance for the diagram with name strName contained in the network referred to by iscNetwork.

Parameters

- **network** (*IscNetwork*) – The IscNetwork instance of the IPSA network.
- **strName** (*str*) – The name of the diagram.

Returns

The diagram of the IPSA network.

Return type

IscDiagram

CreateNewNetwork(*dSystemBaseMVA: float*, *dFrequencyHz: float*, *bWithDiagram: bool*, *bIsDiagramSingleLine: bool*, *dGeoSceneScale: float*, *nSceneMeasurementUnit: int*) → bool

Creates a new IPSA network based on the supplied parameters. Returns False if the network can't be created.

Parameters

- **dSystemBaseMVA** (*float*) – The network base MVA.
- **dFrequencyHz** (*float*) – The nominal network frequency in hertz.
- **bWithDiagram** (*bool*) – Denoting whether the diagram is required.
- **bIsDiagramSingleLine** (*bool*) – True if a normal single line diagram type is required, False if the diagram is a scaled geographic diagram.
- **dGeoSceneScale** (*float*) – The scaling factor used to locate or size network components on geographic diagrams.
- **nSceneMeasurementUnit** (*int*) – The unit used for the geographic scale.
  - 0 if Millimetres
  - 1 if Centimetres
  - 2 if Metres
  - 3 if Kilometres

- 4 if Inches
- 5 if Feet
- 6 if Yards
- 7 if Miles

Returns
Boolean denoting whether a network can be created.

Return type
bool

*MergeFile(sMergeName: **str**)* → **bool**

Merges the IPSA I2F file sMergeName into the current network.

Parameters
sMergeName (***str***) – The name of the file being merged.

Returns
Returns True if successful, False on merge failure.

Return type
bool

*ValidatedMergeFile(sMergeName: **str**)* → **bool**

Performs a consistency check to determine if the IPSA I2F file sMergeName can be merged into the current network. Use the GetFilingErrors() function to get details of the merge errors.

Parameters
sMergeName (***str***) – The name of the file being merged.

Returns
True if successful, False on merge failure.

Return type
bool

GetFilingMessages() → **List**[**str**]

Returns a list of strings detailing the successful merge operations that occurred as a result of the ValidatedMergeFile function.

Returns
List of successful merge operations.

Return type
list(**str**)

GetFilingErrors() → **List**[**str**]

Returns a list of strings detailing the failed merge operations that occurred as a result of the ValidatedMergeFile function.

Returns
List of failed merge operations.

Return type
list(**str**)

WriteFile(*strName: **str***) → **bool**

Saves the IscNetwork instance as a new IPSA i2f network file with the file name strName. The file is saved in the current working directory unless the path is defined in the file name. The file name should include the .i2f extension

Parameters
strName (***str***) – The name of the output file containing the i2f network.

Returns
True if successful.

Return type
bool

WriteArea(*nAreaUID: **int**, strName: **str***) → **bool**

Saves the area group specified by the UID, nAreaUID, as a new IPSA i2f network file with the file name strName. The integer nAreaUID can be obtained using the IscGroup functions. The file is saved in the current working directory unless the path is defined in the file name. The file name should include the .i2f extension

Parameters

- **nAreaUID** (***int***) – The area group UID.
- **strName** (***str***) – The name of the output file containing the i2f network.

Returns
True if successful.

Return type
bool

GetAllDiagrams(*network*)

Returns a tuple of IscDiagram instances for the network referred to by IscNetwork.

Parameters
network (*IscNetwork*) – The IPSA network.

Returns
The network diagram.

Return type

tuple(*IscDiagram*)

GetAllDiagramsNames(*network*) → **List**[**str**]

Returns a list of all the diagram names for the network referred to by IscNetwork.

Parameters

network (*IscNetwork*) – The IPSA network.

Returns

List of diagram names.

Return type

list(**str**)

PrintPDF(*diagram*, *strFileName*) → **None**

Print the IscDiagram instance to a PDF format file with name strFileName.

Parameters

- **diagram** (*IscDiagram*) – The diagram of the IPSA network.
- **strFileName** (***str***) – The name of the pdf file.

MessageBox(*strDialogTitle:* ***str***, *strMessage:* ***str***) → **bool**

Display a message box with title specified by strDialogTitle and a message specified by strMessage. An OK button is provided for the user to dismiss the dialog.

Parameters

- **strDialogTitle** (***str***) – The title of the message box.
- **strMessage** (***str***) – The message displayed on the message box.

Returns

Boolean denoting whether a message box is created.

Return type

bool

AskQuestion(*strDialogTitle:* ***str***, *strQuestion:* ***str***) → **bool**

Display a message box with a title and a question as shown below.

Parameters

- **strDialogTitle** (***str***) – The title of the message box.
- **strQuestion** (***str***) – The question displayed on the message box.

Returns

True when the user clicks Yes, otherwise False.

Return type
bool

AllowStackBarUpdates(*bAllow*: ***bool***) → **None**

Setting bAllow to True prevents the IPSA stack bar from updating during script execution. This can provide speed improvements since redrawing the stack bar is prevented.

Parameters
bAllow (***bool***) – Deciding whether the IPSA stack bar can be updated during script execution.

GetDate() → **str**

Returns the date and time that IPSA was launched, e.g. 06 Nov 2012 22:53:17.

Returns
The date in a string format.

Return type
str

GetUser() → **str**

Returns the name of the current logged on user.

Returns
The name of the current logged on user.

Return type
str

GetHost() → **str**

Returns the host name of the PC.

Returns
The host name of the PC.

Return type
str

GetOrganisation() → **str**

Returns the company organisation data as set in network properties.

Returns
The company organisation data.

Return type
str

GetNetworkTitle() → **str**

Returns the network title as set in network properties.

Returns
The network title.

Return type
str

GetNetworkFileName() → **str**

Returns the filename of the current network.

Returns
The filename of the current network.

Return type
str

GetFileName(*strDialogTitle:* ***str***, *strFileTypes:* ***str***) → **str**

Display the operating system File Open dialog to prompt the user to select a file.

Parameters

- **strDialogTitle** (***str***) – The title of the dialog itself.
- **strFileTypes** (***str***) – The file type filter.

Returns
String containing the file name and path selected by the user.

Return type
str

GetDirectoryName(*strDialogTitle:* ***str***) → **str**

Display the operating system Folder Selection dialog to prompt the user to select a folder.

Parameters
strDialogTitle (***str***) – The title of the dialog itself.

Returns
String containing the path selected by the user.

Return type
str

GetVersion() → **str**

Returns the version number of IPSA software.

Returns
The version number.

Return type
str

HasLoadFlow() → **bool**

Returns True if a load flow license is present.

Returns

Boolean denoting whether a load flow license is presented.

Return type

bool

HasFaultLevel() → **bool**

Returns True if a fault level license is present.

Returns

Boolean denoting whether a fault level license is presented.

Return type

bool

HasTransient() → **bool**

Returns True if a transient stability license is present.

Returns

Boolean denoting whether a transient stability license is presented.

Return type

bool

HasProtection() → **bool**

Returns True if a protection analysis license is present.

Returns

Boolean denoting whether a protection analysis license is presented.

Return type

bool

HasHarmonics() → **bool**

Returns True if a harmonics analysis license is present.

Returns

Boolean denoting whether a harmonics analysis license is presented.

Return type

bool

HasUDM() → **bool**

Returns True if a UDM (User Defined Modelling) license is present.

Returns

Boolean denoting whether a UDM license is presented.

Return type
bool

HasDC() → **bool**

Returns True if a DC component license is present.

Returns
Boolean denoting whether a DC component license is presented.

Return type
bool

HasStaticCon() → **bool**

Returns True if a static converter license is present.

Returns
Boolean denoting whether a static converter license is presented.

Return type
bool

HasTandemGen() → **bool**

Returns True if a tandem generator license is present.

Returns
Boolean denoting whether a tandem generator license is presented.

Return type
bool

HasNonLinDevs() → **bool**

Returns True if a non-linear devices license is present.

Returns
Boolean denoting whether a non-linear devices license is presented.

Return type
bool

HasAutomation() → **bool**

Returns True if an automation analysis license is present.

Returns
Boolean denoting whether an automation analysis license is presented.

Return type
bool

IsLimitedSize() → **bool**

Returns True if the current license imposes a limit on the maximum number of busbars.

Returns

Boolean denoting whether the current license imposes a limit on the maximum number of busbars.

Return type

bool

GetMaxBusbars() → **int**

Returns the maximum number of busbars if it is a limited busbar version, returns 0 if unlimited.

Returns

The maximum number of busbars if in limited busbar version, else 0.

Return type

int

DisplayResultsTable*(nTableType: **int**)* → **None**

Displays the IPSA results table which will contain the results of the last analysis.

Parameters

nTableType – Specify the type of table displayed:

- ipsa.IscInterface.BusbarLF = busbar load flow results
- ipsa.IscInterface.GeneratorLF = generator load flow results
- ipsa.IscInterface.GridInfeedLF = grid infeed load flow results
- ipsa.IscInterface.LoadLF = load object load flow results
- ipsa.IscInterface.IMachineLF = motor load flow results
- ipsa.IscInterface.StaticVCLF = SVC load flow results
- ipsa.IscInterface.MechSwCapLF = switched capacitor load flow results
- ipsa.IscInterface.UMachineLF = universal machine load flow results
- ipsa.IscInterface.FilterLF = harmonic filter load flow results
- ipsa.IscInterface.LineLF = branch load flow results
- ipsa.IscInterface.TransformerLF = transformer load flow results
- ipsa.IscInterface.ThreeWindingTransformerLF = 3 winding transformer load flow results
- ipsa.IscInterface.BatteryLF = DC battery load flow results
- ipsa.IscInterface.DCMachineLF = DC machine load flow results
- ipsa.IscInterface.ConverterLF = AC-DC converter load flow results
- ipsa.IscInterface.ChopperLF = DC-DC converter load flow results

- ipsa.IscInterface.MGSetLF = motor-generator set load flow results
- ipsa.IscInterface.BusbarFL = busbar fault level results
- ipsa.IscInterface.GeneratorFL = generator fault level results
- ipsa.IscInterface.GridInfeedFL = grid infeed fault level results
- ipsa.IscInterface.LoadFL = load object fault level results
- ipsa.IscInterface.IMachineFL = motor fault level results
- ipsa.IscInterface.LineFL = branch fault level results
- ipsa.IscInterface.TransformerFL = transformer fault level results
- ipsa.IscInterface.ThreeWindingTransformerFL = 3 winding transformer fault level results
- ipsa.IscInterface.UniversalMachineFL = universal machine fault level results
- ipsa.IscInterface.BusbarHM = busbar harmonic analysis results
- ipsa.IscInterface.GeneratorHM = generator harmonic analysis results
- ipsa.IscInterface.LoadHM = load object harmonic analysis results
- ipsa.IscInterface.IMachineHM = motor harmonic analysis results
- ipsa.IscInterface.FilterHM = filter harmonic analysis results
- ipsa.IscInterface.LineHM = branch harmonic analysis results
- ipsa.IscInterface.TransformerHM = transformer harmonic analysis results
- ipsa.IscInterface.ThreeWindingTransformerHM = 3 winding transformer harmonic analysis results

Type
int

GetResultsTableText(*nTableType: int*) → **str**

Returns the data contained in the results' table as a comma delimited string which can be pasted directly into a spreadsheet.

Parameters
nTableType (*int*) – The type defined for the DisplayResultsTable function.

Returns
Data contained in the results' table.

Return type
str

CloseResultsTable(nTableType: int) → **None**

Closes the results' table nTableType which is as defined for the DisplayResultsTable function.

Parameters
nTableType (*int*) – The type defined for the DisplayResultsTable function.

GetLogFileName() → **str**

Get the name of log file.

Returns
The name of the log file.

Return type
str

DbgSetLogFileName(strName: str) → **None**

Set the name of the load flow log file to strName. If no file path is specified then the file is created in the IPSA bin directory.

Parameters
strName (*str*) – The name of the load flow log file.

IsLogging() → **bool**

Checks whether a logging is in progress.

Returns
Returns True if logging is in progress.

Return type
bool

DbgStartLogging() → **None**

Start logging of all analysis engine calls.

DbgStopLogging() → **None**

Stop logging of all analysis engine calls.

OpenDBFromFile(strFilename: str) → **bool**

Opens the database from file.

Parameters
strFilename (*str*) – The path name of the file to be opened.

Returns
Returns True if the database is opened successfully.

Return type
bool

CloseDBFromFile(*strFilename: **str***) → **bool**

Closes the specified database file.

Parameters
strFilename (***str***) – The path name of the file to be closed.

Returns
Returns True if the database is closed successfully.

Return type
bool

CloseAllDB() → **bool**

Close all the databases.

Returns
Returns True if databases are closed.

Return type
bool

GetDBNames() → **List**[**str**]

Returns all filenames of the databases that have been loaded.

Returns
Returns list of the databases' filenames.

Return type
list(**str**)

GetDBBranchNames(*strFilename: **str***) → **List**[**str**]

Returns all branch names in a database.

Parameters
strFilename (***str***) – The path name of the database.

Returns
Returns list of the branch names.

Return type
list(**str**)

GetDBTransformerNames(*strFilename: **str***) → **List**[**str**]

Returns all transformer names in a database.

Parameters
strFilename (***str***) – The path name of the database.

Returns
Returns list of the transformer names.

Return type
list(str)

GetDBGeneratorNames(*strFilename: str*) → List[str]

Returns all generator names in a database.

Parameters
strFilename (*str*) – The path name of the database.

Returns
Returns list of the generator names.

Return type
list(str)

GetDBIndMachineNames(*strFilename: str*) → List[str]

Returns all induction machine names in a database.

Parameters
strFilename (*str*) – The path name of the database.

Returns
Returns list of the induction machine names.

Return type
list(str)

GetDBCircuitBreakerNames(*strFilename: str*) → List[str]

Returns all circuit breaker names in a database.

Parameters
strFilename (*str*) – The path name of the database.

Returns
Returns list of the circuit breaker names.

Return type
list(str)

1.5 IscDiagram

class** ipsa.**IscDiagram

The *IscDiagram* class provides access to graphical data on a single IPSA diagram. These functions allow network components to be drawn, display options to be set and deleted.

The creation of items on the diagram also creates the associated network components. The parameters of these components can then be set using the functions described for the particular component types.

The origin for the co-ordinates is normally the top left corner of the diagram. Positive values of X are to the right whilst positive values of Y are down below the origin.

GetName() → **str**

Returns the name of the diagram.

Returns
The name of the diagram.

Return type
str

SetName(*strName: **str***) → **None**

Sets the name of the diagram.

Parameters
strName (***str***) – The name of the diagram.

CreateBusbarPoint(*strName: **str**, dX: **float**, dY: **float***) → **int**

Creates a new busbar component on the diagram. A point busbar symbol is a small dot which does not resize as the diagram zoom level is changed.

Parameters

- **strName** (***str***) – The busbar name.
- **dX** (***float***) – The busbar x coordinate.
- **dY** (***float***) – The busbar y coordinate.

Returns
The unique ID of the new busbar.

Return type
int

CreateBusbarJunction(*strName: **str**, dX: **float**, dY: **float***) → **int**

Creates a new busbar component on the diagram. A junction busbar symbol is the circular junction symbol.

Parameters

- **strName** (***str***) – The busbar name.
- **dX** (***float***) – The busbar x coordinate.
- **dY** (***float***) – The busbar y coordinate.

Returns
The unique ID of the new busbar.

Return type
int

CreateBusbarHexagonal(*strName*: ***str***, *dX*: ***float***, *dY*: ***float***) → **int**

Creates a new busbar component on the diagram. A hexagonal busbar symbol has six sides.

Parameters

- **strName** (***str***) – The busbar name.
- **dX** (***float***) – The busbar x coordinate.
- **dY** (***float***) – The busbar y coordinate.

Returns
The unique ID of the new busbar.

Return type
int

CreateBusbarCircular(*strName*: ***str***, *dX*: ***float***, *dY*: ***float***) → **int**

Creates a new busbar component on the diagram. A circular busbar symbol is a circle.

Parameters

- **strName** (***str***) – The busbar name.
- **dX** (***float***) – The busbar x coordinate.
- **dY** (***float***) – The busbar y coordinate.

Returns
The unique ID of the new busbar.

Return type
int

CreateBusbarRectangular(*strName*: ***str***, *bHorizontal*: ***bool***, *dX*: ***float***, *dY*: ***float***) → **int**

Creates a new busbar component on the diagram. The rectangular symbol is the standard horizontal or vertical busbar.

Parameters

- **strName** (***str***) – The busbar name.
- **bHorizontal** (***bool***) – True draws a horizontal rectangular busbar, while False draws a vertical busbar.
- **dX** (***float***) – The busbar x coordinate.
- **dY** (***float***) – The busbar y coordinate.

Returns
The unique ID of the new busbar.

Return type
int

***DrawBusbarPoint**(nUID: **int**, dX: **float**, dY: **float**)* → **bool**

Draws an existing busbar component on the diagram as defined by the busbar UID. A point busbar symbol is displayed as a small dot which does not resize as the diagram zoom level is changed.

Parameters

- **nUID** (*int*) – The busbar UID.
- **dX** (*float*) – The busbar x coordinate.
- **dY** (*float*) – The busbar y coordinate.

Returns
Boolean denoting whether the busbar was drawn.

Return type
bool

***DrawBusbarJunction**(nUID: **int**, dX: **float**, dY: **float**)* → **bool**

Draws an existing busbar component on the diagram as defined by the busbar UID. A junction busbar symbol is the solid circular junction symbol.

Parameters

- **nUID** (*int*) – The busbar UID.
- **dX** (*float*) – The busbar x coordinate.
- **dY** (*float*) – The busbar y coordinate.

Returns
Boolean denoting whether the busbar was drawn.

Return type
bool

***DrawBusbarHexagonal**(nUID: **int**, dX: **float**, dY: **float**)* → **bool**

Draws an existing busbar component on the diagram as defined by the busbar UID. The hexagonal symbol is the standard filled hexagonal busbar.

Parameters

- **nUID** (*int*) – The busbar UID.
- **dX** (*float*) – The busbar x coordinate.
- **dY** (*float*) – The busbar y coordinate.

Returns

Boolean denoting whether the busbar was drawn.

Return type

bool

DrawBusbarRectangular(*nUID*: ***int***, *bHorizontal*: ***bool***, *dSize*: ***float***, *dX*: ***float***, *dY*: ***float***) → **bool**

Draws an existing busbar component on the diagram as defined by the busbar UID. The rectangular symbol is the standard horizontal or vertical busbar.

Parameters

- **nUID** (*int*) – The busbar UID.
- **bHorizontal** (*bool*) – True draws a horizontal rectangular busbar, while False draws a vertical busbar.
- **dSize** (*float*) – The length of the busbar symbol.
- **dX** (*float*) – The busbar x coordinate.
- **dY** (*float*) – The busbar y coordinate.

Returns

Boolean denoting whether the busbar was drawn.

Return type

bool

DrawBusbarCircular(*nUID*: ***int***, *dSize*: ***float***, *dX*: ***float***, *dY*: ***float***) → **bool**

Draws an existing busbar component on the diagram as defined by the busbar UID. The circular symbol is the larger unfilled circle.

Parameters

- **nUID** (*int*) – The busbar UID.
- **dSize** (*float*) – The radius of the busbar symbol.
- **dX** (*float*) – The busbar x coordinate.
- **dY** (*float*) – The busbar y coordinate.

Returns

Boolean denoting whether the busbar was drawn.

Return type

bool

CreateLine(*strName*: ***str***, *dXFrom*: ***float***, *dYFrom*: ***float***, *dXTo*: ***float***, *dYTo*: ***float***) → **int**

Deprecated. Instead, use CreateBranch.

Creates a new branch component on the diagram.

Parameters

- **strName** (*str*) – The branch name.
- **dXFrom** (*float*) – The x coordinate of the busbar where the branch starts.
- **dYFrom** (*float*) – The y coordinate of the busbar where the branch starts.
- **dXTo** (*float*) – The x coordinate of the busbar where the branch ends.
- **dYTo** (*float*) – The y coordinate of the busbar where the branch ends.

Returns

The unique positive ID of the new branch. A negative value is returned if the "from" end busbar is not found, and zero is returned if the "to" end busbar is not found.

Return type

int

CreateBranch(*strName*: ***str***, *dXFrom*: ***float***, *dYFrom*: ***float***, *dXTo*: ***float***, *dYTo*: ***float***) → **int**

Creates a new branch component on the diagram.

Parameters

- **strName** (*str*) – The branch name.
- **dXFrom** (*float*) – The x coordinate of the busbar where the branch starts.
- **dYFrom** (*float*) – The y coordinate of the busbar where the branch starts.
- **dXTo** (*float*) – The x coordinate of the busbar where the branch ends.
- **dYTo** (*float*) – The y coordinate of the busbar where the branch ends.

Returns

The unique positive ID of the new branch. If the branch cannot be drawn, the return value is 0.

Return type

int

DrawLine(*nUID*: ***int***) → **bool**

Draws the symbol for the line identified by the unique ID. The line is drawn as a single segment between two busbars. The line must have been created using one of the following first:

- IscDiagram.CreateLine
- IscNetwork.CreateBranch
- IscNetwork.CreateTransformer

Parameters
nUID (*int*) – The line UID.

Returns
Boolean denoting whether the line was drawn.

Return type
bool

CreateBreaker(*strName: **str**, dX: **float**, dY: **float***) → **int**

Creates a new circuit breaker on the diagram. Note branch has to have already been drawn.

Parameters

- **strName** (*str*) – The breaker name.
- **dX** (*float*) – The x coordinate of the circuit breaker.
- **dY** (*float*) – The y coordinate of the circuit breaker.

Returns
The unique positive ID of the new circuit breaker. If the breaker cannot be drawn, the return value is 0.

Return type
int

DrawBreaker(*nBreakerUID: **int**, dX: **float**, dY: **float***) → **bool**

Draws the symbol for the breaker identified by the unique ID nBreakerUID at the location dX,dY.

Parameters

- **nBreakerUID** (*int*) – The breaker UID.
- **dX** (*float*) – The x coordinate of the circuit breaker.
- **dY** (*float*) – The y coordinate of the circuit breaker.

Returns
The function returns True if the breaker was drawn

Return type
bool

CreateTransformer(*strName*: ***str***, *dXFrom*: ***float***, *dYFrom*: ***float***, *dXTo*: ***float***, *dYTo*: ***float***) → **int**

Deprecated. Instead, use Create2WTransformer.

Creates a new transformer component on the diagram.

Parameters

- **strName** (***str***) – The branch name.
- **dXFrom** (***float***) – The x coordinate of the busbar where the branch starts.
- **dYFrom** (***float***) – The y coordinate of the busbar where the branch starts.
- **dXTo** (***float***) – The x coordinate of the busbar where the branch ends.
- **dYTo** (***float***) – The y coordinate of the busbar where the branch ends.

Returns

The unique positive ID of the new transformer. A negative value is returned if the "from" end busbar is not found, and zero is returned if the "to" end busbar is not found.

Return type

int

Create2WTransformer(*strName*: ***str***, *dXFrom*: ***float***, *dYFrom*: ***float***, *dXTo*: ***float***, *dYTo*: ***float***) → **int**

Creates a new transformer component on the diagram.

Parameters

- **strName** (***str***) – The branch name.
- **dXFrom** (***float***) – The x coordinate of the busbar where the branch starts.
- **dYFrom** (***float***) – The y coordinate of the busbar where the branch starts.
- **dXTo** (***float***) – The x coordinate of the busbar where the branch ends.
- **dYTo** (***float***) – The y coordinate of the busbar where the branch ends.

Returns

The unique positive ID of the new transformer. If the transformer cannot be drawn, the return value is 0.

Return type
int

DrawTransformer(*nUID: **int***) → **bool**

Draws the symbol for the transformer identified by the unique ID. The line is drawn as a single segment between two busbars. The line must have been created using the following first:

- IscNetwork.CreateTransformer

Parameters
nUID (***int***) – The transformer UID.

Returns
Boolean denoting whether the line was drawn.

Return type
bool

CreateUnbalancedLine(*strName: **str**, dXFrom: **float**, dYFrom: **float**, dXTo: **float**, dYTo: **float***) → **int**

Deprecated. Instead, use CreateUnbalancedBranch.

Creates a new unbalanced line component on the diagram.

Parameters

- **strName** (***str***) – The unbalanced line name.
- **dXFrom** (***float***) – The x coordinate of the busbar where the branch starts.
- **dYFrom** (***float***) – The y coordinate of the busbar where the branch starts.
- **dXTo** (***float***) – The x coordinate of the busbar where the branch ends.
- **dYTo** (***float***) – The y coordinate of the busbar where the branch ends.

Returns
The unique positive ID of the new unbalanced line component. A negative value is returned if the "from" end busbar is not found, and zero is returned if the "to" end busbar is not found.

Return type
int

CreateUnbalancedBranch(*strName: **str**, dXFrom: **float**, dYFrom: **float**, dXTo: **float**, dYTo: **float***) → **int**

Creates a new unbalanced line component on the diagram.

Parameters

- **strName** (*str*) – The unbalanced line name.
- **dXFrom** (*float*) – The x coordinate of the busbar where the branch starts.
- **dYFrom** (*float*) – The y coordinate of the busbar where the branch starts.
- **dXTo** (*float*) – The x coordinate of the busbar where the branch ends.
- **dYTo** (*float*) – The y coordinate of the busbar where the branch ends.

Returns

The unique positive ID of the new unbalanced line component. If the unbalanced line cannot be drawn, the return value is 0.

Return type

int

CreateUnbalancedTransformer(*strName:* ***str***, *dXFrom:* ***float***, *dYFrom:* ***float***, *dXTo:* ***float***, *dYTo:* ***float***) → **int**

Deprecated. Instead, use CreateUnbalanced2WTransformer.

Creates a new unbalanced transformer component on the diagram.

Parameters

- **strName** (*str*) – The unbalanced transformer name.
- **dXFrom** (*float*) – The x coordinate of the busbar where the branch starts.
- **dYFrom** (*float*) – The y coordinate of the busbar where the branch starts.
- **dXTo** (*float*) – The x coordinate of the busbar where the branch ends.
- **dYTo** (*float*) – The y coordinate of the busbar where the branch ends.

Returns

The unique positive ID of the new unbalanced transformer component. A negative value is returned if the "from" end busbar is not found, and zero is returned if the "to" end busbar is not found.

Return type
int

CreateUnbalanced2WTransformer(*strName: **str**, dXFrom: **float**, dYFrom: **float**, dXTo: **float**, dYTo: **float***) → **int**

Creates a new unbalanced transformer component on the diagram.

Parameters

- **strName** (***str***) – The unbalanced transformer name.
- **dXFrom** (***float***) – The x coordinate of the busbar where the branch starts.
- **dYFrom** (***float***) – The y coordinate of the busbar where the branch starts.
- **dXTo** (***float***) – The x coordinate of the busbar where the branch ends.
- **dYTo** (***float***) – The y coordinate of the busbar where the branch ends.

Returns
The unique positive ID of the new unbalanced transformer component. If the unbalanced transformer cannot be drawn, the return value is 0.

Return type
int

AddPointToLine(*nLineUID: **int**, dX: **float**, dY: **float**, bFromEnd: **bool***) → **bool**

Adds a knee point to the line identified by the unique ID.

Parameters

- **nLineUID** (***int***) – The line UID.
- **dX** (***float***) – The knee point x coordinate.
- **dY** (***float***) – The knee point y coordinate.
- **bFromEnd** (***float***) – If True then the knee point is added to the last segment, i.e. furthest from the From end. If False then the knee point is added to the first segment.

Returns
Boolean denoting whether the knee point was added.

Return type
bool

RefreshLine(*nLineUID: **int***) → **None**

Redraws the line identified by the line UID after knee points have been added.

Parameters

nLineUID (***int***) – The line UID.

SplitBranch(*nLineUID: **int**, nSection: **int**, dRatio: **float**, strName: **str***) → **int**

Splits a branch into two sections connected by a new busbar.

Parameters

- **nLineUID** (***int***) – The line UID.
- **nSection** (***int***) – Specifies which section of a multi-section branch is split. For branches with only one section then nSection should be set to 0.
- **dRatio** (***float***) – Specifies how the branch impedances are divided between the new branches. A value of 0.0 sets the split position to be at the "From" end whilst a value of 1.0 specifies the "To" end. Values between 0.0 and 1.0 split the branch in proportion. For multi-section branches dRatio splits the section identified by nSection.
- **strName** (***str***) – The name of the busbar.

Returns

The UID of the new branch if it is greater than 0.) if the branch has not been split. This is because there is a circuit breaker on the branch or the branch is drawn on more than one diagram.

Return type

int

DrawUndrawnItemsAttachedToBusbar(*nBusbarUID: **int***) → **None**

Draws items attached to the busbar identified by the busbar UID if they are not already drawn on the diagram. Note that this will draw branch items as well.

Parameters

nBusbarUID (***int***) – The busbar UID.

DeleteItem(*nUID: **int***) → **bool**

Deletes the graphic item identified by the UID. This may be a line, radial component or busbar.

Parameters

nUID (***int***) – The graphical item UID.

Returns

True if deletion is successful.

Return type
bool

*GetLineLength(nUID: **int**)* → **float**

GetLineLength(pComponent) → **float**

Returns the component length for the graphic symbol on the current diagram. On geographic diagrams this function returns the actual line length, assuming that the diagram is correctly scaled.

Parameters

- **nUID** (*int*) – The line UID.
- **pComponent** (*IscBranch*) – The line IscBranch instance.

Returns
The component length for the graphic symbol.

Return type
float

*SetItemPenColour(nUID: **int**, nRed: **int**, nGreen: **int**, nBlue: **int**, nAlpha: **int**)* → **bool**

Sets the outline colour of the diagram object. The colour is set by the RGB parameters. All colour parameters should be between 0 and 255.

Parameters

- **nUID** (*int*) – The diagram object UID.
- **nRed** (*int*) – The red colour.
- **nGreen** (*int*) – The green colour.
- **nBlue** (*int*) – The blue colour.
- **nAlpha** (*int*) – The transparency of the colour.

Returns
Denoting whether the colour is set.

Return type
bool

*SetItemBrushColour(nUID: **int**, nRed: **int**, nGreen: **int**, nBlue: **int**, nAlpha: **int**)* → **bool**

Sets the fill colour of the diagram object. The colour is set by the RGB parameters. All colour parameters should be between 0 and 255.

Parameters

- **nUID** (*int*) – The diagram object UID.
- **nRed** (*int*) – The red colour.
- **nGreen** (*int*) – The green colour.

- **nBlue** (*int*) – The blue colour.
- **nAlpha** (*int*) – The transparency of the colour.

Returns
Denoting whether the colour is set.

Return type
bool

MapToLatLong(*fScreenX: **float**, fScreenY: **float***) → **List**[**float**]

Returns the latitude and longitude in decimal degrees of the screen position. The diagram is the one referenced by the IscDiagram object that the function is called on. The fScreenX and fScreenY parameters are relative to the nominal centre point of the screen, therefore calling this function with fScreenX = 0.0 and fScreenY = 0.0 returns the centre point of the background map in degrees. Note that the screen X is north/south and screen y is east/west.

Parameters

- **fScreenX** (***float***) – The x coordinate of the screen position.
- **fScreenX** – The y coordinate of the screen position.

Returns
The latitude and longitude of the screen position.

Return type
list(float)

LatLongToMap(*fN: **float**, fE: **float***) → **List**[**float**]

Returns the screen X and Y coordinates of the latitude and longitude. The fScreenX and fScreenY coordinates are relative to the nominal centre point of the screen which can be found by the MapToLatLong function. Note that the screen X is north/south and screen y is east/west.

Parameters

- **fN** (***float***) – The latitude.
- **fE** (***float***) – The longitude.

Returns
The screen X and Y coordinates.

Return type
list(float)

GetUIDFromCoordinates(*dX: **float**, dY: **float***) → **int**

Returns the UID of a component at coordinates (dX, dY). The screen coordinates are relative to the nominal centre point of the screen.

Parameters

- **dX** (*float*) – The screen X coordinate.
- **dY** (*float*) – The screen Y coordinate.

Returns
The UID of the component located. Returns 0, if the component cannot be found,

Return type
int

GetBusbarUIDFromCoordinates(dX: float, dY: float) → int

Returns the UID of a busbar at coordinates (dX, dY). The screen coordinates are relative to the nominal centre point of the screen.

Parameters

- **dX** (*float*) – The screen X coordinate.
- **dY** (*float*) – The screen Y coordinate.

Returns
The UID of the component located. Returns 0, if the component cannot be found,

Return type
int

GetItemX(nUID: int) → float

Returns the screen X coordinate of the busbar. The screen coordinates are relative to the nominal centre point of the screen.

Parameters
nUID (*int*) – The busbar UID.

Returns
The screen X coordinate.

Return type
float

GetItemY(nUID: int) → float

Returns the screen Y coordinate of the busbar. The screen coordinates are relative to the nominal centre point of the screen.

Parameters
nUID (*int*) – The busbar UID.

Returns
The screen Y coordinate.

Return type
float

***GetItemFromXPoints**(nUID: **int**)* → **List**[**float**]

Returns a list of floats for the screen X coordinates of the FROM busbar point, the middle point of the line and all knee points lying on the branch between these two points. The coordinates are for the FROM end of the line. The screen coordinates are relative to the nominal centre point of the screen.

Parameters
nUID (*int*) – The line UID.

Returns
The screen X coordinates.

Return type
float

***GetItemFromYPoints**(nUID: **int**)* → **List**[**float**]

Returns a list of floats for the screen Y coordinates of the FROM busbar point, the middle point of the line and all knee points lying on the branch between these two points. The coordinates are for the FROM end of the line. The screen coordinates are relative to the nominal centre point of the screen.

Parameters
nUID (*int*) – The line UID.

Returns
The screen Y coordinates.

Return type
float

***GetItemToXPoints**(nUID: **int**)* → **List**[**float**]

Returns a list of floats for the screen X coordinates of the TO busbar point, the middle point of the line and all knee points lying on the branch between these two points. The coordinates are for the TO end of the line. The screen coordinates are relative to the nominal centre point of the screen.

Parameters
nUID (*int*) – The line UID.

Returns
The screen X coordinates.

Return type
float

***GetItemToYPoints**(nUID: **int**)* → **List**[**float**]

Returns a list of floats for the screen Y coordinates of the TO busbar point, the middle point of the line and all knee points lying on the branch between these two points. The coordinates are for the TO end of the line. The screen coordinates are relative to the nominal centre point of the screen.

Parameters
nUID (*int*) – The line UID.

Returns
The screen Y coordinates.

Return type
float

CreateAnnotation(strName: str, strAnnotationText: str, dX: float, dY: float) → **int**

Creates a new diagram annotation. The screen coordinates are relative to the nominal centre point of the screen.

Parameters

- **strName** (*str*) – The strName is not used and can be an empty string.
- **strAnnotationText** (*str*) – The text to be displayed on the diagram. The text string can include simple html for text formatting.
- **dX** (*float*) – The x coordinate of the diagram.
- **dY** (*float*) – The y coordinate of the diagram.

Returns
The diagram annotation.

Return type
int

PrintPDF(strFileName: str) → **None**

Print the diagram to a PDF file.

Parameters
nUID (*int*) – The line UID.

Returns
The screen Y coordinate.

Return type
float

1.6 IscNetwork

This object provides the main access to an IPSA network. It is generally created as the result to a call to *IscInterface().ReadFile(strName)*. This class provides functions to retrieve, create and delete network components, perform analysis and get network results.

1.6.1 Network Component Functions

Various functions are provided to allow the creation, deletion and editing of network components. The *Get*... functions return instances of the component objects, for example *GetBusbar* returns an *IscBusbar* instance. Once an *IscBusbar* instance is retrieved the busbar data can then be accessed as described in the *IscBusbar* section.

Component Access

The dictionary keys are the Python script names of components whilst the values are the instances of components. The Python script name can be used to access individual components.

The example code below details how it is possible to iterate through all the components of a particular type in a network:

```
# retrieve the busbar collection
busbars = net.GetBusbars()
# cycle each busbar, retrieve its name and voltage
for bus in busbars.values():
   # do something with the bus
    name = bus.GetName()
```

Two functions are also provided to return dictionaries of the unique component IDs. The dictionary keys are the Python script names while the dictionary values are the integer IDs.

Setting *bFetchFromSystem* to *True* forces IPSA to rebuild its internal component data maps. Setting *bFetchFromSystem* to *False* will only rebuild the internal component data maps if components have been added or deleted since the last *Get*... function call. If the script creates new data components during its execution then the internal component data maps will always be rebuilt and *bFetchFromSystem* can be *True* or *False*.

1.6.2 Component Ratings

Rating sets determine the thermal limits that branches and transformers can tolerate. Each component can be given a set of MVA or kA values which are checked after a load flow calculation to identify if the component is overloaded. In IPSA 1.x four rating sets were provided, namely Standard, Summer, Winter and Short. In IPSA 2 these rating sets are provided by default but users can add additional rating sets. The ratings sets defined by the user, either through the IPSA interface or via scripting, are stored with the network model.

The functions used to access the rating data have therefore been changed from IPSA 1.x in order to address the user-defined rating sets.

1.6.3 Profiles

Profiles represent a set of categories with associated MW and MVAr values. Profiles can be assigned to loads, synchronous machines and universal machines. Each network can have any number of profiles which can used to provide absolute or scaled MW and MVAr values. Every load, generator and universal machine in the network can be assigned one of the profiles and load flow analysis or profile analysis can then be performed. Scaling profiles cannot be assigned to universal machines.

Refer to section 0 for the function to run a profile study.

Different types of profiles are represented by different classes as follows;

- Actual load profile class - *IscLoadProfilePQActual*
- Scaled load profile class - *IscLoadProfilePQScale*
- Actual generator profile class - *IscGeneratorProfilePQActual*
- Scaled generator profile class - *IscGeneratorProfilePQScale*
- Actual universal machine profile class - *IscUMachineProfilePQActual*

Add Profile Categories

Profiles comprise a number of categories and associated MW and MVAr values. Each category is simply a string which identifies the category name. Examples of profile categories could be:

- Spring, Summer, Autumn, Winter
- Normal, Max Load, Min Load, Emergency
- 00:00hrs, 01:00hrs, 02:00hrs, 03:00hrs etc.

The category names are only for user interaction and do not relate to other network components or analysis settings such as equipment ratings.

Add Profile Data

Each profile category can be assigned a specific MW and MVAr load for the various profile types. The MW or MVAr value assigned to each category is either an actual value or a per unit scaling value depending on the profile type.

Add Profiles to Components

Once a profile has been created it can then be assigned to any number of individual loads, generators and universal machines in the network. The field index *ProfileUID* is set to assign a profile to a network load, generator or universal machine. This is detailed in the corresponding component sections and the code below illustrates the use of all the load profile functions.

```
# define the categories and loads
categories = {0:"00:00",
              1:"06:00",
              2:"12:00",
              3:"18:00"}

mw = {0: 0.8,
      1: 0.775,
      2: 0.75,
      3: 0.712}
mvar = {0: 0.48,
        1: 0.465,
        2: 0.45,
        3: 0.4272}

# create a load profile
profileUID = ipsanetwork.CreateLoadProfilePQActual('Test Profile')

# get the profile ID
profile = ipsanetwork.GetLoadProfilePQActual(profileUID)

# add the categories to the profile and set the data
profile.SetCategoryNames(categories)
profile.SetPMW(mw)
profile.SetQMVAr(mvar)

# finally assign the profile to all network loads
loads = ipsa_network.GetLoads()
```

(continues on next page)

(continued from previous page)

```
for load in loads.values():
    load.SetIValue(ipsa.IscLoad.ProfileUID)
```

Running Profile Studies

All categories in the selected profiles are run and the results are obtained from the functions provided in each component class. All profiles must have the same set of category names. The load flow solution parameters are set using the *IscAnalysisLF* class.

1.6.4 IscNetwork Class

class *ipsa.***IscNetwork**

Class providing the main access to an IPSA network.

SetBusbarSlack(*strBusbar: **str***) → **None**

Sets the busbar as the slack busbar for a particular part of the network.

Parameters

strBusbar (***str***) – The Python busbar name which is returned by IscNetComponent.GetName().

RefreshSystem() → **None**

Forces IPSA to rebuild its internal component data maps. This function can be used if the network has been modified outside of scripting while a script is running.

WriteFile(*strName: **str***) → **bool**

Saves the current network file at the path and the file name.

Parameters

strName (***str***) – The file name.

Returns

Denoting whether the file is saved.

Return type

bool

WriteArea(*nAreaUID: **int**, strName: **str***) → **bool**

Saves the area group UID as a new IPSA i2f network file. The file is saved in the current working directory. The file name should include the .i2f extension.

Parameters

- **nAreaUID** – The area group UID. nAreaUID can be obtained using the IscGroup functions.

- **strName** (*str*) – The file name.

Returns
Denoting whether the file is saved.

Return type
bool

MergeFile(*strMergeName: **str***) → **bool**

Merges the file into the current network file.

Parameters
strMergeName (*str*) – The merged file name.

Returns
Denoting whether the file is successfully saved.

Return type
bool

ValidatedMergeFile(*strMergeName: **str***) → **bool**

Performs a consistency check to determine if the IPSA I2F file can be merged into the current network. Use the GetFilingErrors() function to get details of the merge errors.

Parameters
strMergeName (*str*) – The merged file name.

Returns
Denoting whether the file is successfully saved.

Return type
bool

CommitVersion(*strVersionName: **str***) → **int**

Creates a new network version which includes all non-versioned network changes.

Parameters
strVersionName (*str*) – The new network version name.

Returns
An integer representing the version ID.

Return type
int

GetVersionUuid(*nVersion: **int***) → **str**

Returns a unique string (UUID) representing the version name for the given version.

Parameters
nVersion (*int*) – The selected version.

Returns
The version name.

Return type
str

SetToVersion(nVersion: int) → **bool**

Selects the version of the current network.

Parameters
nVersion (*int*) – The selected version.

Returns
Denoting whether the version is successfully set or whether it does not exist.

Return type
bool

CreateChangeFile(nVersion: int, strMergeName: str) → **bool**

Creates an IPSA merge file based on the network differences between the given version and the current version.

Parameters

- **nVersion** (*int*) – The selected version.
- **strMergeName** (*str*) – The merged file name.

Returns
Denoting whether the file is successfully created.

Return type
bool

GetCurrentVersion() → **int**

Returns the current working version. Any changes to the network are made to this version.

Returns
The current version.

Return type
int

GetParentVersion(nVersion: int) → **int**

Returns the parent version for the selected version.

Parameters
nVersion (*int*) – The selected version.

Returns
The parent version.

Return type
int

GetVersionDiffAdded(*nVersion: **int***) → **List**[**int**]

Returns a list of component UIDs which have been added to the network in the current selected version and that were not in the selected version.

Parameters
nVersion (***int***) – The selected version.

Returns
List of component UIDs.

Return type
int

GetVersionDiffChanged(*nVersion: **int***) → **List**[**int**]

Returns a list of component UIDs which have been edited in the current selected version compared to the selected version.

Parameters
nVersion (***int***) – The selected version.

Returns
List of component UIDs.

Return type
int

GetVersionDiffDeleted(*nVersion: **int***) → **List**[**int**]

Returns a list of component UIDs which have been deleted from the network in the current selected version and that were in the selected version.

Parameters
nVersion (***int***) – The selected version.

Returns
List of component UIDs.

Return type
int

ResetResults() → **None**

Reset all analysis results.

GetSystemBaseMVA() → **float**

Returns the current system MVA defined for the IPSA network Default: 100 MVA

Returns
Network system MVA value

Return type
float

GetNumberOfIslands() → **int**

Returns the number of islands.

Returns
The number of islands.

Return type
int

GetIslandsUIDs() → **Dict**[**str**, **List**[**int**]]

Returns a dictionary of integer busbar nUIDs belonging to the islands. The keys are the island slack busbar names or the first busbar names if no slack busbar is set for that island.

Returns
The busbars belonging to each island.

Return type
dict(**str**,**list**(**int**))

GetNoSlackIslandsUIDs() → **Dict**[**str**, **List**[**int**]]

Returns a dictionary of integer busbar UIDs belonging to islands with no slack busbars. The keys are the first busbar names.

Returns
The busbars with no slack belonging to each island.

Return type
dict(**str**,**list**(**int**))

GetNoGeneratorIslandsUIDs() → **Dict**[**str**, **List**[**int**]]

Returns a dictionary of integer busbar UIDs belonging to the islands with no generators or grid infeeds. The keys are the island slack busbar names or the first busbar names if no slack busbar is set for that island.

Returns
The busbars with no generators or grid infeeds belonging to each island.

Return type
dict(**str**,**list**(**int**))

GetBusbars(*bFetchFromSystem:* ***bool***)

Returns a dictionary of busbars. The keys are the busbar names.

Parameters
bFetchFromSystem (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Dictionary of busbars.

Return type
dict(**str**,*IscBusbar*)

GetBusbarsOrderedByVoltage(*bFetchFromSystem*: ***bool***) → **Tuple**[**int**]

Returns a tuple of busbar UIDs, sorted in ascending order of voltage and then by busbar name.

Parameters
bFetchFromSystem (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Tuple of busbars UIDs.

Return type
tuple(**int**)

GetBusbarAttachedBranches(*nBusbarUID*: ***int***, *bFetchFromSystem*: ***bool***) → **Tuple**[**int**]

Returns a tuple of branch UIDs attached to the busbar specified by busbar UID. Only branches are returned, not transformers.

Parameters

- **nBusbarUID** (*int*) – The selected busbar UID.
- **bFetchFromSystem** (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Tuple of busbars UIDs.

Return type
tuple(**int**)

GetBusbarAttachedTransformers(*nBusbarUID*: ***int***, *bFetchFromSystem*: ***bool***) → **Tuple**[**int**]

Returns a tuple of transformer UIDs attached to the busbar specified by busbar UID. Only transformers are returned, not branches or 3W transformers.

Parameters

- **nBusbarUID** (*int*) – The selected busbar UID.
- **bFetchFromSystem** (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Tuple of transformer UIDs.

Return type
tuple(int)

GetBusbarAttached3WTransformers*(nBusbarUID: int, bFetchFromSystem: bool)* → **Tuple[int]**

Returns a tuple of 3-winding transformer UIDs attached to the busbar specified by busbar UID. Only 3-winding transformers are returned, not 2-winding transformers or branches.

Parameters

- **nBusbarUID** (*int*) – The selected busbar UID.
- **bFetchFromSystem** (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Tuple of 3-winding transformer UIDs.

Return type
tuple(int)

GetBusbarAttachedUnbalancedBranches*(nBusbarUID: int, bFetchFromSystem: bool)* → **Tuple[int]**

Returns a tuple of unbalanced branch UIDs attached to the busbar specified by busbar UID. Only unbalanced branches are returned, not unbalanced transformers.

Parameters

- **nBusbarUID** (*int*) – The selected busbar UID.
- **bFetchFromSystem** (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Tuple of unbalanced branch UIDs.

Return type
tuple(int)

GetBusbarAttachedUnbalancedTransformers(*nBusbarUID:* ***int***, *bFetchFromSystem:* ***bool***) → **Tuple**[**int**]

Returns a tuple of unbalanced transformer UIDs attached to the busbar specified by busbar UID. Only unbalanced transformers are returned, not unbalanced branches.

Parameters

- **nBusbarUID** (***int***) – The selected busbar UID.
- **bFetchFromSystem** (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Tuple of unbalanced transformer UIDs.

Return type

tuple(**int**)

GetBranches(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscBranch instances. Key values (sPyName) are the Python names and the associated values are IscBranch instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of branches.

Return type

dict(**str**,*IscBranch*)

GetTransformers(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscTransformer instances. Keys (sPyName) are the Python names and the associated values are IscTransformer instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of transformers.

Return type

dict(**str**,*IscTransformer*)

***Get3WTransformers**(bFetchFromSystem: **bool**)*

Returns a dictionary of Isc3WTransformer instances. Keys (sPyName) are the Python names and the associated values are Isc3WTransformer instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of 3WTransformers.

Return type

dict(**str**,*Isc3WTransformer*)

***GetLoads**(bFetchFromSystem: **bool**)*

Returns a dictionary of IscLoad instances. Keys (sPyName) are the Python names and the associated values are IscLoad instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of loads.

Return type

dict(**str**,*IscLoad*)

***GetSynMachines**(bFetchFromSystem: **bool**)*

Returns a dictionary of IscSynMachine instances. Keys (sPyName) are the Python names and the associated values are IscSynMachine instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of synchronous machines.

Return type

dict(**str**,*IscSynMachine*)

***GetGridInfeeds**(bFetchFromSystem: **bool**)*

Returns a dictionary of IscGridInfeed instances. Keys (sPyName) are the Python names and the associated values are IscGridInfeed instances.

Parameters

bFetchFromSystem (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of grid infeeds.

Return type

dict(**str**,*IscGridInfeed*)

GetFilters(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscFilter instances. Keys (sPyName) are the Python names and the associated values are IscFilter instances.

Parameters

bFetchFromSystem (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of filters.

Return type

dict(**str**,*IscFilter*)

GetIndMachines(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscIndMachine instances. Keys (sPyName) are the Python names and the associated values are IscIndMachine instances.

Parameters

bFetchFromSystem (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of induction machines.

Return type

dict(**str**,*IscIndMachine*)

GetMechSwCapacitors(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscMechSwCapacitor instances. Keys (sPyName) are the Python names and the associated values are IscMechSwCapacitor instances.

Parameters

bFetchFromSystem (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of mechanical switch capacitors.

Return type

dict(**str**,*IscMechSwCapacitor*)

GetStaticVCs(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscStaticVC instances. Keys (sPyName) are the Python names and the associated values are IscStaticVC instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of static var compensators.

Return type

dict(**str**,*IscStaticVC*)

GetUMachines(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscUMachine instances. Keys (sPyName) are the Python names and the associated values are IscUMachine instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of universal machines.

Return type

dict(**str**,*IscUMachine*)

GetHarmonics(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscHarmonic instances. Keys (sPyName) are the Python names and the associated values are IscHarmonic instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of harmonics.

Return type

dict(**str**,*IscHarmonic*)

GetCircuitBreakers*(bFetchFromSystem:* ***bool****)*

Returns a dictionary of IscCircuitBreaker instances. Keys (sPyName) are the Python names and the associated values are IscCircuitBreaker instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of circuit breakers.

Return type

dict(**str**,*IscCircuitBreaker*)

GetBatteries*(bFetchFromSystem:* ***bool****)*

Returns a dictionary of IscBattery instances. Keys (sPyName) are the Python names and the associated values are IscBattery instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of batteries.

Return type

dict(**str**,*IscBattery*)

GetDCMachines*(bFetchFromSystem:* ***bool****)*

Returns a dictionary of IscDCMachine instances. Keys (sPyName) are the Python names and the associated values are IscDCMachine instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of DC machines.

Return type

dict(**str**,*IscDCMachine*)

GetConverters*(bFetchFromSystem:* ***bool****)*

Returns a dictionary of IscConverter instances. Keys (sPyName) are the Python names and the associated values are IscConverter instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of converters.

Return type

dict(**str**,*IscConverter*)

GetChoppers(*bFetchFromSystem*: ***bool***)

Returns a dictionary of IscChopper instances. Keys (sPyName) are the Python names and the associated values are IscChopper instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of choppers.

Return type

dict(**str**,*IscChopper*)

GetMGSets(*bFetchFromSystem*: ***bool***)

Returns a dictionary of IscMGSet instances. Keys (sPyName) are the Python names and the associated values are IscMGSet instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of MG sets.

Return type

dict(**str**,*IscMGSet*)

GetProtectionDevices(*bFetchFromSystem*: ***bool***)

Returns a dictionary of IscProtectionDevice instances. Keys (sPyName) are the Python names and the associated values are IscProtectionDevice instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Dictionary of protection devices.

Return type
dict(**str**,*IscProtectionDevice*)

GetUnbalancedLoads(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscUnbalancedLoad instances. Keys (sPyName) are the Python names and the associated values are IscUnbalancedLoad instances.

Parameters
bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Dictionary of unbalanced loads.

Return type
dict(**str**,*IscUnbalancedLoad*)

GetUnbalancedLines(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscUnbalancedLine instances. Keys (sPyName) are the Python names and the associated values are IscUnbalancedLine instances.

Parameters
bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Dictionary of unbalanced lines.

Return type
dict(**str**,*IscUnbalancedLine*)

GetUnbalancedTransformers(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscUnbalancedTransformer instances. Keys (sPyName) are the Python names and the associated values are IscUnbalancedTransformer instances.

Parameters
bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Dictionary of unbalanced transformers.

Return type

dict(**str**,*IscUnbalancedTransformer*)

GetVoltageRegulators(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscVoltageRegulator instances. Keys (sPyName) are the Python names and the associated values are IscVoltageRegulator instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of voltage regulators.

Return type

dict(**str**,*IscVoltageRegulator*)

GetAnnotations(*bFetchFromSystem:* ***bool***)

Returns a dictionary of IscAnnotation instances. Keys (sPyName) are the Python names and the associated values are IscAnnotation instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of annotations.

Return type

dict(**str**,*IscAnnotation*)

GetGroups()

Returns a dictionary of IscGroup instances. Keys (sPyName) are the Python names and the associated values are IscGroup instances.

Returns

Dictionary of groups.

Return type

dict(**str**,*IscGroup*)

GetGroupsForItem(*nUID:* ***int***) → **Tuple**[**int**]

Returns a tuple containing the group UIDs for each group that the component UID is a member of.

Parameters

nUID (***int***) – Component UID.

Returns
Tuple of group UIDs.

Return type
tuple(**int**)

GetPlugins()

Returns a dictionary of IscPlugin instances. Keys (sPyName) are the Python names and the associated values are IscPlugin instances.

Returns
Dictionary of plugins.

Return type
dict(**str**,*IscPlugin*)

GetBusbarUIDs(*bFetchFromSystem:* ***bool***)

Returns a dictionary of all busbar UIDs in the network. The keys are the integer UIDs and the values are the IscBusbar instances.

Parameters
bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Dictionary of all busbar UIDs.

Return type
dict(**int**,*IscBusbar*)

GetProtectionDeviceUIDs(*bFetchFromSystem:* ***bool***)

Returns a dictionary of all protection device UIDs in the network. The keys are the integer UIDs and the values are the IscProtectionDevice instances.

Parameters
bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns
Dictionary of all protection devices UIDs.

Return type
dict(**int**,*IscProtectionDevice*)

TraceBusbarUIDs(*nBranchUID:* ***int***, *bOpenBreakers:* ***bool***, *nGroupUID:* ***int***) → **List**[**int**]

Performs a network trace to identify all busbars that are connected to the selected branch. The network trace stops when it reaches any busbar that is a member of the group of the selected group UID or when it reaches a transformer.

Parameters

- **nBranchUID** (*int*) – The selected branch UID.
- **bOpenBreakers** (*bool*) – If True then the trace also stops if it finds an open circuit breaker.
- **nGroupUID** (*int*) – The selected group UID.

Returns
List of all busbar UIDs found by the trace.

Return type
list(**int**)

GetBusbarSlacks() → **List**[**str**]

Returns a list of all the busbar names contained in the network busbar slack list.

Returns
List of busbar names.

Return type
list(**str**)

GetBusbar(*nUID: **int***)

GetBusbar(*strPythonName: **str***)

Returns an IscBusbar instance for the busbar identified by the UID or the Python name.

You can use either nUID specifying the busbar UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected busbar UID.
- **strPythonName** (*str*) – The selected busbar name.

Returns
The busbar instance or None if such is not found.

Return type
IscBusbar

GetBranch(*nUID: **int***)

GetBranch(*strPythonName: **str***)

Returns an IscBranch instance for the branch identified by the UID or the Python name.

You can use either nUID specifying the branch UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected branch UID.
- **strPythonName** (*str*) – The selected branch name.

Returns
The branch instance or None if such is not found.

Return type
IscBranch

GetTransformer(*nUID*: ***int***)

GetTransformer(*strPythonName*: ***str***)

Returns an IscTransformer instance for the transformer identified by the UID or the Python name.

You can use either nUID specifying the transformer UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected transformer UID.
- **strPythonName** (*str*) – The selected transformer name.

Returns
The transformer instance or None if such is not found.

Return type
IscTransformer

Get3WTransformer(*nUID*: ***int***)

Get3WTransformer(*strPythonName*: ***str***)

Returns an Isc3WTransformer instance for the three winding transformer identified by the UID or the Python name.

You can use either nUID specifying the three winding transformer UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected three winding transformer UID.
- **strPythonName** (*str*) – The selected three winding transformer name.

Returns
The three winding transformer instance or None if such is not found.

Return type
Isc3WTransformer

GetLoad*(nUID: **int**)*

GetLoad*(strPythonName: **str**)*

Returns an IscLoad instance for the load identified by the UID or the Python name.

You can use either nUID specifying the load UID, or strPythonName specifying its name.

Parameters

- **nUID** (***int***) – The selected load UID.
- **strPythonName** (***str***) – The selected load name.

Returns
The load instance or None if such is not found.

Return type
IscLoad

GetSynMachine*(nUID: **int**)*

GetSynMachine*(strPythonName: **str**)*

Returns an IscSynMachine instance for the synchronous machine identified by the UID or the Python name.

You can use either nUID specifying the synchronous machine UID, or strPythonName specifying its name.

Parameters

- **nUID** (***int***) – The selected synchronous machine UID.
- **strPythonName** (***str***) – The selected synchronous machine name.

Returns
The synchronous machine instance or None if such is not found.

Return type
IscSynMachine

GetGridInfeed*(nUID: **int**)*

GetGridInfeed*(strPythonName: **str**)*

Returns an IscGridInfeed instance for the grid infeed identified by the UID or the Python name.

You can use either nUID specifying the grid infeed UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected grid infeed UID.
- **strPythonName** (*str*) – The selected grid infeed name.

Returns
The grid infeed instance or None if such is not found.

Return type
IscGridInfeed

GetIndMachine(*nUID: **int***)

GetIndMachine(*strPythonName: **str***)

Returns an IscIndMachine instance for the induction motor identified by the UID or the Python name.

You can use either nUID specifying the induction motor UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected induction motor UID.
- **strPythonName** (*str*) – The selected induction motor name.

Returns
The induction motor instance or None if such is not found.

Return type
IscIndMachine

GetFilter(*nUID: **int***)

GetFilter(*strPythonName: **str***)

Returns an IscFilter instance for the harmonic filter identified by the UID or the Python name.

You can use either nUID specifying the harmonic filter UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected harmonic filter UID.
- **strPythonName** (*str*) – The selected harmonic filter name.

Returns
The harmonic filter instance or None if such is not found.

Return type
IscFilter

GetMechSwCapacitor(*nUID: **int***)

GetMechSwCapacitor(*strPythonName:* ***str***)

Returns an IscMechSwCapacitor instance for the mechanically switched capacitor identified by the UID or the Python name.

You can use either nUID specifying the mechanically switched capacitor UID, or strPythonName specifying its name.

Parameters

- **nUID** (***int***) – The selected mechanically switched capacitor UID.
- **strPythonName** (***str***) – The selected mechanically switched capacitor name.

Returns
The mechanically switched capacitor instance or None if such is not found.

Return type
IscMechSwCapacitor

GetStaticVC(*nUID:* ***int***)

GetStaticVC(*strPythonName:* ***str***)

Returns an IscStaticVC instance for the static VAR compensator identified by the UID or the Python name.

You can use either nUID specifying the static VAR compensator UID, or strPythonName specifying its name.

Parameters

- **nUID** (***int***) – The selected static VAR compensator UID.
- **strPythonName** (***str***) – The selected static VAR compensator name.

Returns
The static VAR compensator instance or None if such is not found.

Return type
IscStaticVC

GetUMachine(*nUID:* ***int***)

GetUMachine(*strPythonName:* ***str***)

Returns an IscUMachine instance for the universal machine identified by the UID or the Python name.

You can use either nUID specifying the universal machine UID, or strPythonName specifying its name.

Parameters

- **nUID** (***int***) – The selected universal machine UID.

- **strPythonName** (*str*) – The selected universal machine name.

Returns
The universal machine instance or None if such is not found.

Return type
IscUMachine

GetHarmonic(*nUID: **int***)

GetHarmonic(*strPythonName: **str***)

Returns an IscHarmonic instance for the harmonic source identified by the UID or the Python name.

You can use either nUID specifying the harmonic source UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected harmonic source UID.
- **strPythonName** (*str*) – The selected harmonic source name.

Returns
The harmonic source instance or None if such is not found.

Return type
IscHarmonic

GetCircuitBreaker(*nUID: **int***)

GetCircuitBreaker(*strPythonName: **str***)

Returns an IscCircuitBreaker instance for the circuit breaker identified by the UID or the Python name.

You can use either nUID specifying the circuit breaker UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected circuit breaker UID.
- **strPythonName** (*str*) – The selected circuit breaker name.

Returns
The circuit breaker instance or None if such is not found.

Return type
IscCircuitBreaker

GetBattery(*nUID: **int***)

GetBattery(*strPythonName: **str***)

Returns an IscBattery instance for the DC battery identified by the UID or the Python name.

You can use either nUID specifying the DC battery UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected DC battery UID.
- **strPythonName** (*str*) – The selected DC battery name.

Returns
The DC battery instance or None if such is not found.

Return type
IscBattery

GetDCMachine*(nUID: **int**)*

GetDCMachine*(strPythonName: **str**)*

Returns an IscDCMachine instance for the DC machine identified by the UID or the Python name.

You can use either nUID specifying the DC machine UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected DC machine UID.
- **strPythonName** (*str*) – The selected DC machine name.

Returns
The DC machine instance or None if such is not found.

Return type
IscDCMachine

GetConverter*(nUID: **int**)*

GetConverter*(strPythonName: **str**)*

Returns an IscConverter instance for the AC/DC convertor identified by the UID or the Python name.

You can use either nUID specifying the AC/DC convertor UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected AC/DC convertor UID.
- **strPythonName** (*str*) – The selected AC/DC convertor name.

Returns
The AC/DC convertor instance or None if such is not found.

Return type
IscConverter

GetChopper(*nUID*: ***int***)

GetChopper(*strPythonName*: ***str***)

Returns an IscChopper instance for the AC/DC convertor identified by the UID or the Python name.

You can use either nUID specifying the AC/DC convertor UID, or strPythonName specifying its name.

Parameters

- **nUID** (***int***) – The selected AC/DC convertor UID.
- **strPythonName** (***str***) – The selected AC/DC convertor name.

Returns

The AC/DC chopper instance or None if such is not found.

Return type

IscChopper

GetMGSet(*nUID*: ***int***)

GetMGSet(*strPythonName*: ***str***)

Returns an IscMGSet instance for the motor generator set identified by the UID or the Python name.

You can use either nUID specifying the motor generator set UID, or strPython-Name specifying its name.

Parameters

- **nUID** (***int***) – The selected motor generator set UID.
- **strPythonName** (***str***) – The selected motor generator set name.

Returns

The motor generator set instance or None if such is not found.

Return type

IscMGSet

GetVoltageRegulator(*nUID*: ***int***)

GetVoltageRegulator(*strPythonName*: ***str***)

Returns an IscVoltageRegulator instance for the voltage regulator identified by the UID or the Python name.

You can use either nUID specifying the voltage regulator UID, or strPythonName specifying its name.

Parameters

- **nUID** (***int***) – The selected voltage regulator UID.
- **strPythonName** (***str***) – The selected voltage regulator name.

Returns
The voltage regulator instance or None if such is not found.

Return type
IscVoltageRegulator

GetProtectionDevice(*nUID*: ***int***)

GetProtectionDevice(*strPythonName*: ***str***)

Returns an IscProtectionDevice instance for the protection device identified by the UID or the Python name.

You can use either nUID specifying the protection device UID, or strPythonName specifying its name.

Parameters

- **nUID** (***int***) – The selected protection device UID.
- **strPythonName** (***str***) – The selected protection device name.

Returns
The protection device instance or None if such is not found.

Return type
IscProtectionDevice

GetAnnotation(*nUID*: ***int***)

GetAnnotation(*strPythonName*: ***str***)

Returns an IscAnnotation instance for the diagram annotation identified by the UID or the Python name.

You can use either nUID specifying the diagram annotation UID, or strPythonName specifying its name.

Parameters

- **nUID** (***int***) – The selected diagram annotation UID.
- **strPythonName** (***str***) – The selected diagram annotation name.

Returns
The diagram annotation instance or None if such is not found.

Return type
IscAnnotation

GetGroup(*nUID*: ***int***)

GetGroup(*strPythonName*: ***str***)

Returns an IscGroup instance for the group identified by the UID or the Python name.

You can use either nUID specifying the group UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected group UID.
- **strPythonName** (*str*) – The selected group name.

Returns

The group instance or None if such is not found.

Return type

IscGroup

GetPlugin(*nUID*: ***int***)

GetPlugin(*strPythonName*: ***str***)

Returns an IscPlugin instance for the plugin identified by the UID or the Python name.

You can use either nUID specifying the plugin UID, or strPythonName specifying its name.

Parameters

- **nUID** (*int*) – The selected plugin UID.
- **strPythonName** (*str*) – The selected plugin name.

Returns

The plugin instance or None if such is not found.

Return type

IscPlugin

GetUnbalancedLoad(*nUID*: ***int***)

GetUnbalancedLoad(*strPythonName*: ***str***)

Returns an IscUnbalancedLoad instance for the unbalanced load identified by the UID or the Python name.

Parameters

- **nUID** (*int*) – The selected unbalanced load UID.
- **strPythonName** (*str*) – The selected unbalanced load name.

Returns

The unbalanced load instance or None if such is not found.

Return type

IscUnbalancedLoad

GetUnbalancedLine(*nUID*: ***int***)

GetUnbalancedLine(*strPythonName:* ***str***)

Returns an IscUnbalancedLine instance for the unbalanced line identified by the UID or the Python name.

Parameters

- **nUID** (***int***) – The selected unbalanced line UID.
- **strPythonName** (***str***) – The selected unbalanced line name.

Returns

The unbalanced line instance or None if such is not found.

Return type

IscUnbalancedLine

GetUnbalancedTransformer(*nUID:* ***int***)

GetUnbalancedTransformer(*strPythonName:* ***str***)

Returns an IscUnbalancedTransformer instance for the unbalanced transformer identified by the UID or the Python name.

Parameters

- **nUID** (***int***) – The selected unbalanced transformer UID.
- **strPythonName** (***str***) – The selected unbalanced transformer name.

Returns

The unbalanced transformer instance or None if such is not found.

Return type

IscUnbalancedTransformer

GetNetworkData()

Returns an IscNetworkData instance of the network. The IscNetworkData object provides access to network wide properties such as the base MVA.

Returns

A network data instance of the network.

Return type

IscNetworkData

GetBusbarUID(*strName:* ***str***) → **int**

Returns the UID of a busbar with the given name.

Parameters

strName (***str***) – The selected busbar name.

Returns

The busbar UID, 0 if no matches or -N if we have N matches.

Return type

int

GetBranchUID(*nFromID:* ***int***, *nToID:* ***int***, *strName:* ***str***) → **int**

Returns the UID of a branch with the given name between two busbars that are specified by their UIDs.

Parameters

- **nFromID** (*int*) – The UID of the From busbar.
- **nToID** (*int*) – The UID of the To busbar.
- **strName** (*str*) – The selected branch name.

Returns

The branch UID, 0 if no matches or -N if we have N matches.

Return type

int

GetBranchUIDs(*bFetchFromSystem:* ***bool***)

Returns a dictionary of all branch UIDs in the network. The keys are the integer UIDs and the values are the IscBranch instances.

Parameters

bFetchFromSystem (*bool*) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of all branches.

Return type

dict(int,*IscBranch*)

GetTransformerUID(*nFromID:* ***int***, *nToID:* ***int***, *strName:* ***str***) → **int**

Returns the UID of a transformer with the given name between two busbars that are specified by their UIDs.

Parameters

- **nFromID** (*int*) – The UID of the From busbar.
- **nToID** (*int*) – The UID of the To busbar.
- **strName** (*str*) – The selected transformer name.

Returns

The transformer UID, 0 if no matches or -N if we have N matches.

Return type

int

GetTransformerUIDs(*bFetchFromSystem*: ***bool***)

Returns a dictionary of all transformer UIDs in the network. The keys are the integer UIDs and the values are the IscTransformer instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of all transformer UIDs.

Return type

dict(**int**,*IscTransformer*)

Get3WTransformerUID(*nFromID*: ***int***, *nToID*: ***int***, *nTeritaryID*: ***int***, *strName*: ***str***) → **int**

Returns the UID of a 3 winding transformer with the given name between three busbars that are specified by their UIDs.

Parameters

- **nFromID** (***int***) – The UID of the From busbar.
- **nToID** (***int***) – The UID of the To busbar.
- **nTeritaryID** (***int***) – The UID of the Teritary busbar.
- **strName** (***str***) – The selected 3 winding transformer name.

Returns

The 3 winding transformer UID, 0 if no matches or -N if we have N matches.

Return type

int

Get3WTransformerUIDs(*bFetchFromSystem*: ***bool***)

Returns a dictionary of all busbar UIDs in the network. The keys are the integer UIDs and the values are the IscBusbar instances.

Parameters

bFetchFromSystem (***bool***) – If set to True, IPSA rebuilds the data maps. If set to False, it only rebuilds if a new component has been built since last Get() function.

Returns

Dictionary of all 3WTransformers.

Return type

dict(**int**,*Isc3WTransformer*)

GetLoadUID(*nBusID*: ***int***, *strName*: ***str***) → **int**

Returns the UID of a load with specified name at busbar specified by its UID.

Parameters

- **nBusID** (***int***) – The UID of the busbar.
- **strName** (***str***) – The selected load name.

Returns

The load UID, 0 if no matches or -N if we have N matches.

Return type

int

GetLoadUIDs(*nBusID*: ***int***) → **List**[**int**]

Returns all loads connected to the busbars specified by the given UID.

Parameters

nBusID (***int***) – The UID of the busbar.

Returns

List of load UIDs.

Return type

list(**int**)

GetSynMachineUID(*nBusID*: ***int***, *strName*: ***str***) → **int**

Returns the UID of a synchronous machine with specified name at busbar specified by its UID.

Parameters

- **nBusID** (***int***) – The UID of the busbar.
- **strName** (***str***) – The selected synchronous machine name.

Returns

The synchronous machine UID, 0 if no matches or -N if we have N matches.

Return type

int

GetSynMachineUIDs(*nBusID*: ***int***) → **List**[**int**]

Returns all synchronous machines connected to the busbars specified by the given UID.

Parameters

nBusID (***int***) – The UID of the busbar.

Returns

List of synchronous machine UIDs.

Return type
list(int)

GetGridInfeedUID(nBusID: int, strName: str) → int

Returns the UID of a grid infeed with specified name at busbar specified by its UID.

Parameters

- **nBusID** (*int*) – The UID of the busbar.
- **strName** (*str*) – The selected grid infeed name.

Returns
The grid infeed UID, 0 if no matches or -N if we have N matches.

Return type
int

GetGridInfeedUIDs(nBusID: int) → List[int]

Returns all grid infeeds connected to the busbars specified by the given UID.

Parameters
nBusID (*int*) – The UID of the busbar.

Returns
List of grid infeed UIDs.

Return type
list(int)

GetIndMachineUID(nBusID: int, strName: str) → int

Returns the UID of an induction machine with specified name at busbar specified by its UID.

Parameters

- **nBusID** (*int*) – The UID of the busbar.
- **strName** (*str*) – The selected induction machine name.

Returns
The induction machine UID, 0 if no matches or -N if we have N matches.

Return type
int

GetIndMachineUIDs(nBusID: int) → List[int]

Returns all induction machines connected to the busbars specified by the given UID.

Parameters
nBusID (*int*) – The UID of the busbar.

Returns
List of induction machine UIDs.

Return type
list(int)

GetFilterUID(nBusID: int, strName: str) → int

Returns the UID of a filter with specified name at busbar specified by its UID.

Parameters

- **nBusID** (*int*) – The UID of the busbar.
- **strName** (*str*) – The selected filter name.

Returns
The filter UID, 0 if no matches or -N if we have N matches.

Return type
int

GetFilterUIDs(nBusID: int) → List[int]

Returns all filters connected to the busbars specified by the given UID.

Parameters
nBusID (*int*) – The UID of the busbar.

Returns
List of filter UIDs.

Return type
list(int)

GetMechSwCapacitorUID(nBusID: int, strName: str) → int

Returns the UID of a mechanically switched capacitor with specified name at busbar specified by its UID.

Parameters

- **nBusID** (*int*) – The UID of the busbar.
- **strName** (*str*) – The selected mechanically switched capacitor name.

Returns
The mechanically switched capacitor UID, 0 if no matches or -N if we have N matches.

Return type
int

*GetMechSwCapacitorUIDs(nBusID: **int**)* → **List**[**int**]

Returns all mechanically switched capacitors connected to the busbars specified by the given UID.

Parameters

nBusID (***int***) – The UID of the busbar.

Returns

List of mechanically switched capacitor UIDs.

Return type

list(**int**)

*GetStaticVCUID(nBusID: **int**, strName: **str**)* → **int**

Returns the UID of a static VAr compensator with specified name at busbar specified by its UID.

Parameters

- **nBusID** (***int***) – The UID of the busbar.
- **strName** (***str***) – The selected static VAr compensator name.

Returns

The static VAr compensator UID, 0 if no matches or -N if we have N matches.

Return type

int

*GetStaticVCUIDs(nBusID: **int**)* → **List**[**int**]

Returns all static VAr compensators connected to the busbars specified by the given UID.

Parameters

nBusID (***int***) – The UID of the busbar.

Returns

List of static VAr compensator UIDs.

Return type

list(**int**)

*GetUMachineUID(nBusID: **int**, strName: **str**)* → **int**

Returns the UID of a universal machine with specified name at busbar specified by its UID.

Parameters

- **nBusID** (***int***) – The UID of the busbar.
- **strName** (***str***) – The selected universal machine name.

Returns

The universal machine UID, 0 if no matches or -N if we have N matches.

Return type

int

GetUMachineUIDs(*nBusID: **int***) → **List[int]**

Returns all universal machines connected to the busbars specified by the given UID.

Parameters

nBusID (***int***) – The UID of the busbar.

Returns

List of universal machine UIDs.

Return type

list(int)

GetHarmonicUID(*nBusID: **int**, strName: **str***) → **int**

Returns the UID of a harmonic source with specified name at busbar specified by its UID.

Parameters

- **nBusID** (***int***) – The UID of the busbar.
- **strName** (***str***) – The selected harmonic source name.

Returns

The harmonic source UID, 0 if no matches or -N if we have N matches.

Return type

int

GetHarmonicUIDs(*nBusID: **int***) → **List[int]**

Returns all harmonic sources connected to the busbars specified by the given UID.

Parameters

nBusID (***int***) – The UID of the busbar.

Returns

List of harmonic source UIDs.

Return type

list(int)

GetCircuitBreakerUID(*nBranchOrTxID: **int**, nClosestBusbarUID: **int***) → **int**

Returns the UID of a circuit breaker located on the branch or transformer specified by its UID. The From or To end of the branch is specified by the nClosest-BusbarUID parameter.

Parameters

- **nBranchOrTxID** (*int*) – The UID of the branch or the transformer.
- **nClosestBusbarUID** (*int*) – Identifies the busbar at either the From or To end.

Returns
The circuit breaker UID, 0 if no matches.

Return type
int

GetFromCircuitBreakerUID(*nBranchOrTxID: int*) → int

Returns the UID of a circuit breaker located on the "From" end of the branch or transformer specified by its UID.

Parameters
nBranchOrTxID (*int*) – The UID of the branch or the transformer.

Returns
The circuit breaker UID, 0 if no matches.

Return type
int

GetToCircuitBreakerUID(*nBranchOrTxID: int*) → int

Returns the UID of a circuit breaker located on the "To" end of the branch or transformer specified by its UID.

Parameters
nBranchOrTxID (*int*) – The UID of the branch or the transformer.

Returns
The circuit breaker UID, 0 if no matches.

Return type
int

GetBatteryUID(*nBusID: int, strName: str*) → int

Returns the UID of a battery with specified name at busbar specified by its UID.

Parameters

- **nBusID** (*int*) – The UID of the busbar.
- **strName** (*str*) – The selected battery name.

Returns
The battery UID, 0 if no matches or -N if we have N matches.

Return type
int

GetBatteryUIDs(nBusID: int) → List[int]

Returns all batteries connected to the busbars specified by the given UID.

Parameters
nBusID (*int*) – The UID of the busbar.

Returns
List of battery UIDs.

Return type
list(int)

GetDCMachineUID(nBusID: int, strName: str) → int

Returns the UID of a DC Machine with specified name at busbar specified by its UID.

Parameters

- **nBusID** (*int*) – The UID of the busbar.
- **strName** (*str*) – The selected DC Machine name.

Returns
The DC Machine UID, 0 if no matches or -N if we have N matches.

Return type
int

GetDCMachineUIDs(nBusID: int) → List[int]

Returns all DC Machines connected to the busbars specified by the given UID.

Parameters
nBusID (*int*) – The UID of the busbar.

Returns
List of DC Machine UIDs.

Return type
list(int)

GetConverterUID(nFromID: int, nToID: int, strName: str) → int

Returns the UID of a converter with the given name between two busbars that are specified by their UIDs.

Parameters

- **nFromID** (*int*) – The UID of the From busbar.
- **nToID** (*int*) – The UID of the To busbar.
- **strName** (*str*) – The selected converter name.

Returns
The converter UID, 0 if no matches or -N if we have N matches.

Return type
int

***GetConverterUIDs**(nFromID: int, nToID: int)* → List[int]

Returns all converters connected between two busbars that are specified by their UIDs.

Parameters

- **nFromID** (*int*) – The UID of the From busbar.
- **nToID** (*int*) – The UID of the To busbar.

Returns
List of converter UIDs.

Return type
list(int)

***GetChopperUID**(nFromID: int, nToID: int, strName: str)* → int

Returns the UID of a chopper with the given name between two busbars that are specified by their UIDs.

Parameters

- **nFromID** (*int*) – The UID of the From busbar.
- **nToID** (*int*) – The UID of the To busbar.
- **strName** (*str*) – The selected chopper name.

Returns
The chopper UID, 0 if no matches or -N if we have N matches.

Return type
int

***GetChopperUIDs**(nFromID: int, nToID: int)* → List[int]

Returns all choppers connected between two busbars that are specified by their UIDs.

Parameters

- **nFromID** (*int*) – The UID of the From busbar.
- **nToID** (*int*) – The UID of the To busbar.

Returns
List of chopper UIDs.

Return type
list(int)

***GetMGSetUID**(nFromID: **int**, nToID: **int**, strName: **str**)* → **int**

Returns the UID of a motor/generator with the given name between two busbars that are specified by their UIDs.

Parameters

- **nFromID** (***int***) – The UID of the From busbar.
- **nToID** (***int***) – The UID of the To busbar.
- **strName** (***str***) – The selected motor/generator name.

Returns

The motor/generator UID, 0 if no matches or -N if we have N matches.

Return type

int

***GetMGSetUIDs**(nFromID: **int**, nToID: **int**)* → **List**[**int**]

Returns all motors/generators connected between two busbars that are specified by their UIDs.

Parameters

- **nFromID** (***int***) – The UID of the From busbar.
- **nToID** (***int***) – The UID of the To busbar.

Returns

List of motor/generator UIDs.

Return type

list(**int**)

***GetUnbalancedLoadUID**(nBusID: **int**, strName: **str**)* → **int**

Returns the UID of an unbalanced load with specified name at busbar specified by its UID.

Parameters

- **nBusID** (***int***) – The UID of the busbar.
- **strName** (***str***) – The selected unbalanced load name.

Returns

The unbalanced load UID, 0 if no matches or -N if we have N matches.

Return type

int

***GetUnbalancedLineUID**(nFromID: **int**, nToID: **int**, strName: **str**)* → **int**

Returns the UID of an unbalanced line with the given name between two busbars that are specified by their UIDs.

Parameters

- **nFromID** (*int*) – The UID of the From busbar.
- **nToID** (*int*) – The UID of the To busbar.
- **strName** (*str*) – The selected unbalanced line name.

Returns

The unbalanced line UID, 0 if no matches or -N if we have N matches.

Return type

int

GetUnbalancedTransformerUID(*nFromID: **int**, nToID: **int**, strName: **str***) → **int**

Returns the UID of an unbalanced transformer with the given name between two busbars that are specified by their UIDs.

Parameters

- **nFromID** (*int*) – The UID of the From busbar.
- **nToID** (*int*) – The UID of the To busbar.
- **strName** (*str*) – The selected unbalanced transformer name.

Returns

The unbalanced transformer UID, 0 if no matches or -N if we have N matches.

Return type

int

GetVoltageRegulatorUID(*nBranchID: **int***) → **int**

Returns the UID of a voltage regulator at branch specified by its UID.

Parameters

nBranchID (*int*) – The UID of the branch.

Returns

The voltage regulator UID.

Return type

int

GetProfileUID(*nUID: **int***) → **int**

Returns the integer UID of the profile for the component UID.

Parameters

nUID (*int*) – The UID of component. nUID may be the UID of a load, generator, grid infeed or Universal Machine.

Returns

The profile for the component UID, 0 if the component nUID does not have a profile assigned to it, or if nUID is not a load, generator, grid infeed or universal machine.

Return type
int

CreateBusbar(*strName: **str***) → **int**

Returns the UID for the newly created busbar.

Warning: It is up to the script to ensure that the busbar name is unique.

Parameters
strName (***str***) – The branch name string if required.

Returns
The UID for the newly created busbar, 0 on failure.

Return type
int

CreateBusbarNoGraphics(*strName: **str***)

Returns an IscBusbar object for the newly created busbar.

Warning: It is up to the script to ensure that the busbar name is unique.

Parameters
strName (***str***) – The busbar name string if required.

Returns
The IscBusbar object for the newly created busbar.

Return type
IscBusbar

CreateBranch(*nFromBusbarUID: **int**, nToBusbarUID: **int**, strName: **str***) → **int**

CreateBranch(*pFromBusbar, pToBusbar, strName: **str***)

Returns the UID or an IscBranch object for the newly created branch.

Parameters

- **nFromBusbarUID** (***int***) – The “From” busbar UID.
- **nToBusbarUID** (***int***) – The “To” busbar UID.
- **pFromBusbar** (*IscBusbar*) – The “From” busbar.
- **pToBusbar** (*IscBusbar*) – The “To” busbar.
- **strName** (***str***) – The branch name string if required.

Returns
The UID for the newly created branch, 0 on failure.

Return type
int

Returns

The IscBranch object for the newly created branch.

Return type

IscBranch

CreateTransformer(*nFromBusbarUID: **int**, nToBusbarUID: **int**, strName: **str***) → **int**

CreateTransformer(*pFromBusbar, pToBusbar, strName: **str***)

Returns the UID or an IscTransformer object for the newly created transformer.

Parameters

- **nFromBusbarUID** (***int***) – The "From" busbar UID.
- **nToBusbarUID** (***int***) – The "To" busbar UID.
- **pFromBusbar** (*IscBusbar*) – The "From" busbar.
- **pToBusbar** (*IscBusbar*) – The "To" busbar.
- **strName** (***str***) – The transformer name string if required.

Returns

The UID for the newly created transformer, 0 on failure.

Return type

int

Returns

The IscTransformer object for the newly created transformer.

Return type

IscTransformer

Create3WTransformer(*nFromBusbarUID: **int**, nToBusbarUID: **int**, nTeritaryBusUID: **int**, strName: **str***) → **int**

Create3WTransformer(*pFromBusbar, pToBusbar, pTeritaryBus, strName: **str***)

Returns the UID or an Isc3WTransformer object for the newly created 3-winding transformer.

Parameters

- **nFromBusbarUID** (***int***) – The "From" busbar UID.
- **nToBusbarUID** (***int***) – The "To" busbar UID.
- **nTeritaryBusUID** (***int***) – The "Teritary" busbar UID.
- **pFromBusbar** (*IscBusbar*) – The "From" busbar.
- **pToBusbar** (*IscBusbar*) – The "To" busbar.
- **pTeritaryBus** (*IscBusbar*) – The "Teritary" busbar.
- **strName** (***str***) – The 3-winding transformer name string if required.

Returns
The UID for the newly created 3-winding transformer, 0 on failure.

Return type
int

Returns
The Isc3WTransformer object for the newly created 3-winding transformer.

Return type
Isc3WTransformer

CreateLoad(*nAtBusbarUID: **int**, strName: **str***) → **int**

CreateLoad(*pAtBusbar, strName: **str***)

Returns the UID or an IscLoad object for the newly created load.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The load name string if required.

Returns
The UID for the newly created load, 0 on failure.

Return type
int

Returns
The IscLoad object for the newly created load.

Return type
IscLoad

CreateIndMachine(*nAtBusbarUID: **int**, strName: **str***) → **int**

CreateIndMachine(*pAtBusbar, strName: **str***)

Returns the UID or an IscIndMachine object for the newly created induction machine.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The induction machine name string if required.

Returns
The UID for the newly created induction machine, 0 on failure.

Return type
int

Returns
The IscIndMachine object for the newly created induction machine.

Return type
IscIndMachine

CreateSynMachine(*nAtBusbarUID: **int**, strName: **str***) → **int**

CreateSynMachine(*pAtBusbar, strName: **str***)

Returns the UID or an IscSynMachine object for the newly created synchronous machine.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The synchronous machine name string if required.

Returns
The UID for the newly created synchronous machine, 0 on failure.

Return type
int

Returns
The IscSynMachine object for the newly created synchronous machine.

Return type
IscSynMachine

CreateGridInfeed(*nAtBusbarUID: **int**, strName: **str***) → **int**

CreateGridInfeed(*pAtBusbar, strName: **str***)

Returns the UID or an IscGridInfeed object for the newly created grid infeed.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The grid infeed name string if required.

Returns
The UID for the newly created grid infeed, 0 on failure.

Return type
int

Returns
The IscGridInfeed object for the newly created grid infeed.

Return type
IscGridInfeed

CreateFilter(*nAtBusbarUID*: ***int***, *strName*: ***str***) → **int**

CreateFilter(*pAtBusbar*, *strName*: ***str***)

Returns the UID or an IscFilter object for the newly created filter.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The filter name string if required.

Returns
The UID for the newly created filter, 0 on failure.

Return type
int

Returns
The IscFilter object for the newly created filter.

Return type
IscFilter

CreateHarmonic(*nAtBusbarUID*: ***int***, *strName*: ***str***) → **int**

CreateHarmonic(*pAtBusbar*, *strName*: ***str***)

Returns the UID or an IscHarmonic object for the newly created harmonic source.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The harmonic source name string if required.

Returns
The UID for the newly created harmonic source, 0 on failure.

Return type
int

Returns
The IscHarmonic object for the newly created harmonic source.

Return type
IscHarmonic

CreateMechSwCapacitor(*nAtBusbarUID: **int**, strName: **str***) → **int**

CreateMechSwCapacitor(*pAtBusbar, strName: **str***)

Returns the UID or an IscMechSwCapacitor object for the newly created mechanically switched capacitor.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The capacitor name string if required.

Returns

The UID for the newly created mechanically switched capacitor, 0 on failure.

Return type

int

Returns

The IscMechSwCapacitor object for the newly created mechanically switched capacitor.

Return type

IscMechSwCapacitor

CreateCircuitBreaker(*nBranchOrTxUID: **int**, bAtFromEnd: **bool**, strName: **str***) → **int**

CreateCircuitBreaker(*pBranchOrTx, bAtFromEnd: **bool**, strName: **str***)

Returns the UID or an IscCircuitBreaker object for the newly created circuit breaker. In order to draw this component, the function IscDiagram.DrawUndrawnItemsAttachedToBusbar needs to be called before IscDiagram.DrawLine.

Parameters

- **nBranchOrTxUID** (***int***) – The UID of the branch or the transformer where the circuit breaker is located.
- **pBranchOrTx** (*IscBranch or IscTransformer*) – The IscBranch or IscTransformer object of the branch or transformer where the circuit breaker is located.
- **bAtFromEnd** (***bool***) – Adds the circuit breaker to the "From" end of the component, if True.
- **strName** (***str***) – The circuit breaker name string if required.

Returns

The UID for the newly created circuit breaker, 0 on failure.

Return type
int

Returns
The IscCircuitBreaker object for the newly created circuit breaker.

Return type
IscCircuitBreaker

CreateStaticVC(*nAtBusbarUID*: ***int***, *strName*: ***str***) → **int**

CreateStaticVC(*pAtBusbar*, *strName*: ***str***)

Returns the UID or an IscStaticVC object for the newly created static VAr compensator.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The static VAr compensator name string if required.

Returns
The UID for the newly created static VAr compensator, 0 on failure.

Return type
int

Returns
The IscStaticVC object for the newly created static VAr compensator.

Return type
IscStaticVC

CreateUMachine(*nAtBusbarUID*: ***int***, *strName*: ***str***) → **int**

CreateUMachine(*pAtBusbar*, *strName*: ***str***)

Returns the UID or an IscUMachine object for the newly created universal machine.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The universal machine name string if required.

Returns
The UID for the newly created universal machine, 0 on failure.

Return type
int

Returns
The IscUMachine object for the newly created universal machine.

Return type
IscUMachine

CreateBattery(*nAtBusbarUID: **int**, strName: **str***) → **int**

CreateBattery(*pAtBusbar, strName: **str***)

Returns the UID or an IscBattery object for the newly created battery.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The battery name string if required.

Returns
The UID for the newly created battery, 0 on failure.

Return type
int

Returns
The IscBattery object for the newly created battery.

Return type
IscBattery

CreateDCMachine(*nAtBusbarUID: **int**, strName: **str***) → **int**

CreateDCMachine(*pAtBusbar, strName: **str***)

Returns the UID or an IscDCMachine object for the newly created DC machine.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The DC machine name string if required.

Returns
The UID for the newly created DC machine, 0 on failure.

Return type
int

Returns
The IscDCMachine object for the newly created DC machine.

Return type
IscDCMachine

*CreateConverter(nFromBusbarUID: **int**, nToBusbarUID: **int**, strName: **str**)* → **int**

*CreateConverter(pFromBusbar, pToBusbar, strName: **str**)*

Returns the UID or an IscConverter object for the newly created AC/DC converter.

Parameters

- **nFromBusbarUID** (*int*) – The "From" busbar UID.
- **nToBusbarUID** (*int*) – The "To" busbar UID.
- **pFromBusbar** (*IscBusbar*) – The "From" busbar.
- **pToBusbar** (*IscBusbar*) – The "To" busbar.
- **strName** (*str*) – The AC/DC converter name string if required.

Returns

The UID for the newly created AC/DC converter, 0 on failure.

Return type

int

Returns

The IscConverter object for the newly created AC/DC converter.

Return type

IscConverter

*CreateChopper(nFromBusbarUID: **int**, nToBusbarUID: **int**, strName: **str**)* → **int**

*CreateChopper(pFromBusbar, pToBusbar, strName: **str**)*

Returns the UID or an IscChopper object for the newly created chopper.

Parameters

- **nFromBusbarUID** (*int*) – The "From" busbar UID.
- **nToBusbarUID** (*int*) – The "To" busbar UID.
- **pFromBusbar** (*IscBusbar*) – The "From" busbar.
- **pToBusbar** (*IscBusbar*) – The "To" busbar.
- **strName** (*str*) – The DC/DC converter name string if required.

Returns

The UID for the newly created chopper, 0 on failure.

Return type

int

Returns

The IscChopper object for the newly created chopper.

Return type
IscChopper

CreateMGSet(*nFromBusbarUID: **int**, nToBusbarUID: **int**, strName: **str***) → **int**

CreateMGSet(*pFromBusbar, pToBusbar, strName: **str***)

Returns the UID or an IscMGSet object for the newly created motor/generator set.

Parameters

- **nFromBusbarUID** (***int***) – The "From" busbar UID.
- **nToBusbarUID** (***int***) – The "To" busbar UID.
- **pFromBusbar** (*IscBusbar*) – The "From" busbar.
- **pToBusbar** (*IscBusbar*) – The "To" busbar.
- **strName** (***str***) – The motor/generator set name string if required.

Returns
The UID for the newly created motor/generator set, 0 on failure.

Return type
int

Returns
The IscMGSet object for the newly created motor/generator set.

Return type
IscMGSet

CreateVoltageRegulator(*nBranchUID: **int**, strName: **str***) → **int**

CreateVoltageRegulator(*pBranch, strName: **str***)

Returns the UID or an IscVoltageRegulator object for the newly created voltage regulator.

Parameters

- **nBranchUID** (***int***) – The branch the voltage regulator is upon
- **pBranch** (*IscBranch*) – The branch the voltage regulator is upon
- **strName** (***str***) – The voltage regulator name string if required.

Returns
The UID for the newly created voltage regulator, 0 on failure.

Return type
int

Returns
The IscVoltageRegulator object for the newly created voltage regulator.

Return type
IscVoltageRegulator

CreateUnbalancedLoad(*nAtBusbarUID: **int**, strName: **str***) → **int**

CreateUnbalancedLoad(*pAtBusbar, strName: **str***)

Returns the UID or an IscUnbalancedLoad object for the newly created unbalanced load.

Parameters

- **nAtBusbarUID** (***int***) – The busbar UID.
- **pAtBusbar** (*IscBusbar*) – The busbar.
- **strName** (***str***) – The unbalanced load name string if required.

Returns
The UID for the newly created unbalanced load, 0 on failure.

Return type
int

Returns
The IscUnbalancedLoad object for the newly created unbalanced load.

Return type
IscUnbalancedLoad

CreateUnbalancedLine(*nFromBusbarUID: **int**, nToBusbarUID: **int**, strName: **str***) → **int**

CreateUnbalancedLine(*pFromBusbar, pToBusbar, strName: **str***)

Returns the UID or an IscUnbalancedLine object for the newly created unbalanced line.

Parameters

- **nFromBusbarUID** (***int***) – The “From” busbar UID.
- **nToBusbarUID** (***int***) – The “To” busbar UID.
- **pFromBusbar** (*IscBusbar*) – The “From” busbar.
- **pToBusbar** (*IscBusbar*) – The “To” busbar.
- **strName** (***str***) – The unbalanced line name string if required.

Returns
The UID for the newly created unbalanced line, 0 on failure.

Return type
int

Returns
The IscUnbalancedLine object for the newly created unbalanced line.

Return type
IscUnbalancedLine

CreateUnbalancedTransformer(*nFromBusbarUID: **int**, nToBusbarUID: **int**, strName: **str***) → **int**

CreateUnbalancedTransformer(*pFromBusbar, pToBusbar, strName: **str***)

Returns the UID or an IscUnbalancedTransformer object for the newly created unbalanced transformer.

Parameters

- **nFromBusbarUID** (***int***) – The "From" busbar UID.
- **nToBusbarUID** (***int***) – The "To" busbar UID.
- **pFromBusbar** (*IscBusbar*) – The "From" busbar.
- **pToBusbar** (*IscBusbar*) – The "To" busbar.
- **strName** (***str***) – The unbalanced transformer name string if required.

Returns
The UID for the newly created unbalanced transformer, 0 on failure.

Return type
int

Returns
The IscUnbalancedTransformer object for the newly created unbalanced transformer.

Return type
IscUnbalancedTransformer

CreateGroup(*strName: **str**, nGroupType: **int***) → **int**

Create a new empty group of components and returns the group UID. Group types:

- 0 = No group type
- 1 = Area type group (contains all busbars in an area)
- 2 = Mixed item group
- 3 = Load scaling group
- 4 = Load transfer group
- 5 = Protection device group

Parameters

- **strName** (***str***) – The group name.

- **nGroupType** (*int*) – The group type.

Returns
The group UID, 0 on failure.

Return type
int

CreateGroupNoGraphics(*strName: **str**, nGroupType: **int***)

Create a new empty group of components and returns the group object. Group types:

- 0 = No group type
- 1 = Area type group (contains all busbars in an area)
- 2 = Mixed item group
- 3 = Load scaling group
- 4 = Load transfer group
- 5 = Protection device group

Parameters

- **strName** (*str*) – The group name.
- **nGroupType** (*int*) – The group type.

Returns
The IscGroup object.

Return type
IscGroup

CreatePlugin(*nCompUID: **int**, sPluginName: **str**, sName: **str***) → **int**

Returns the UID for the newly created plugin. A different plugin UID is required for each component with a plugin, therefore this function should be used every time a plugin is assigned to a component, even if the same type of plugin is being assigned.

Parameters

- **nCompUID** (*int*) – The UID of the component to which the plugin is to be assigned to.
- **sPluginName** (*str*) – The name of the plugin itself, for example 'Constant Current Load'.
- **sName** (*str*) – The user defined plugin name or empty string.

Returns
The plugin UID, 0 on failure.

Return type
int

DeleteBusbar*(pBusbar)* → **bool**

Deletes a busbar by passing the IscBusbar object for deletion.

Parameters
pBusbar (*IscBusbar*) – The IscBusbar object for deletion.

Returns
True if successful.

Return type
bool

DeleteBranch*(pBranch)* → **bool**

Deletes a branch by passing the IscBranch object for deletion and all the circuit breakers attached to it.

Parameters
pBranch (*IscBranch*) – The IscBranch object for deletion.

Returns
True if successful.

Return type
bool

DeleteTransformer*(pTransformer)* → **bool**

Deletes a transformer by passing the IscTransformer object for deletion.

Parameters
pTransformer (*IscTransformer*) – The IscTransformer object for deletion.

Returns
True if successful.

Return type
bool

Delete3WTransformer*(p3WTransformer)* → **bool**

Deletes a 3-winding transformer by passing the Isc3WTransformer object for deletion.

Parameters
p3WTransformer (*Isc3WTransformer*) – The Isc3WTransformer object for deletion.

Returns
True if successful.

Return type
bool

DeleteLoad(*pLoad*) → **bool**

Deletes a load by passing the IscLoad object for deletion.

Parameters
pLoad (*IscLoad*) – The IscLoad object for deletion.

Returns
True if successful.

Return type
bool

DeleteSynMachine(*pSynMachine*) → **bool**

Deletes a synchronous machine by passing the IscSynMachine object for deletion.

Parameters
pSynMachine (*IscSynMachine*) – The IscSynMachine object for deletion.

Returns
True if successful.

Return type
bool

DeleteIndMachine(*pIndMachine*) → **bool**

Deletes an induction machine by passing the IscIndMachine object for deletion.

Parameters
pIndMachine (*IscIndMachine*) – The IscIndMachine object for deletion.

Returns
True if successful.

Return type
bool

DeleteGridInfeed(*pGridInfeed*) → **bool**

Deletes a grid infeed by passing the IscSynMachine object for deletion.

Parameters
pGridInfeed (*IscSynMachine*) – The IscSynMachine object for deletion.

Returns
True if successful.

Return type
bool

DeleteFilter(*pFilter*) → **bool**

Deletes a filter by passing the IscFilter object for deletion.

Parameters
pFilter (*IscFilter*) – The IscFilter object for deletion.

Returns
True if successful.

Return type
bool

DeleteMechSwCapacitor(*pMechSwCapacitor*) → **bool**

Deletes a mechanical switched capacitor by passing the IscMechSwCapacitor object for deletion.

Parameters
pMechSwCapacitor (*IscMechSwCapacitor*) – The IscMechSwCapacitor object for deletion.

Returns
True if successful.

Return type
bool

DeleteStaticVC(*pStaticVC*) → **bool**

Deletes a synchronous machine by passing the IscStaticVC object for deletion.

Parameters
pStaticVC (*IscStaticVC*) – The IscStaticVC object for deletion.

Returns
True if successful.

Return type
bool

DeleteUMachine(*pUMachine*) → **bool**

Deletes an universal machine by passing the IscUMachine object for deletion.

Parameters
pUMachine (*IscUMachine*) – The IscUMachine object for deletion.

Returns
True if successful.

Return type
bool

***DeleteHarmonic**(pHarmonic)* → **bool**

Deletes a harmonic source by passing the IscHarmonic object for deletion.

Parameters

pHarmonic (*IscHarmonic*) – The IscHarmonic object for deletion.

Returns

True if successful.

Return type

bool

***DeleteCircuitBreaker**(pCircuitBreaker)* → **bool**

Deletes a circuit breaker by passing the IscCircuitBreaker object for deletion.

Parameters

pCircuitBreaker (*IscCircuitBreaker*) – The IscCircuitBreaker object for deletion.

Returns

True if successful.

Return type

bool

***DeleteBattery**(pBattery)* → **bool**

Deletes a battery by passing the IscBattery object for deletion.

Parameters

pBattery (*IscBattery*) – The IscBattery object for deletion.

Returns

True if successful.

Return type

bool

***DeleteDCMachine**(pDCMachine)* → **bool**

Deletes a DC machine by passing the IscDCMachine object for deletion.

Parameters

pDCMachine (*IscDCMachine*) – The IscDCMachine object for deletion.

Returns

True if successful.

Return type

bool

***DeleteConverter**(pConverter)* → **bool**

Deletes a converter by passing the IscConverter object for deletion.

Parameters
pConverter (*IscConverter*) – The IscConverter object for deletion.

Returns
True if successful.

Return type
bool

DeleteChopper(*pChopper*) → **bool**

Deletes a chopper by passing the IscChopper object for deletion.

Parameters
pChopper (*IscChopper*) – The IscChopper object for deletion.

Returns
True if successful.

Return type
bool

DeleteMGSet(*pMGSet*) → **bool**

Deletes a motor/generator set by passing the IscMGSet object for deletion.

Parameters
pMGSet (*IscMGSet*) – The IscMGSet object for deletion.

Returns
True if successful.

Return type
bool

DeleteVoltageRegulator(*pVoltageRegulator*) → **bool**

Deletes a voltage regulator by passing the IscVoltageRegulator object for deletion.

Parameters
pVoltageRegulator (*IscVoltageRegulator*) – The IscVoltageRegulator object for deletion.

Returns
True if successful.

Return type
bool

DeleteAnnotation(*pAnnotation*) → **bool**

Deletes an annotation by passing the IscAnnotation object for deletion.

Parameters
pAnnotation (*IscAnnotation*) – The IscAnnotation object for deletion.

Returns
True if successful.

Return type
bool

DeleteUnbalancedLoad(*pUnbalancedLoad*) → **bool**

Deletes an unbalanced load by passing the IscUnbalancedLoad object for deletion.

Parameters
pUnbalancedLoad (*IscUnbalancedLoad*) – The IscUnbalancedLoad object for deletion.

Returns
True if successful.

Return type
bool

DeleteUnbalancedLine(*pUnbalancedLine*) → **bool**

Deletes an unbalanced line by passing the IscUnbalancedLine object for deletion.

Parameters
pUnbalancedLine (*IscUnbalancedLine*) – The IscUnbalancedLine object for deletion.

Returns
True if successful.

Return type
bool

DeleteUnbalancedTransformer(*pUnbalancedTransformer*) → **bool**

Deletes an unbalanced transformer by passing the IscUnbalancedTransformer object for deletion.

Parameters
pUnbalancedTransformer (*IscUnbalancedTransformer*) – The IscUnbalancedTransformer object for deletion.

Returns
True if successful.

Return type
bool

DeleteGroup(*pGroup*) → **bool**

Deletes a group by passing the IscGroup object for deletion.

Parameters
pGroup (*IscGroup*) – The IscGroup object for deletion.

Returns
True if successful.

Return type
bool

DeletePlugin(*pPlugin*) → **bool**

Deletes a plugin by passing the IscPlugin object for deletion.

Parameters
pPlugin (*IscPlugin*) – The IscPlugin object for deletion.

Returns
True if successful.

Return type
bool

DeleteBusBarSlack(*strBusbar:* ***str***) → **bool**

Deletes a slack busbar from the network busbar slack list. **It does not delete the busbar in the same way as DeleteBusbar(pBusbar)**, instead it uses the busbar name for deletion.

Parameters
strBusbar (***str***) – The slack busbar name.

Returns
True if successful.

Return type
bool

GetRatingIndex(*strName:* ***str***) → **int**

Returns an integer representing the rating set for a specified name.

Parameters
strName (***str***) – The specified name.

Returns
The rating set index, or -1 if no rating set with that name exists in the network.

Return type
int

GetBranchRatingName(*nIndex:* ***int***) → **str**

Returns the name representing the rating set identified by an index.

Parameters
nIndex (*int*) – The specified index.

Returns
The rating set name, or empty set if no rating set with that index exists in the network.

Return type
str

SetRatingName(nIndex: int, strName: str) → None

Sets the name of the rating set identified by an index to specified name. If the rating set name does not exist it will be created by the function.

Parameters

- **nIndex** (*int*) – The specified index.
- **strName** (*str*) – The specified name.

SetLimitsForOverloadChecks(dMaxVoltsPU: float, dMinVoltsPU: float, nRatingIndex: int, strDiagram: str) → None

Sets the limits for overload checking on diagrams.

Parameters

- **dMaxVoltsPU** (*float*) – The maximum voltage in per unit.
- **dMinVoltsPU** (*float*) – The minimum voltage in per unit.
- **nRatingIndex** (*int*) – The index of the rating set to be used for the thermal overload checks.
- **strDiagram** (*str*) – The name of the diagram that these limits will be applied to.

CreateLoadProfilePQActual(strName: str) → int

Returns the load profile UID representing a load profile which uses actual MW and MVAr values. No checking is made on duplicate profile names.

Parameters
strName (*str*) – The profile name.

Returns
The load profile UID, 0 if a load profile cannot be created.

Return type
int

CreateLoadProfilePQActualNoGraphics(strName: str)

Returns an IscLoadProfilePQActual object representing a load profile which uses actual MW and MVAr values. No checking is made on duplicate profile names.

Parameters
strName (*str*) – The profile name.

Returns
IscLoadProfilePQActual object.

Return type
IscLoadProfilePQActual

CreateGeneratorProfilePQActual(strName: str) → **int**

Returns the generator profile UID representing a generator profile which uses actual MW and MVAr values. No checking is made on duplicate profile names.

Parameters
strName (*str*) – The profile name.

Returns
The generator profile UID, 0 if a generator profile cannot be created.

Return type
int

CreateGeneratorProfilePQActualNoGraphics(strName: str)

Returns an IscGeneratorProfilePQActual object representing a generator profile which uses actual MW and MVAr values. No checking is made on duplicate profile names.

Parameters
strName (*str*) – The profile name.

Returns
IscGeneratorProfilePQActual object.

Return type
IscGeneratorProfilePQActual

CreateUMachineProfilePQActual(strName: str) → **int**

Returns the universal machine profile UID representing a universal machine profile which uses actual MW and MVAr values. No checking is made on duplicate profile names.

Parameters
strName (*str*) – The profile name.

Returns
The universal machine profile UID, 0 if a universal machine profile cannot be created.

Return type
int

CreateUMachineProfilePQActualNoGraphics(*strName: str*)

Returns an IscUMachineProfilePQActual object representing a universal machine profile which uses actual MW and MVAr values. No checking is made on duplicate profile names.

Parameters
strName (*str*) – The profile name.

Returns
IscUMachineProfilePQActual object.

Return type
IscUMachineProfilePQActual

CreateLoadProfilePQScale(*strName: str*) → **int**

Returns the load profile UID representing a load which scales the existing MW and MVAr values. No checking is made on duplicate profile names.

Parameters
strName (*str*) – The profile name.

Returns
The load profile UID, 0 if a generator profile cannot be created.

Return type
int

CreateLoadProfilePQScaleNoGraphics(*strName: str*)

Returns an IscLoadProfilePQScale object representing a load profile which scales the existing MW and MVAr values. No checking is made on duplicate profile names.

Parameters
strName (*str*) – The profile name.

Returns
IscLoadProfilePQScale object.

Return type
IscLoadProfilePQScale

CreateGeneratorProfilePQScale(*strName: str*) → **int**

Returns the generator profile UID representing a generator which scales the existing MW and MVAr values. No checking is made on duplicate profile names.

Parameters
strName (*str*) – The profile name.

Returns
The generator profile UID, 0 if a generator profile cannot be created.

Return type
int

CreateGeneratorProfilePQScaleNoGraphics(*strName: **str***)

Returns an IscGeneratorProfilePQScale object representing a generator profile which scales the existing MW and MVAr values. No checking is made on duplicate profile names.

Parameters
strName (***str***) – The profile name.

Returns
IscGeneratorProfilePQScale object.

Return type
IscGeneratorProfilePQScale

GetLoadProfilePQActuals()

Returns a dictionary of all IscLoadProfilePQActual objects in the network for actual load profiles. The keys are the profile UIDs and the values are the IscLoadProfilePQActual objects.

Returns
A dictionary of all IscLoadProfilePQActual objects in the network for actual load profiles.

Return type
dict(**int**,IscIscLoadProfilePQActual)

GetGeneratorProfilePQActuals()

Returns a dictionary of all IscGeneratorProfilePQActual objects in the network for actual generator profiles. The keys are the profile UIDs and the values are the IscGeneratorProfilePQActual objects.

Returns
A dictionary of all IscGeneratorProfilePQActual objects in the network for actual generator profiles.

Return type
dict(**int**,IscGeneratorProfilePQActual)

GetUMachineProfilePQActuals()

Returns a dictionary of all IscUMachineProfilePQActual objects in the network for actual universal machine profiles. The keys are the profile UIDs and the values are the IscUMachineProfilePQActual objects.

Returns
A dictionary of all IscUMachineProfilePQActual objects in the network for actual universal machine profiles.

Return type
: **dict**(**int**,IscUMachineProfilePQActual)

GetLoadProfilePQScales()

Returns a dictionary of all IscLoadProfilePQScale objects in the network for scaled load profiles. The keys are the profile UIDs and the values are the IscLoadProfilePQScale objects.

Returns
: A dictionary of all IscLoadProfilePQScale objects in the network for scaled load profiles.

Return type
: **dict**(**int**,IscLoadProfilePQScale)

GetGeneratorProfilePQScales()

Returns a dictionary of all IscGeneratorProfilePQScale objects in the network for scaled generator profiles. The keys are the profile UIDs and the values are the IscGeneratorProfilePQScale objects.

Returns
: A dictionary of all IscGeneratorProfilePQScale objects in the network for scaled generator profiles.

Return type
: **dict**(**int**,IscGeneratorProfilePQScale)

GetLoadProfilePQActual(*nUID*: ***int***)

GetLoadProfilePQActual(*strPythonName*: ***str***)

Returns an IscLoadProfilePQActual object for the actual MW/MVAr load profile with a specified UID or python name.

Parameters

- **nUID** (***int***) – The profile UID.
- **strPythonName** (***str***) – The profile name.

Returns
: IscLoadProfilePQActual object for the actual MW/MVAr load profile. Returns None if a profile cannot be found.

Return type
: IscLoadProfilePQActual

GetGeneratorProfilePQActual(*nUID*: ***int***)

GetGeneratorProfilePQActual(*strPythonName*: ***str***)

Returns an IscGeneratorProfilePQActual object for the actual MW/MVAr generator profile with a specified UID or python name.

Parameters

- **nUID** (*int*) – The profile UID.
- **strPythonName** (*str*) – The profile name.

Returns
IscGeneratorProfilePQActual object for the actual MW/MVAr generator profile. Returns None if a profile cannot be found.

Return type
IscGeneratorProfilePQActual

GetUMachineProfilePQActual(*nUID*: ***int***)

GetUMachineProfilePQActual(*strPythonName*: ***str***)

Returns an IscUMachineProfilePQActual object for the actual MW/MVAr universal machine profile with a specified UID or python name.

Parameters

- **nUID** (*int*) – The profile UID.
- **strPythonName** (*str*) – The profile name.

Returns
IscUMachineProfilePQActual object for the actual MW/MVAr universal machine profile. Returns None if a profile cannot be found.

Return type
IscUMachineProfilePQActual

GetLoadProfilePQScale(*nUID*: ***int***)

GetLoadProfilePQScale(*strPythonName*: ***str***)

Returns an IscLoadProfilePQScale object for the scaled MW/MVAr load profile with a specified UID or python name.

Parameters

- **nUID** (*int*) – The profile UID.
- **strPythonName** (*str*) – The profile name.

Returns
IscLoadProfilePQScale object for the scaled MW/MVAr load profile. Returns None if a profile cannot be found.

Return type
IscLoadProfilePQScale

GetGeneratorProfilePQScale(*nUID*: ***int***)

GetGeneratorProfilePQScale(*strPythonName: **str***)

Returns an IscGeneratorProfilePQScale object for the scaled MW/MVAr generator profile with a specified UID or python name.

Parameters

strPythonName (***str***) – The profile name.

Returns

IscGeneratorProfilePQScale object for the scaled MW/MVAr generator profile. Returns None if a profile cannot be found.

Return type

IscGeneratorProfilePQScale

DeleteLoadProfilePQActual(*pProfile*) → **bool**

Deletes the actual load profile from the network by passing an IscLoadProfilePQActual object.

Parameters

pProfile (*IscLoadProfilePQActual*) – The profile to be deleted.

Returns

True if successful.

Return type

bool

DeleteLoadProfilePQScale(*pProfile*) → **bool**

Deletes the scaled load profile from the network by passing an IscLoadProfilePQScale object.

Parameters

pProfile (*IscLoadProfilePQScale*) – The profile to be deleted.

Returns

True if successful.

Return type

bool

DeleteGeneratorProfilePQActual(*pProfile*) → **bool**

Deletes the actual generator profile from the network by passing an IscGeneratorProfilePQActual object.

Parameters

pProfile (*IscGeneratorProfilePQActual*) – The profile to be deleted.

Returns

True if successful.

Return type
bool

DeleteGeneratorProfilePQScale(pProfile) → **bool**

Deletes the scaled generator profile from the network by passing an IscGeneratorProfilePQScale object.

Parameters
pProfile (*IscGeneratorProfilePQScale*) – The profile to be deleted.

Returns
True if successful.

Return type
bool

DeleteUMachineProfilePQActual(pProfile) → **bool**

Deletes the actual universal machine profile from the network by passing an IscUMachineProfilePQActual object.

Parameters
pProfile (*IscUMachineProfilePQActual*) – The profile to be deleted.

Returns
True if successful.

Return type
bool

RunProfile() → **int**

Runs the profile study. Returns the number of profile categories which have been run.

Returns
The number of profile categories which have been run.

Return type
int

*GetDiagram(strName: **str**)*

Returns an IscDiagram instance for the diagram with given name contained in the network.

Parameters
strName (***str***) – The name of the diagram.

Returns
The diagram of the IPSA network.

Return type
IscDiagram

GetAllDiagrams()

Returns a list of IscDiagram objects for the network.

Returns
List of IscDiagram objects for the network.

Return type
list(*IscDiagram*)

GetAllDiagramsNames() → **List**[**str**]

Returns a list of the names of the diagrams for the network.

Returns
The names of the diagrams for the network.

Return type
list(**str**)

GetAnalysisLF()

Returns an IscAnalysisLF object which can be used to get and set the load flow analysis parameters.

Returns
IscAnalysisLF object.

Return type
IscAnalysisLF

SetResultsForTheseUIDs(*nUIDs: **int***) → **None**

This function restricts the number of results that are returned from the load flow calculation engine to Python in order to reduce the execution time. Call this function before DoLoadFlow() or DoSimpleLoadFlow().

Parameters
nUIDs (***int***) – The component UIDs.

DoLoadFlow(*bNoEngineLoad: **bool**, bDontUpdateData: **bool**, bUseDC: **bool** = False*) → **bool**

Performs a load flow calculation.

Parameters

- **bNoEngineLoad** (***bool***) – If False (default), loads the engine from the IPSA model before doing a load flow calculation. If True, skips the load from the IPSA model and uses whatever network is currently loaded in the engine.
- **bDontUpdateData** (***bool***) – If False (default), allows the load flow results being written back to the network model data (e.g. Busbar voltages and angles). If True, skips this stage, so the network model

remains the same as it was loaded. **Note that calling the function with no arguments is allowed and works as if it has been called with bNoEngineLoad and bDontUpdateData set to False.**

- **bUseDC** (*bool*) – Tells the user that they can run a DC load flow instead of a normal load flow. If True, the program will run a DC load flow instead of an AC load flow. Default value of bUseDC is False.

Returns
True if the load flow converges, False on a non-convergence.

Return type
bool

DoSimpleLoadFlow()

Performs a load flow calculation without prompting the user to confirm analysis options. Identical to the DoLoadFlow(False, False) call with no user interaction.

Returns
True if the load flow converges, False on a non-convergence.

Return type
bool

GetAnalysisDCLF()

Returns an IscAnalysisDCLF object which can be used to get and set the DC load flow analysis parameters.

Returns
IscAnalysisDCLF object.

Return type
IscAnalysisDCLF

DoDCLoadFlow()

Performs a DC load flow calculation while assuming you do not want to update the engines or results.

Returns
True if the load flow converges, False on a non-convergence.

Return type
bool

SetBranchStatus(*nUID: **int**, nStatus: **int***) → **None**

Changes the status of the branch or transformer UID in the calculation engine. This is a convenience function which can be used when performance is important and the branch status does not need to be stored with the network. **Note: If the nUID is not a branch or transformer UID, it does nothing!**

Parameters

- **nUID** (*int*) – The branch or transformer UID.
- **nStatus** (*int*) – The status.

SetLoadStatus(*nUID: **int**, nStatus: **int***) → **None**

Changes the status of the load UID in the calculation engine. This is a convenience function which can be used when performance is important and the load status does not need to be stored with the network.

Parameters

- **nUID** (*int*) – The load UID.
- **nStatus** (*int*) – The status.

SetLoadPower(*nUID: **int**, dMW: **float**, dMVAr: **float***) → **None**

Changes the power of the load UID in the calculation engine. This is a convenience function which can be used when performance is important and the load power does not need to be stored with the network.

Parameters

- **nUID** (*int*) – The load UID.
- **dMW** (*float*) – The MW power.
- **dMVAr** (*float*) – The MVAr power.

SetGeneratorStatus(*nUID: **int**, nStatus: **int***) → **None**

Changes the status of the generator UID in the calculation engine. This is a convenience function which can be used when performance is important and the generator status does not need to be stored with the network.

Parameters

- **nUID** (*int*) – The generator UID.
- **nStatus** (*int*) – The status.

SetGeneratorPower(*nUID: **int**, dMW: **float**, dMVAr: **float***) → **None**

Changes the power of the generator UID in the calculation engine. This is a convenience function which can be used when performance is important and the generator power does not need to be stored with the network.

Parameters

- **nUID** (*int*) – The generator UID.
- **dMW** (*float*) – The MW power.
- **dMVAr** (*float*) – The MVAr power.

GetLoadFlowMessage() → **str**

Returns the last load flow engine message.

Returns
The last load flow engine message.

Return type
str

SetEngineMessageSuppression(*nLevel: int*) → **None**

Sets the verbosity of the load flow messages that are generated in the IPSA progress window. This can provide a speed improvement for complex scripts

- 0 = Displays all messages
- 1 = Shows only error messages
- 2 = Shows no engine error messages

Parameters
nLevel (*int*) – The verbosity of the load flow messages.

GetLFSummaryResults() → **None**

Call this function to obtain the load flow summary results.

GetHighestBusbarVoltagePU() → **float**

Returns the highest busbar voltage in per unit.

Returns
The highest busbar voltage in per unit.

Return type
float

GetLowestBusbarVoltagePU() → **float**

Returns the lowest busbar voltage in per unit. GetLFSummaryResults()must be called first.

Returns
The lowest busbar voltage in per unit.

Return type
float

GetTotalGenerationOutputMW() → **float**

Returns the total network generation real power, excluding slack generators, in MW. GetLFSummaryResults() must be called first.

Returns
The total network generation real power, excluding slack generators, in MW.

Return type
float

GetTotalGenerationOutputMVAr() → **float**

Returns the total network generation reactive power, excluding slack generators, in MVAr. GetLFSummaryResults() must be called first.

Returns
The total network generation reactive power, excluding slack generators, in MVAr.

Return type
float

GetTotalLoadInputMW() → **float**

Returns the total network load real power in MW. GetLFSummaryResults() must be called first.

Returns
The total network load real power in MW.

Return type
float

GetTotalLoadInputMVAr() → **float**

Returns the total network load reactive power in MVAr. GetLFSummaryResults() must be called first.

Returns
The total network load reactive power in MVAr.

Return type
float

GetTotalInductionInputMW() → **float**

Returns the total network induction motor real power in MW. GetLFSummaryResults() must be called first.

Returns
The total network induction motor real power in MW.

Return type
float

GetTotalInductionInputMVAr() → **float**

Returns the total network induction motor load in MVAr. GetLFSummaryResults() must be called first.

Returns
The total network induction motor load in MVAr.

Return type
float

GetTotalUniMachineOutputMW() → **float**

Returns the total network universal machine generation real power in MW. GetLFSummaryResults() must be called first.

Returns

The total network universal machine generation real power in MW.

Return type

float

GetTotalUniMachineOutputMVAr() → **float**

Returns the total network universal machine generation reactive power in MVAr. GetLFSummaryResults() must be called first.

Returns

The total network universal machine generation reactive power in MVAr.

Return type

float

GetSlackOutputMW() → **float**

Returns the total network slack generation real power in MW. GetLFSummaryResults() must be called first.

Returns

The total network slack generation real power in MW.

Return type

float

GetSlackOutputMVAr() → **float**

Returns the total network slack generation reactive power in MVAr. GetLFSummaryResults() must be called first.

Returns

The total network slack generation reactive power in MVAr.

Return type

float

GetNumberOutsideLimits() → **int**

Returns the number of busbars outside voltage limits plus the number of overloaded branches and transformers.

Returns

The number of busbars outside voltage limits plus the number of overloaded branches and transformers.

Return type

int

GetOutsideLimitText() → **str**

Returns a string detailing the busbar, branch or transformer with the most excessive overload/overvoltage in percentage terms. GetNumberOutsideLimits() must be called first. The name returned is the Python name of the component, e.g. Busbar1.Busbar2.Transformer

Returns
A string detailing the busbar, branch or transformer with the most excessive overload/overvoltage in percentage terms.

Return type
str

AreLFLimitsIdentical() → **bool**

Returns True if the LF limits are identical.

Returns
True if the LF limits are identical.

Return type
bool

SaveLFState() → **int**

Saves the current LF state and returns a state handle to restore it with.

Returns
State handle to restore the current LF state.

Return type
int

*RestoreLFState(nStateIndex: **int**)* → **bool**

Restore the LF state. This function can fail if the number of items in a network is different from when the state was saved, which can happen in a subtle way if zero impedance branches are switched in or out.

Parameters
nStateIndex (*int*) – The state index.

Returns
True if the restore operation succeeded.

Return type
bool

DeleteAllLFStates() → **None**

Delete all LF saved states.

GetBusbarsOutsideLimits() → **Dict**[**int**, **bool**]

Returns a dictionary of busbar UIDs that are outside voltage limits for the previous load flow study.

Returns
A dictionary of busbar UIDs that are outside voltage limits for the previous load flow study.

Return type
dict(int,bool)

GetBranchesOutsideLimits() → Dict[int, bool]

Returns a dictionary of branch UIDs that are above their ratings for the previous load flow study.

Returns
A dictionary of branch UIDs that are above their ratings for the previous load flow study.

Return type
dict(int,bool)

GetTransformersOutsideLimits() → Dict[int, bool]

Returns a dictionary of transformer UIDs that are above their ratings for the previous load flow study.

Returns
A dictionary of transformer UIDs that are above their ratings for the previous load flow study.

Return type
dict(int,bool)

RunArcFlashForBusbar(nBusbarUID: int, dBusFaultCurrentkA: float, dOperatingTimeSec: float) → bool

Performs an ArcFlash calculation for a single busbar using the fault current in kA and the operating time. The default reduction for comparison is 15% less for the current and 2.5x the arc duration given.

Parameters

- **nBusbarUID** (*int*) – The UID of the selected busbar.
- **dBusFaultCurrentkA** (*float*) – The fault current in kA.
- **dOperatingTimeSec** (*float*) – The operating time in seconds.

Returns
Returns True if it is successful.

Return type
bool

RunTotalArcFlash(*bRunIPSAFaultLevel:* ***bool***, *dOperatingTimeSec:* ***float***, *dReducedOperatingTimeSec:* ***float***) → **List**[**Dict**[**int**, **bool**]]

Runs a thorough arc flash calculation for the whole network. **Note that here either the analysis class default for the fault current calculation is used or IPSA can run a fault level to calculate the fault current at each busbar.** Returns a list of pairs that map the UID to a boolean of whether the code ran correctly or not.

Parameters

- **bRunIPSAFaultLevel** (***bool***) – Variable denoting whether it runs the IPSA fault lever before the arc flash.
- **dOperatingTimeSec** (***float***) – The operating time in seconds.
- **dReducedOperatingTimeSec** (***float***) – The reduced operating time in seconds.

Returns
A a list of pairs that map the UID to a boolean of whether the code ran correctly or not.

Return type
list(**dict**(**int**,**bool**))

DoFlatStart(*bSetBuses:* ***bool***, *bSetTransformerTaps:* ***bool***, *bSetIMSlips:* ***bool***) → **None**

Runs a flatstart preparation for load flow depending on whether the user wants to flat start the busbar voltages, transformer tap positions, induction machine rotor slips or a combination of all 3.

Parameters

- **bSetBuses** (***bool***) – Enabling flat start for the busbar voltages.
- **bSetTransformerTaps** (***bool***) – Enabling flat start for the transformer tap positions.
- **bSetIMSlips** (***bool***) – Enabling flat start for the induction machine rotor slips.

GetAnalysisFL()

Returns an IscAnlaysisFL object which can be used to get and set the fault level analysis parameters.

Returns
IscAnlaysisFL object.

Return type

IscAnlaysisFL

DoFaultLevel() → **bool**

Performs a fault level calculation.

Returns

True if successful.

Return type

bool

DoIECFaultLevel() → **bool**

Performs an IEC 60909 fault calculation.

Returns

True if successful.

Return type

bool

GetAnalysisHM()

Returns an IscAnlaysisHM object which can be used to get and set the load flow analysis parameters.

Returns

IscAnlaysisHM object.

Return type

IscAnlaysisHM

DoHarmPenetration() → **bool**

Performs a harmonic penetration calculation.

Returns

True if successful.

Return type

bool

DoHarmSensitivity() → **bool**

Performs a harmonic voltage sensitivity calculation.

Returns

True if successful.

Return type

bool

DoStorageFlip(*lGeneratorsUID*: ***List[int]***) → **None**

Flips the storage of all defined Energy Storage units in the given list of UIDs.

Parameters

lGeneratorsUID (*list(int)*) – The given list of generators UIDs.

DoSingleStorageFlip(*nGeneratorUID: **int***) → **None**

Flips the storage of the Energy Storage unit defined by its UID.

Parameters

nGeneratorUID (*int*) – The generator UID.

DoGlobalStorageFlip(*bFlipsImports: **bool**, bFlipExports: **bool***) → **None**

Flips all the storage units defined in the network depending on whether you want to flip imports to exports or vice versa.

Parameters

- **bFlipsImports** (*bool*) – Variable denoting whether you want to flip imports to exports.
- **bFlipExports** (*bool*) – Variable denoting whether you want to flip exports to imports.

RunContingency(*nUID: **int**, bUseProfiles: **bool***) → **None**

Performs the contingency study identified by the integer UID.

Parameters

- **nUID** (*int*) – The contingency study UID.
- **bUseProfiles** (*bool*) – If False then the contingency study is performed using the standard load and generator data. If True then the contingency study is performed using load and generator profiles assigned in the network. In this instance the switching operation is performed first followed by a load flow calculation for all of the profile categories.

CreateContingency(*nDepth: **int**, bExtendToBreakers: **bool***) → **int**

Creates a new contingency study and returns the UID of the study created. The depth of the study is configured as follows:

- 1 = N - 1
- 2 = N - 2
- 3 = N - 3
- 4 = N - 1 - 1

Parameters

- **nDepth** (*int*) – The depth of the study.
- **bExtendToBreakers** (*bool*) – If False then individual branches and transfers are switched out during the study. If True then the nearest

circuit breakers are switched out allowing multiple components to be switched for each study.

Returns

The UID of the contingency created.

Return type

int

CreateSpecificContingency(*nDepth: **int**, bExtendToBreakers: **bool**, lBusbarsRequired*) → **int**

Will design and create a specific contingency of given depth with only the busbars defined by the given list.

Parameters

- **nDepth** (***int***) – The depth of the study.
- **bExtendToBreakers** (***bool***) – If False then individual branches and transfers are switched out during the study. If True then the nearest circuit breakers are switched out allowing multiple components to be switched for each study.
- **lBusbarsRequired** (***list***(*IscBusbar*)) – The specified list of busbars.

Returns

The UID of the contingency created.

Return type

int

GetStudies(*nReportType: **int***) → **List**[**str**]

Returns a list of strings containing the individual automation or contingency study titles.

Automation studies:

- 100 = All studies in the order run
- 101 = All solved studies in the order run
- 102 = All solved studies listed by severity of overload
- 103 = All solved studies listed by the number of items exceeding limits
- 104 = All studies that failed to solve

Contingency studies:

- 120 = All studies in the order run
- 121 = All solved studies in the order run
- 122 = All solved studies listed by severity of overload

- 123 = All solved studies listed by the number of items exceeding limits
- 124 = All studies that failed to solve

Parameters
nReportType (*int*) – The index denoting an automation or a contingency study.

Returns
The individual automation or contingency study titles.

Return type
list(**str**)

GetStudyRowTitles(*nReportType: **int***) → **str**

Returns a string in html format for the table header row associated with the automation or contingency results.

Automation studies:

- 100 = All studies in the order run
- 101 = All solved studies in the order run
- 102 = All solved studies listed by severity of overload
- 103 = All solved studies listed by the number of items exceeding limits
- 104 = All studies that failed to solve

Contingency studies:

- 120 = All studies in the order run
- 121 = All solved studies in the order run
- 122 = All solved studies listed by severity of overload
- 123 = All solved studies listed by the number of items exceeding limits
- 124 = All studies that failed to solve

Parameters
nReportType (*int*) – The index denoting an automation or a contingency study.

Returns
String in html format.

Return type
str

GetStudyRowOutput(*nReportType:* ***int***, *strStudyTitle:* ***str***) → **str**

Returns a string in html format for the table rows associated with the specified automation or contingency study.

Automation studies:

- 100 = All studies in the order run
- 101 = All solved studies in the order run
- 102 = All solved studies listed by severity of overload
- 103 = All solved studies listed by the number of items exceeding limits
- 104 = All studies that failed to solve

Contingency studies:

- 120 = All studies in the order run
- 121 = All solved studies in the order run
- 122 = All solved studies listed by severity of overload
- 123 = All solved studies listed by the number of items exceeding limits
- 124 = All studies that failed to solve

Parameters

- **nReportType** (*int*) – The index denoting an automation or a contingency study.
- **strStudyTitle** (*str*) – The specified automation or contingency study.

Returns

String in html format.

Return type

str

GetStudyIDs(*nReportType:* ***int***) → **List[int]**

Returns a list containing the individual automation or contingency study IDs.

Automation studies:

- 100 = All studies in the order run
- 101 = All solved studies in the order run
- 102 = All solved studies listed by severity of overload
- 103 = All solved studies listed by the number of items exceeding limits
- 104 = All studies that failed to solve

Contingency studies:

- 120 = All studies in the order run
- 121 = All solved studies in the order run
- 122 = All solved studies listed by severity of overload
- 123 = All solved studies listed by the number of items exceeding limits
- 124 = All studies that failed to solve

Parameters
nReportType (*int*) – The index denoting an automation or a contingency study.

Returns
The individual automation or contingency study IDs.

Return type
list(**int**)

GetContingencyStudyItemResults(*nStudyID: int*) → **Dict**[**int**, **int**]

Returns a dict of the component UIDs to the result ID for each component for the study with the given ID. The result IDs can be understood as followed:

- 1 = Busbar over voltage (balanced or unbalanced)
- 2 = Busbar under voltage (balanced or unbalanced)
- 3 = Branch over rating (balanced or unbalanced)
- 4 = Transformer over rating (2- or 3- winding, or unbalanced)
- 0 = Otherwise

Parameters
nStudyID (*int*) – The contingency study ID.

Returns
The map of the component UIDs to the result IDs for the contingency study ID.

Return type
dict[**int**, **int**]

GetAutomationStudyItemResults(*nStudyID: int*) → **Dict**[**int**, **int**]

Returns a dict of the component UIDs to the result ID for each component for the study with the given ID. The result IDs can be understood as followed:

- 1 = Busbar over voltage (balanced or unbalanced)
- 2 = Busbar under voltage (balanced or unbalanced)

- 3 = Branch over rating (balanced or unbalanced)
- 4 = Transformer over rating (2- or 3- winding, or unbalanced)
- 0 = Otherwise

Parameters
nStudyID (*int*) – The automation study ID.

Returns
The map of the component UIDs to the result IDs for the automation study ID.

Return type
dict[int, int]

GetStudyProfileIndex(nStudyID: int) → **int**

Returns the profile category index associated with the contingency or automation study. This is used to identify which profile category is associated with the study ID.

Parameters
nStudyID (*int*) – The study ID.

Returns
The profile category index associated with the contingency or automation study.

Return type
int

GetStudyItemsSwitchedOutUIDs(nStudyID: int) → **List[int]**

Returns a list of integers containing the component UIDs for switched out components in contingency study ID.

Parameters
nStudyID (*int*) – The contingency study ID.

Returns
The component UIDs for switched out components in contingency study ID.

Return type
list(int)

GetContingencyStudyResultMagnitude(nStudyID: int, nResultID: int) → **float**

Returns the result magnitude for the result ID in contingency study ID. The nResultID is obtained from the GetContingencyStudyItemResults function. For busbars the return value is the per unit busbar voltage. For branches and transformers the return value is the largest power flow in MVA.

Parameters

- **nStudyID** (*int*) – The contingency study ID.
- **nResultID** (*int*) – The result ID.

Returns

The result magnitude for the result ID in contingency study ID.

Return type

float

GetContingencyStudyDynamicallyOverloadedUIDs(*nStudyID: **int***) → **List**[**int**]

Returns a list of integers which represent lines which are overloaded due to the action of a dynamic rating plugin. Dynamic rating plugins can be used to model the thermal response of OHLs, transformers and cables and provide ratings which are based on these models. The normal IPSA rating of a component is overridden if it has a dynamic rating plugin applied. In this case this function returns the UIDs of all such overloaded components in contingency study ID.

Parameters

nStudyID (*int*) – The contingency study ID.

Returns

The lines which are overloaded due to the action of a dynamic rating plugin.

Return type

list(**int**)

GetContingencyBranchRatingIndex() → **int**

Returns the IPSA rating index of the rating set used during the contingency study.

Returns

The IPSA rating index.

Return type

int

RunReliability() → **bool**

Performs the reliability study on the current network.

Returns

True if successful.

Return type

bool

GetReliabilityCI() → **float**

Returns the customer interruptions (CI) for the full network.

Returns
The customer interruptions (CI) for the full network.

Return type
float

GetReliabilityCML() → float

Returns the customer minutes lost (CMLs) for the full network.

Returns
The customer minutes lost (CMLs) for the full network.

Return type
float

GetReliabilitySAIFI() → float

Returns the system average interruption frequency index (SAIFI) for the full network.

Returns
The system average interruption frequency index (SAIFI) for the full network.

Return type
float

GetReliabilityASIFI() → float

Returns the average service interruption frequency index (ASIFI) for the full network.

Returns
The average service interruption frequency index (ASIFI) for the full network.

Return type
float

GetReliabilitySAIDI() → float

Returns the system average interruption duration index (SAIDI) for the full network.

Returns
The system average interruption duration index (SAIDI) for the full network.

Return type
float

GetReliabilityCAIDI() → float

Returns the customer average interruption duration index (CAIDI) for the full network.

Returns
The customer average interruption duration index (CAIDI) for the full network.

Return type
float

GetReliabilityASIDI() → **float**

Returns the average system interruption duration index (ASIDI) for the full network.

Returns
The average system interruption duration index (ASIDI) for the full network.

Return type
float

GetReliabilityASAI() → **float**

Returns the average service availability index (ASAI) for the full network.

Returns
The average service availability index (ASAI) for the full network.

Return type
float

GetReliabilityASUI() → **float**

Returns the average service unavailability index (ASUI) for the full network.

Returns
The average service unavailability index (ASUI) for the full network.

Return type
float

GetBusbarsWithArcFlashResults() → **List**[**int**]

Returns a list of busbar UIDs which have arc flash results. This is then used to get arc flash results for individual busbars.

Returns
Busbar UIDs which have arc flash results.

Return type
list(**int**)

GetArcFlashCSV(*nBusbarUID:* ***int****, bUseLegacyStandard:* ***bool***) → **str**

Creates a CSV result for a given busbar arcflash calculation and uses the 2018 standard if bUseLegacyStandard is set to False.

Parameters

- **nBusbarUID** (*int*) – The busbar UID.
- **bUseLegacyStandard** (*bool*) – Variable denoting whether the legacy standard used.

Returns
The CSV result for a given busbar arcflash calculation.

Return type
str

GetTotalArcFlashCSV() → str

Returns total CSV formatted function for ArcFlash results from all busbars.

Returns
The total CSV formatted function for ArcFlash results from all busbars.

Return type
str

GetArcFlashReportText(nUID: int) → str

Returns a string containing the arc flash result for the busbar identified by the UID.

Parameters
nUID (*int*) – The busbar ID.

Returns
The average service unavailability index (ASUI) for the full network.

Return type
str

GetAnalysisAF()

Returns an IscAnalysisAF object which can be used to get and set the ArcFlash analysis parameters.

Returns
IscAnlaysisAF object.

Return type
IscAnlaysisAF

SetBusbarOverloadLimits(dBusVoltHighPU: float, dBusVoltlowPU: float) → None

Sets the network global high and low limits for busbar overloads.

Parameters

- **dBusVoltHighPU** (*float*) – The high limit for busbar overloads in per unit.
- **dBusVoltlowPU** (*float*) – The low limit for busbar overloads in per unit.

SetBranchOverloadLimits(*dBranchRatingHighPC:* ***float***, *dBranchRatingLowPC:* ***float***, *nRatingIndex:* ***int***) → **None**

Sets the network global percentage ratings for branches with a given rating index that is lifted from IscBranch (i.e., Standard, Summer, Winter, Short).

Parameters

- **dBranchRatingHighPC** (***float***) – The high network global percentage rating limit.
- **dBranchRatingLowPC** (***float***) – The low network global percentage rating limit.
- **nRatingIndex** (***int***) – The given rating index.

Profile Class Functions

The functions for the 5 profile classes (*IscLoadProfilePQActual*, *IscLoadProfilePQScale*, *IscGeneratorProfilePQActual*, *IscGeneratorProfilePQScale*, *IscUMachineProfilePQActual*) are as follows:

class *ipsa.****Isc__ProfilePQ__***

Provides access to the actual given profile class.

SetName(*strName:* ***str***) → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

SetCategoryNames(*dictCategories:* ***Dict[int, str]***) → **None**

Sets up the profile categories for the profile instance. The dictionary should comprise a set of integer keys and string values. The string values are used as the individual category labels whilst the integer keys are only used internally. It is recommended that the keys are numbered sequentially starting from 0.

For example, passing the following dictionary would add 3 categories to the profile with the strings as the categories:

categories = {0: “00:00”, 1: “00:30”, 2: “01:00”}

Parameters

dictCategories (***dict(int,str)***) – The profile categories for the profile instance.

GetCategoryNames() → **Dict**[**int**, **str**]

Returns the profile categories for the profile instance. The string values are used as the individual category labels whilst the integer keys are only used internally.

Returns
The profile categories for the profile instance.

Return type
dict(**int**,**str**)

SetPMW(*dictCategoryToMW: **Dict[int, float]***) → **None**

Assigns MW values to the profile categories. The dictionary should comprise a set of integer keys and float values. The float values are the MW data values whilst the integer keys should be identical to those being used when defining the categories. For scaling profiles the values are the per unit scaling values. For example, passing the following dictionary would set the MW data:

dictCategoryToMW = {0: 1.23, 1: 3.73, 2: 5.67}

Parameters
dictCategoryToMW (***dict(int,float)***) – MW or pu values to the profile categories.

GetPMW() → **Dict**[**int**, **float**]

Returns the MW values assigned to the profile categories. The float values are the MW data values whilst the integer keys should be identical to those used defining the categories. For scaling profiles the values are the per unit scaling values.

Returns
MW or pu values to the profile categories.

Return type
dict(**int**,**float**)

SetQMVAr(*dictCategoryToMVAr: **Dict[int, float]***) → **None**

Assigns MVAr values to the profile categories. The dictionary should comprise a set of integer keys and float values. The float values are the MVAr data values whilst the integer keys should be identical to those being used when defining the categories. For scaling profiles the values are the per unit scaling values. For example, passing the following dictionary would set the MVAr data:

dictCategoryToMVAr = {0: 1.23, 1: 3.73, 2: 5.67}

Parameters
dictCategoryToMVAr (***dict(int,float)***) – MVAr or pu values to the profile categories.

GetQMVAr(*dictCategoryToMVAr: **Dict[int, float]***) → **None**

Returns the MVAr values assigned to the profile categories. The float values are the MVAr data values whilst the integer keys should be identical to those used defining the categories. For scaling profiles the values are the per unit scaling values.

Returns

MVAr or pu values to the profile categories.

Return type

dict(**int**,**float**)

1.7 IscAnalysis

There are separate classes for each analysis type, e.g. load flow, fault level and harmonic analysis. The *IscNetwork* class provides functions to obtain an *IscAnalysis* instance for each analysis type, for example *GetAnalysisLF()* returns an *IscAnalysisLF* object. Motor start analysis options are provided under the fault level analysis class.

1.7.1 Analysis classes

IscAnalysisLF

Field Values

Table 1: **IscAnalysisLF Field Values**

Type	Field Name	Description
Float	Convergence	Accuracy for load flow solution.
Integer	MaxIterations	Maximum number of iterations to run the load flow.
Float	UndervoltagePU	Lower voltage limit for busbars (reporting only).
Float	OvervoltagePU	Upper voltage limit for busbars (reporting only).
Integer	LockTaps	Lock all transformer taps based on the following settings: • 0 = Do not lock taps • 1 = Lock taps during outage analysis only • 2 = Lock taps

continues on next page

Table 1 – continued from previous page

Type	Field Name	Description
Boolean	NoPhaseShift	Do not apply phase shifts to load flow. • False = Use phase shifting in load flow • True = No phase shifting
Integer	TapOscIterStart	Starts counting the iteration number of transformer tap oscillations.
Integer	TapOscSuccessive	Number of successive iterations of tap oscillation.
Integer	TapOscLimit	Tap oscillation limit after which transformer taps are locked.
Integer	TapOscIterEnd	Stops counting the iteration number of transformer tap oscillations.
Integer	FillkARatings	Automatically complete kA rating fields for lines.
Boolean	UseLoadScaling	Enable scaling of loads in the LF calculation.
Float	RealLoadScale	Per unit factor used to scale all real loads (default = 1.0).
Float	ReactiveLoadScale	Per unit factor used to scale all reactive loads (default = 1.0).
Boolean	CheckProtection	Set *True* to check protection devices after load flow.
Boolean	UseLegacyPhiftCheck	Enables or disables the legacy load flow engine code.
Boolean	DisplayOptionDialog	Setting this field to *True* causes the load flow options dialog to be displayed whenever a load flow is required.
Float	FeederSlackVoltagePU	Sets the busbar voltage for the slack busbar when performing a feeder load flow.
Integer	FeederSetTarget	Set to 1 to specify a target power when performing a feeder load flow.
Boolean	SingleTapMovement	Setting this item to *True* forces all tap changes to be moved a maximum of one step in each load flow iteration.
Boolean	SlowTapMovement	Setting this item to *True* forces all tap changes to be adjusted every fourth load flow iteration instead of every iteration.
Integer	WhichImpedance	The default setting is 0 which will use the normal resistance value for branches when performing calculations. Set this value to 1 to use the minimum resistance value for branches when performing calculations.

continues on next page

Table 1 – continued from previous page

Type	Field Name	Description
Integer	IslandMethod	The default setting is 0 where any island with a slack will have load flow results, assuming the network converges. Setting this value to 1 means any island with a slack busbar must also have a voltage-controlling generator on that busbar in order to have load flow results (again assuming that the network converges).
Boolean	AutoSelectSlacks	Set *True* to automatically select slack busbars in islands where the user has not manually specified a slack busbar.
Boolean	InitFlatStart	Sets *True* to perform flat starts for all load flows until this parameter is reset to *False*.
Boolean	FSSetBusbarVoltages	Set *True* to reset all busbar voltages during a flat start.
Boolean	FSIgnoreVoltageControlSettings	Set *True* to ignore transformer voltage control settings during a flat start.
Float	FSVoltageMagnitudePU	Sets the busbar voltage magnitude in per unit during a flat start.
Float	FSVoltageAngleDeg	Sets the busbar voltage angle in degrees during a flat start.
Boolean	FSSetTransformerTaps	Set *True* to reset the transformer taps during a flat start.
Boolean	FSIgnoreFixedTaps	Set *True* to ignore fixed tap positions during a flat start.
Integer	FSNominalTaps	Set to 1 to specify that the transformer nominal tap position will be used during a flat start.
Float	FSTapStartPC	Sets the tap position of transformers in percentage during a flat start.
Boolean	FSSetInductionMachineSlips	Set *True* to force the induction motor slips to a specified value during a flat start.
Float	FSSlipPC	Sets the induction motor slips in percentage during a flat start.
Integer	ProfileUse	• 0 = Do not apply load and generator profiles • 1 = Apply load and generator profile category specified by the *ProfileLoadCategory* field

continues on next page

Table 1 – continued from previous page

Type	Field Name	Description
String	ProfileLoadCategory	Pass a string representing the required load/generator profile category name to be used for the next load flow.
Float	ProfileTimeSliceHrs	Pass a float representing the number of hours in each profile category.

IscAnalysisLF Class

class** ipsa.**IscAnalysisLF

Analysis class for the load flow analysis.

***GetIValue**(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The integer value for the field.

Return type
int

***GetDValue**(nFieldIndex: **int**)* → **float**

Returns a float value for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The float value for the field.

Return type
float

***GetSValue**(nFieldIndex: **int**)* → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The string value for the field.

Return type
str

GetBValue(*nFieldIndex*: ***int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters

nFieldIndex (***int***) – The given enumerated field.

Returns

The boolean value for the field.

Return type

bool

SetIValue(*nFieldIndex*: ***int***, *nValue*: ***int***) → **bool**

Sets the integer value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **nValue** (***int***) – The integer value that will be set.

Returns

True if successful.

Return type

bool

SetDValue(*nFieldIndex*: ***int***, *dValue*: ***float***) → **bool**

Sets the float value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **dValue** (***float***) – The float value that will be set.

Returns

True if successful.

Return type

bool

SetSValue(*nFieldIndex*: ***int***, *strValue*: ***str***) → **bool**

Sets the string value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **strValue** (***str***) – The string value that will be set.

Returns

True if successful.

Return type
bool

SetBValue(*nFieldIndex: **int**, bValue: **bool***) → **bool**

Sets the integer value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **bValue** (***bool***) – The boolean value that will be set.

Returns
True if successful.

Return type
bool

GetFieldType(*nFieldIndex: **int***) → **str**

Returns the field type as a string for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The field type.

Return type
str

GetFieldName(*nFieldIndex: **int***) → **str**

Returns the field name as a string for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The field name.

Return type
str

IscAnalysisFL

Field Values

Table 2: **IscAnalysisFL Field Values**

Type	Field Name	Description
Integer	FaultEngine	Sets the fault level engine to either the standard Ipsa method or the IEC60909 method. Should be one of: • 0 = Standard Ipsa method • 1 = IEC60909 method
Integer	FaultStudyType	Specifies the type of fault study. Should be one of: • *FaultAllBusbars* • *FaultSelectedBusbars* • *FaultSingleBusbar* • *FaultLine* • *FaultTransformer* • *FaultWaveformBus* • *FaultWaveformBranch* • *FaultBreakerDuty* • *FaultMotorStart*
Float	FaultTime	Time of fault in seconds.
Float	FaultResistance	Fault resistance in per unit on the system base.
Float	FaultReactance	Fault reactance in per unit on the system base.
Integer	FaultEngineType	Type of fault to be applied. Should be one of: • LineGround • LineLine • LineLineGround • LineLineLine
Integer	FaultEngineResult-Type	Type of fault result obtained. Should be one of: • SymRMS • AsymPeak • AsymRMS • BusWave • BranchWave
Integer	MaxFaultIterations	Maximum number of iterations to run the fault level.
Boolean	FaultFlatStart	Sets voltages at 1 p.u. before calculating fault levels, returns *True* if successful.

continues on next page

Table 2 – continued from previous page

Type	Field Name	Description
Integer	UseSaturated-Impedances	Uses generator saturated impedances in fault calculation.
Integer	AssumeAVRAction	Assumes generator impedances decay to transient rather than steady state values.
Integer	SMSaliency	Sets the synchronous machine salience to either the given value or (Xq = Xd). Should be one of: • 0 = As given: The direct axis and quadrature axis parameters entered for each generator will be used in the fault calculations. • 1 = (Xq = Xd): Steady-state quadrature axis parameters are assumed to be the same as direct axis parameters for all generators.
Integer	XRCalcMethod	Sets the X/R calculation method to either DC decay or Driving Point. Should be one of: • 0 = DC decay: The DC component decays with time, following a single exponential curve, and the X/R ratio will change. **(Note: Under this option the calculation takes the DC component at the time of fault, and the DC component at the specified time after the fault, and then fits a single exponential to these values.)** • 1 = Driving point: The X/R ratio is calculated at the time the fault occurs and does not change.
Integer	XRSMEnhanced	Set to 0 to use the Ipsa 2.3.2 method of calculating the DC decay. Set to 1 to use the Ipsa 2.4.2 enhanced method of calculating the DC decay.
Boolean	Fault-Use2ndHarmonic	If selected then second harmonic fault level will be included in any peak fault calculation for line-to-line faults.
Integer	SingleBusToFault	Busbar UID to apply fault on.
Integer	BranchToFault	Branch UID to apply fault on.
Float	DistanceAlong-Branch	Distance along branch to apply fault on. This is a per unit value with zero representing the “From” end of the branch and 1.0 representing the “To” end of the branch.

continues on next page

Table 2 – continued from previous page

Type	Field Name	Description
Boolean	FaultUseCDPs	Switch to decide whether the fault engine will include the impact of converter driven plants.
Integer	FaultCDPStudy-Mode	The calculation method for CDPs. Currently only the simple method (0) works for PyIPSA. To input the data for the advanced method (1) you would need to open the IPSA UI.
Integer	FaultCDPInterp-Method	Chooses the interpolation method for the universal machines that represent the CDPs (in the advanced method): • 0 = Machine specific settings • 1 = Globally use linear interpolation • 2 = Globally use cubic interpolation
Float	IEC909DefaultPhase/	Specifies the default synchronous machine power factor. *IEC909UseDefaultPF* should be set to *True* to use this value.
Integer	IEC909Method	Sets the method used to determine the X/R ratio as defined by: • 1 = IEC 60909 Method A • 2 = IEC 60909 Method B • 3 = IEC 60909 Method C
Boolean	IEC909IgnoreImpeda	If set to *True* then IEC60909 impedance correction factors will not be applied to generators and power station transformers.
Integer	IEC909VoltageCorrec	One of the following IEC60909 voltage level based correction factors to be applied to the pre-fault voltage at the faulted busbar: • 1 = Ignore • 2 = Cmax (LV + 6%) • 3 = Cmax (LV + 10%) • 4 = Cmin
Boolean	IEC909UseDefaultPF	Set to *True* to use the synchronous machine default power factor.
Boolean	IEC909NearTo	Setting to *True* causes all faults as assumed to be “Near-To”. If it is not selected then the analysis will neglect any decay effects.
Integer	IEC909TFRatingIndex	Identifies which rating set to use for transformers.

continues on next page

Table 2 – continued from previous page

Type	Field Name	Description
Integer	FaultPlotSteps	Fault plot steps per iteration for waveform plots.
Float	FaultPlotMaxTime	Fault plot max time in seconds for waveform plots.
Boolean	FaultPlotinCycles	Fault plot time in cycles for waveform plots, returns *True* if successful.
Boolean	FaultPlotinkA	Fault plot current in kA for waveform plots, returns *True* if successful. • 1 = not selected • 2 = selected
Boolean	FaultPlotRed	Fault plot red phase for waveform plots, returns *True* if successful.
Boolean	FaultPlotYellow	Fault plot yellow phase for waveform plots, returns *True* if successful.
Boolean	FaultPlotBlue	Fault plot blue phase for waveform plots, returns *True* if successful.
Boolean	FaultPlotDC	Fault plot DC component for waveform plots, returns *True* if successful.
Boolean	FaultPlotRMS	Fault plot RMS component for waveform plots, returns *True* if successful.
Boolean	FaultPlot2Harm	Fault plot 2nd harmonic component for waveform plots, returns *True* if successful.
Boolean	FaultPlotMaxAsymmRed	Fault plot maximum asymmetry in red phase for waveform plots, returns *True* if successful.
Integer	MotorToStart	The motor calculation is started for the motor UID.

IscAnalysisFL Class

class *ipsa.***IscAnalysisFL**

Analysis class for the fault level analysis. Motor start analysis options are provided under the fault level analysis class.

GetIValue(*nFieldIndex: int*) → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (*int*) – The given enumerated field.

Returns

The integer value for the field.

Return type

int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a float value for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The float value for the field.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The string value for the field.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The boolean value for the field.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the integer value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **nValue** (***int***) – The integer value that will be set.

Returns
True if successful.

Return type
bool

***SetDValue**(nFieldIndex: **int**, dValue: **float**)* → **bool**

Sets the float value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **dValue** (***float***) – The float value that will be set.

Returns
True if successful.

Return type
bool

***SetSValue**(nFieldIndex: **int**, strValue: **str**)* → **bool**

Sets the string value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **strValue** (***str***) – The string value that will be set.

Returns
True if successful.

Return type
bool

***SetBValue**(nFieldIndex: **int**, bValue: **bool**)* → **bool**

Sets the integer value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **bValue** (***bool***) – The boolean value that will be set.

Returns
True if successful.

Return type
bool

***GetFieldType**(nFieldIndex: **int**)* → **str**

Returns the field type as a string for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The field type.

Return type
str

GetFieldName(*nFieldIndex: **int***) → **str**

Returns the field name as a string for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The field name.

Return type
str

SetBusesToFault(*nUIDs: **List**[**int**]*) → **None**

Specifies which busbars will be faulted as defined by the list of busbar UIDs. Only applicable when the FaultStudyType is set to FaultSelectedBusbars.

Parameters
nUIDs (***list(int)***) – The list of busbar UIDs which will be faulted.

GetBusesToFault() → **List**[**int**]

Returns a list of busbar UIDs representing the busbars that have been selected to be faulted.

Returns
The list of faulted busbars.

Return type
list(**int**)

IscAnalysisHM

Field Values

Table 3: **IscAnalysisHM Field Values**

Type	Field Name	Description
Integer	HarmonicStudy-Type	Type of harmonic study. Should be one of: • 0 = Harmonic voltage penetration • 1 = Harmonic voltage waveform • 2 = Harmonic impedance scan
Integer	FundamentalTHD	Use fundamental voltage for THD calculation.

continues on next page

Table 3 – continued from previous page

Type	Field Name	Description
Integer	MinimumHarmoni-cOrder	Minimum harmonic order.
Integer	MaximumHarmoni-cOrder	Maximum harmonic order.
Integer	HarmonicWave-formBusbar HarmonicWave-formBusbar2 HarmonicWave-formBusbar3 HarmonicWave-formBusbar4 HarmonicWave-formBusbar5 HarmonicWave-formBusbar6	Busbar to produce waveform for. Up to six busbars can be specified.
Integer	HarmonicSe-quence	Sequence network to use for harmonics. • 0 = Zero sequence impedance used for triplen orders, positive sequence impedances used for all others • 1 = Only the positive sequence network impedances are used • 2 = Only the zero sequence network impedances are used
Integer	HarmonicUseLong-Lines	Global override using long lines.
Integer	HarmonicGlob-alLineModel	Global override line model. One of the following: • 0 = Polynomial resistance model • 1 = Resistance square root model • 2 = Constant X/R model
Integer	HarmonicGlobal-TransformerModel	Global override transformer model. One of the following: • 0 = Polynomial resistance mode • 1 = Resistance square root model • 2 = Constant X/R model

continues on next page

Table 3 – continued from previous page

Type	Field Name	Description
Integer	HarmonicGlobalShuntModel	Global override shunt model. One of the following: • 0 = Use default resistance to give X/R = 2000.0 if no resistance passed • 1 = Ideal shunt with no resistance
Integer	HarmonicGlobalLoadModel	Global override load model. One of the following: • 0 = Series RX model • 1 = Parallel RX 1 model • 2 = Parallel RX 2 model • 3 = X plus parallel RX model
Float	HarmonicOffNominalFrequencyHz	Off-nominal frequency (Hz).
Integer	HarmonicOnlyScanResonant	Only scan harmonic resonant zones in detail.
Float	HarmonicScanStepSizePU	Step size for harmonic sensitivity scans (pu).
Integer	HarmonicPlotVoltageType	Plot the harmonic voltage as it varies with harmonic order. Should be one of: • 0 = Plot voltage waveform • 1 = Plot harmonic voltages as a bar chart
Integer	HarmonicPlotImpedanceType	Plot the harmonic impedance as it varies with harmonic order. Should be one of: • 0 = Z - Plot the total impedance. • 1 = R - Plot the resistance. • 2 = X - Plot the reactance.
Boolean	HarmonicPlotSeparateFundamental	If this option is selected then the fundamental waveform will be plotted separately from the harmonics waveform. If this option is not selected then the waveforms will be superimposed.
Boolean	HarmonicPlotZ	If this option is *True* then the harmonic impedance Z will be plotted.
Boolean	HarmonicPlotR	If this option is *True* then the harmonic resistance waveform will be plotted.
Boolean	HarmonicPlotX	If this option is *True* then the harmonic reactance waveform will be plotted.

continues on next page

Table 3 – continued from previous page

Type	Field Name	Description
Boolean	HarmonicPlotUseLogarithmic	If this option is *True* then plot axes will be logarithmic.
Boolean	HarmonicPlotUseFrequency	If this option is *True* then the harmonics impedance will be plotted against frequency in Hertz, else it will be plotted against the harmonic order.
Boolean	HarmonicPlotUseOhms	If this option is *True* then the impedance plot will be in per unit ohms on the system base, else it will be in actual Ohms.

IscAnalysisHM Class

*class ipsa.**IscAnalysisHM***

Analysis class for the harmonic analysis.

***GetIValue**(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (*int*) – The given enumerated field.

Returns

The integer value for the field.

Return type

int

***GetDValue**(nFieldIndex: **int**)* → **float**

Returns a float value for the enumerated field.

Parameters

nFieldIndex (*int*) – The given enumerated field.

Returns

The float value for the field.

Return type

float

***GetSValue**(nFieldIndex: **int**)* → **str**

Returns a string value for the enumerated field.

Parameters

nFieldIndex (*int*) – The given enumerated field.

Returns

The string value for the field.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The boolean value for the field.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the integer value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **nValue** (***int***) – The integer value that will be set.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the float value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **dValue** (***float***) – The float value that will be set.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: **int**, strValue: **str***) → **bool**

Sets the string value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **strValue** (***str***) – The string value that will be set.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: **int***, *bValue: **bool***) → **bool**

Sets the integer value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **bValue** (***bool***) – The boolean value that will be set.

Returns
True if successful.

Return type
bool

GetFieldType(*nFieldIndex: **int***) → **str**

Returns the field type as a string for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The field type.

Return type
str

GetFieldName(*nFieldIndex: **int***) → **str**

Returns the field name as a string for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The field name.

Return type
str

SetBusesToAnalyse(*nUIDs: **List[int]***) → **None**

Specifies which busbars will be analysed as defined by the list of busbar UIDs.

Parameters
nUIDs (***list(int)***) – The list of busbar UIDs which will be analysed.

GetBusesToAnalyse() → **List**[**int**]

Returns a list of busbar UIDs representing the busbars that have been selected to be analysed.

Returns
The list of analysed busbars.

Return type
list(**int**)

IscAnalysisDCLF

Field Values

Table 4: **IscAnalysisDCLF Field Values**

Type	Field Name	Description
Boolean	CalculateNodalTLF	If this option is selected then the nodal transmission loss factors will be calculated for the network.
Boolean	CalculateLODF	If this option is selected then the line outage distribution factors will be calculated for the network.
Integer	BranchLossEstimationMethod	Type of method used to estimate the losses in the branches in the network. Should be one of: • 0 = None (i.e. no branch losses) • 1 = PI-model. This estimates the branch losses by placing a real power load at either end of every branch for which a resistance value has been provided.
Float	InductionMachineEfficiencyPC	The efficiency of all induction machines in percent. This will be used to estimate the electrical power to the machines from the machines mechanical output power.
Boolean	OnlyLargestIsland	If this option is selected then only the largest island in the network will be included in the DC load flow.
Boolean	NoPhaseShift	Do not apply phase shifts to DC load flow. **Note this flag is for future use!** False = Use phase shifting in DCload flow True = No phase shifting
Boolean	UseLoadScaling	Enable scaling of loads in the DC load flow calculation.

continues on next page

Table 4 – continued from previous page

Type	Field Name	Description
Float	RealLoadScale	Per unit factor used to scale all real loads (default = 1.0).
Integer	WhichImpedance	The default setting is 0 which will use the normal resistance value for branches when performing calculations. Set this value to 1 to use the minimum resistance value for branches when performing calculations.
Boolean	AutoSelectSlacks	Set *True* to automatically select slack busbars in islands where the user has not manually specified a slack busbar.

IscAnalysisDCLF Class

*class ipsa.***IscAnalysisDCLF**

Analysis class for the DC load flow analysis.

GetIValue(*nFieldIndex: int*) → int

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (*int*) – The given enumerated field.

Returns
The integer value for the field.

Return type
int

GetDValue(*nFieldIndex: int*) → float

Returns a float value for the enumerated field.

Parameters
nFieldIndex (*int*) – The given enumerated field.

Returns
The float value for the field.

Return type
float

GetSValue(*nFieldIndex: int*) → str

Returns a string value for the enumerated field.

Parameters
nFieldIndex (*int*) – The given enumerated field.

Returns
The string value for the field.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The boolean value for the field.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the integer value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **nValue** (***int***) – The integer value that will be set.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the float value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **dValue** (***float***) – The float value that will be set.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: **int**, strValue: **str***) → **bool**

Sets the string value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.

- **strValue** (*str*) – The string value that will be set.

Returns
True if successful.

Return type
bool

SetBValue(nFieldIndex: int, bValue: bool) → bool

Sets the integer value for the enumerated field.

Parameters

- **nFieldIndex** (*int*) – The given enumerated field.
- **bValue** (*bool*) – The boolean value that will be set.

Returns
True if successful.

Return type
bool

GetFieldType(nFieldIndex: int) → str

Returns the field type as a string for the enumerated field.

Parameters
nFieldIndex (*int*) – The given enumerated field.

Returns
The field type.

Return type
str

GetFieldName(nFieldIndex: int) → str

Returns the field name as a string for the enumerated field.

Parameters
nFieldIndex (*int*) – The given enumerated field.

Returns
The field name.

Return type
str

IscAnalysisAF

Field Values

Table 5: **IscAnalysisAF Field Values**

Type	Field Name	Description
Integer	IEEEStandard	Standard according to IEEE-1584 for the arc flash calculation used: • 0 = 2002 standard • 1 = 2018 standard
Float	BoundaryEnergyJcm2	Boundary energy defined at the standard level for a 2nd degree burn (defaults to 5 J/cm2).
Float	ReducedFaultCurrentPC	Reduction of fault current for more conservative arc flash calculation (default to 15%).

IscAnalysisAF Class

class *ipsa.****IscAnalysisAF***

Analysis class for the ArcFlash analysis.

GetIValue*(nFieldIndex:* ***int****)* → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (***int***) – The given enumerated field.

Returns

The integer value for the field.

Return type

int

GetDValue*(nFieldIndex:* ***int****)* → **float**

Returns a float value for the enumerated field.

Parameters

nFieldIndex (***int***) – The given enumerated field.

Returns

The float value for the field.

Return type

float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The string value for the field.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The boolean value for the field.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the integer value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **nValue** (***int***) – The integer value that will be set.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the float value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **dValue** (***float***) – The float value that will be set.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: **int***, *strValue: **str***) → **bool**

Sets the string value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **strValue** (***str***) – The string value that will be set.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: **int***, *bValue: **bool***) → **bool**

Sets the integer value for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The given enumerated field.
- **bValue** (***bool***) – The boolean value that will be set.

Returns
True if successful.

Return type
bool

GetFieldType(*nFieldIndex: **int***) → **str**

Returns the field type as a string for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The field type.

Return type
str

GetFieldName(*nFieldIndex: **int***) → **str**

Returns the field name as a string for the enumerated field.

Parameters
nFieldIndex (***int***) – The given enumerated field.

Returns
The field name.

Return type
str

1.8 IscNetComponent

The *IscNetComponent* class is the base class for all IPSA components. All functions that are exposed (described below) are accessible via the derived component classes. The functions in this section should therefore be used in conjunction with one of the IPSA component classes, e.g. for accessing busbar data the following code would be used:

```
busbar = ipsa_network.GetBusbar("Busbar1")
nBusbarUID = busbar.GetUID()
```

1.8.1 Extension Data

It is possible to add extension data to an object of any type. The definitions of the data extension fields are held as static data associated with the component, i.e. all components of the same type have the same extension data fields. The actual field values on each component are stored with the component.

All extension data is handled transparently by the IPSA filing modules and is not currently used for analysis by IPSA. All extension data fields are persistent when filed.

The field names for extended data fields **should not** contain spaces. Only alphanumeric characters and underscores are permitted.

1.8.2 Field Values

Below is a list of the field values for IscNetComponent which map each derived component object to a field value, sometimes used within the code.

Table 6: **IscNetComponent Field Values - Types**

Field Name	PyIPSA class
Unknown	An unknown IscNetComponent object
Busbar	IscBusbar
Load	IscLoad
Generator	IscSynMachine
IndMachine	IscIndMachine
Harmonic	IscHarmonic
HarmonicFilter	IscFilter
MechSwCapacitor	IscMechSwCapacitor
StaticVArC	IscStaticVC
Battery	IscBattery

continues on next page

Table 6 – continued from previous page

Field Name	PyIPSA class
DCMachine	IscDCmachine
UniMachine	IscUMachine
GridInfeed	IscGridInfeed
Line	IscBranch
Transformer	IscTransformer
ThreeWTransformer	Isc3WTransformer
ACDCConverter	IscConverter
DCDCConverter	IscChopper
MGset	IscMGset
AVR	(Not mapped to PyIPSA)
Governor	(Not mapped to PyIPSA)
DCConverterCtl	(Not mapped to PyIPSA)
ACConverterCtl	(Not mapped to PyIPSA)
DCMachineCtl	(Not mapped to PyIPSA)
PluginModel	IscPlugin
CircuitBreaker	IscCircuitBreaker
SeriesRegulator	IscVoltageRegulator
ProtectionContainer	(Not mapped to PyIPSA)
Annotation	IscAnnotation
AnalysisLF	IscAnalysisLF
AnalysisFL	IscAnalysisFL
AnalysisMS	(Not mapped to PyIPSA)
AnalysisBD	(Not mapped to PyIPSA)
AnalysisTS	(Not mapped to PyIPSA)
AnalysisHM	IscAnalysisHM
AnalysisProt	(Not mapped to PyIPSA)
Automation	(Not mapped to PyIPSA)
Contingency	(Not mapped to PyIPSA)
Study	(Not mapped to PyIPSA)
Network	IscNetwork
ResultsDisplayStyle	(Not mapped to PyIPSA)
ResultsDisplayLF	(Not mapped to PyIPSA)
SQL	(Not mapped to PyIPSA)

1.8.3 IscNetComponent Class

class** ipsa.**IscNetComponent

The base class for all IPSA components.

GetUID() → **int**

Returns the unique ID of the component.

Returns

The unique ID of the component.

Return type

int

GetName() → **str**

Gets the name as a string - this is the name Python knows the object by (only identical to the IPSA name for busbars).

Returns

The name of the component.

Return type

str

***SetName**(strName: **str**)* → **None**

Sets the name to the component to the specified name.

Parameters

strName (***str***) – The component name.

***GetRealName**(strName: **str**)* → **str**

Gets the user defined component name as a string for the specified component name.

Parameters

strName (***str***) – The component Python name.

Returns

Returns the IPSA component name.

Return type

str

***SetRealName**(strName: **str**)* → **None**

Sets the user defined IPSA component name.

Parameters

strName (***str***) – The IPSA component name.

GetFieldType(*nFieldIndex: **int***) → **str**

Returns the field type as a string for the given enumerated field.

Parameters

nFieldIndex (***int***) – The given enumerated field.

Returns

Returns 'String', 'Integer', 'Float' or 'Boolean'.

Return type

str

GetFieldName(*nFieldIndex: **int***) → **str**

Returns the field name as a string for the given enumerated field.

Parameters

nFieldIndex (***int***) – The given enumerated field.

Returns

The field name.

Return type

str

GetFromBusbarUID(*nBranchUID: **int***) → **int**

Returns the FROM busbar UID of the given branch.

Parameters

nBranchUID (***int***) – The branch UID.

Returns

The FROM busbar UID.

Return type

int

GetToBusbarUID(*nBranchUID: **int***) → **int**

Returns the TO busbar UID of the given branch.

Parameters

nBranchUID (***int***) – The branch UID.

Returns

The TO busbar UID.

Return type

int

GetType() → **int**

Returns an integer that matches one of the class field indices (e.g., IscNetComponent.Busbar).

Returns

The integer that matches one of the class' field indices.

Return type

int

AddDataExtension(*strName: **str**, default: **int** | **float** | **str***) → **int**

Adds an integer data field and returns the new field index. Sets the default value.

Note: The variable of the function is not called default.

You can use either nDefault, dDefault, or strDefault specifying the default value.

Parameters

- **strName** (*str*) – The name of the field.
- **nDefault** (*int*) – The integer default value.
- **dDefault** (*float*) – The float default value.
- **strDefault** (*str*) – The string default value.

Returns

The new field index.

Return type

int

AddListIntDataExtension(*strName: **str***) → **int**

Adds a list of integers data field and returns the new field index. Sets the default value to an empty list.

Parameters

strName (*str*) – The name of the field.

Returns

The new field index.

Return type

int

AddListDblDataExtension(*strName: **str***) → **int**

Adds a list of doubles data field and returns the new field index. Sets the default value to an empty list.

Parameters

strName (*str*) – The name of the field.

Returns

The new field index.

Return type

int

AddListStrDataExtension(*strName: **str***) → **int**

Adds a list of strings data field and returns the new field index. Sets the default value to an empty list.

Parameters
strName (***str***) – The name of the field.

Returns
The new field index.

Return type
int

GetListIntExtensionValue(*nFieldIndex: **int**, nIndex: **int***) → **int**

Get a single integer value from the list for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nIndex** (***int***) – The index of the selected element.

Returns
The element value.

Return type
int

GetListDblExtensionValue(*nFieldIndex: **int**, nIndex: **int***) → **float**

Get a single float value from the list for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nIndex** (***int***) – The index of the selected element.

Returns
The element value.

Return type
float

GetListStrExtensionValue(*nFieldIndex: **int**, nIndex: **int***) → **str**

Get a single string value from the list for the enumerated field.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nIndex** (***int***) – The index of the selected element.

Returns
The element value.

Return type
str

GetListIntSize(*nFieldIndex: int*) → int

Get size of the list of integers for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The size of the field list.

Return type
int

GetListDblSize(*nFieldIndex: int*) → int

Get size of the list of doubles for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The size of the field list.

Return type
int

GetListStrSize(*nFieldIndex: int*) → int

Get size of the list of strings for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The size of the field list.

Return type
int

SetListIntExtensionValue(*nFieldIndex: int, nIndex: int, nValue: int*) → bool

Sets the value of an element in a list of integers for the enumerated field at given position to given value.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nIndex** (*int*) – The index of the selected element.
- **nValue** (*int*) – The selected value.

Returns
True if the operation was successful.

Return type
bool

SetListDblExtensionValue(*nFieldIndex: **int**, nIndex: **int**, dValue: **float***) → **bool**

Sets the value of an element in a list of doubles for the enumerated field at given position to given value.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nIndex** (***int***) – The index of the selected element.
- **dValue** (***float***) – The selected value.

Returns
True if the operation was successful.

Return type
bool

SetListStrExtensionValue(*nFieldIndex: **int**, nIndex: **int**, strValue: **str***) → **bool**

Sets the value of an element in a list of strings for the enumerated field at given position to given value.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nIndex** (***int***) – The index of the selected element.
- **strValue** (***str***) – The selected value.

Returns
True if the operation was successful.

Return type
bool

PushBackListIntExtensionValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Adds an item to the end of a list of integers for the enumerated field with the given value.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The selected value.

Returns
True if the operation was successful.

Return type
bool

*PushBackListDblExtensionValue(nFieldIndex: **int**, dValue: **float**)* → **bool**

Adds an item to the end of a list of doubles for the enumerated field with the given value.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The selected value.

Returns

True if the operation was successful.

Return type

bool

*PushBackListStrExtensionValue(nFieldIndex: **int**, strValue: **str**)* → **bool**

Adds an item to the end of a list of strings for the enumerated field with the given value.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The selected value.

Returns

True if the operation was successful.

Return type

bool

*GetExtensionFieldIndex(strName: **str**)* → **int**

Returns the field index for the extended data field.

Parameters

strName (***str***) – The name of the extended data field.

Returns

The field index.

Return type

int

GetExtensionNames() → **Dict[int, str]**

Returns a dictionary of extension field indexes and field names. The dictionary keys are integers representing all the extended data fields. The dictionary values are the field names of the individual extended data fields. Each extended data field is therefore represented by {nIndex:strName}, where integer nIndex is the field index and string strName is the field name.

Returns

Dictionary of extension field indexes and field names.

Return type
dict(**int**, **str**)

GetNumberOfDataComponents() → **int**

Deprecated. Returns the number of data components within the IscNetComponent object. For most IscNetComponents this will return 1. To obtain the number of sections in a branch the function *IscBranch.GetSections()* should instead be used

Returns
Number of data components in the IscNetComponent object.

Return type
int

1.9 IscNetworkData

The *IscNetworkData* class provides access to the IPSA network data such as the system base MVA, to set and get data values.

1.9.1 Field Values

Table 7: **IscNetworkData Field Values**

Type	Field Name	Description
String	Title	Gets or sets the network title.
String	Author	Gets or sets the network author.
String	CreationTime	Gets the date and time when the network was first created.
String	Comment	Gets the comment entered for the network data.
Float	Base	Gets or sets the system base MVA for the network.
Boolean	BaseinKVA	*True* if the Base is in KVA otherwise the base is in MVA.
Float	Frequency	Gets or sets the system base frequency in hertz for the network.

1.9.2 IscNetworkData Class

class *ipsa.****IscNetworkData***

Provides access to the IPSA network data.

GetIValue(*nFieldIndex:* ***int***) → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The integer value.

Return type

int

GetDValue(*nFieldIndex:* ***int***) → **float**

Returns a double value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The double value.

Return type

float

GetSValue(*nFieldIndex:* ***int***) → **str**

Returns a string value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The string value.

Return type

str

GetBValue(*nFieldIndex:* ***int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The boolean value.

Return type

bool

SetIValue(*nFieldIndex: **int***, *nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns

True if successful.

Return type

bool

SetDValue(*nFieldIndex: **int***, *dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns

True if successful.

Return type

bool

SetSValue(*nFieldIndex: **int***, *strValue: **int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns

True if successful.

Return type

bool

SetBValue(*nFieldIndex: **int***, *bValue: **bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns

True if successful.

Return type
bool

GetBranchRatingName(nIndex: int) → **str**

Returns the name representing the rating set identified by an index.

Parameters
nIndex (*int*) – The specified index.

Returns
The rating set name, or empty set if no rating set with that index exists in the network.

Return type
str

1.10 IscBusbar

The *IscBusbar* class provides access to an IPSA busbar, to set and get data values and to retrieve load flow and fault level results.

1.10.1 Field Values

Table 8: **IscBusbar Field Values**

Type	Field Name	Description
String	Name	Gets the busbar name.
Float	NomVoltkV	Nominal bus voltage in kV

continues on next page

Table 8 – continued from previous page

Type	Field Name	Description
Integer	ControlType	Gets the type of busbar, e.g. slack, PV, PQ, etc. • 0 = No voltage control at bus • 1 = Slack busbar • 2 = Real power and voltage control by generator • 3 = No longer used • 4 = No longer used • 5 = Voltage controlled by transformer • 6 = No longer used • 7 = Multiple types of voltage control, i.e. generator and transformer • 8 = Voltage controlled by remote PV generator • 9 = Voltage controlled by local switched capacitor • 10 = Voltage controlled by remote switched capacitor
Integer	Type	Gets the physical type of busbar e.g. straight joint, mains joint etc. • 0 = Unset • 1 = Straight joint • 2 = Mains joint • 3 = Service cable joint • 4 = Service termination joint • 5 = Overhead termination joint • 6 = Ground mounted substation node
Float	VoltPU	Gets the voltage magnitude in per unit.
Float	VoltAngleRad	Gets the voltage angle in radians.
String	Comment	Gets the comments.

continues on next page

Table 8 – continued from previous page

Type	Field Name	Description
Integer	ArcBusbarConfiguration	Specific busbar configuration for this bus according to the definitions penned out by IEEE-1584 standard: • 0 = Unknown • 1 = VCB (vertical closed box) • 2 = VCBB (vertical closed bolted box) • 3 = HCB (horizontal closed box) • 4 = VOA (vertical open air box) • 5 = HOA (horizontal open air box)
Float	ArcEnclosureWidthMM	Width of the busbar enclosure for the arcflash in mm.
Float	ArcEnclosureHeightMM	Height of the busbar enclosure for the arcflash in mm.
Float	ArcEnclosureDepthMM	Depth of the busbar enclosure for the arcflash in mm.
Float	ArcConductorGapMM	Air gap between the conductors that the arc flash jumps across in mm.
Float	ArcWorkingDistanceMM	Working distance for the bus container in mm.

1.10.2 IscBusbar Class

class** ipsa.**IscBusbar

Provides access to an IPSA busbar.

***SetName**(strName: **str**)* → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

***GetIValue**(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters
: **nFieldIndex** (*int*) – The field index.

Returns
: The integer value.

Return type
: int

GetDValue(nFieldIndex: int) → **float**

Returns a double value for the enumerated field.

Parameters
: **nFieldIndex** (*int*) – The field index.

Returns
: The double value.

Return type
: float

GetSValue(nFieldIndex: int) → **str**

Returns a string value for the enumerated field.

Parameters
: **nFieldIndex** (*int*) – The field index.

Returns
: The string value.

Return type
: str

GetBValue(nFieldIndex: int) → **bool**

Returns a boolean value for the enumerated field.

Parameters
: **nFieldIndex** (*int*) – The field index.

Returns
: The boolean value.

Return type
: bool

SetIValue(nFieldIndex: int, nValue: int) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: **int**, strValue: **int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: **int**, bValue: **bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns
True if successful.

Return type
bool

GetFaultDValue(*nCircuitBreakerFieldIndex: **int***) → **float**

Returns a float value for the circuit breaker field. **Note that nCircuitBreaker-FieldIndex should be one of MakePeakkA, BreakRMSkA, BreakDCPC, Break-**

TimemS or NomCurrentkA – it is an IscCircuitBreaker field index. This function is used to get fault (breaker) ratings for a busbar.

Parameters

nCircuitBreakerFieldIndex (*int*) – MakePeakkA, BreakRMSkA, BreakDCPC, BreakTimemS or NomCurrentkA – it is an IscCircuitBreaker field index.

Returns

The float value for the selected field.

Return type

float

SetFaultDValue*(nCircuitBreakerFieldIndex: **int**)* → **bool**

Sets the value for the circuit breaker field. **Note that nCircuitBreakerFieldIndex should be one of MakePeakkA, BreakRMSkA, BreakDCPC, BreakTimemS or NomCurrentkA – it is an IscCircuitBreaker field index.** This function is used to set fault (breaker) ratings for a busbar.

Parameters

nCircuitBreakerFieldIndex (*int*) – MakePeakkA, BreakRMSkA, BreakDCPC, BreakTimemS or NomCurrentkA – it is an IscCircuitBreaker field index.

Returns

True if successful.

Return type

bool

GetVoltageMagnitudePU*(nStudyUid: **int**)* → **float**

GetVoltageMagnitudePU*()* → **float**

GetVoltageMagnitudePU*(dOrder: **float**)* → **float**

Returns the voltage magnitude in per unit.

If a UID is provided this is for the associated automation or contingency study.

If a float dOrder is provided, this is the harmonic voltage magnitude for the given harmonic order.

Parameters

- **nStudyUid** (*int*) – The UID of the study.
- **dOrder** (*float*) – The harmonic order.

Returns

The voltage magnitude, if a UID is provided this returns the voltage magnitude for the associated study. If a dOrder is provided this returns the voltage magnitude for the harmonic order.

Return type
float

*GetVoltageMagnitudekV(nStudyUid: **int**)* → **float**

GetVoltageMagnitudekV() → **float**

Returns the voltage magnitude in kV. If a UID is provided this is for the associated automation or contingency study.

Parameters
nStudyUid (*int*) – The UID of the study.

Returns
The voltage magnitude, if a UID is provided this returns the voltage magnitude for the associated study.

Return type
float

GetVoltageAngleRad() → **float**

*GetVoltageAngleRad(nStudyUid: **int**)* → **float**

Returns the voltage angle in radians. If a UID is provided this is for the associated automation or contingency study.

Parameters
nStudyUid (*int*) – The UID of the study.

Returns
The voltage angle, if a UID is provided this returns the voltage angle for the associated study.

Return type
float

GetVoltageAngleDeg() → **float**

*GetVoltageAngleDeg(nStudyUid: **int**)* → **float**

Returns the voltage angle in degrees. If a UID is provided this is for the associated automation or contingency study.

Parameters
nStudyUid (*int*) – The UID of the study.

Returns
The voltage angle, if a UID is provided this returns the voltage angle for the associated study.

Return type
float

GetRealMismatchMW() → **float**

Returns the load flow MW mismatch.

Returns
The load flow MW mismatch.

Return type
float

GetReactiveMismatchMVAr() → **float**

Returns the load flow MVAr mismatch.

Returns
The load flow MVAr mismatch.

Return type
float

GetRealGenerationMW() → **float**

Returns the total MW of generation at a busbar.

Returns
The total MW of generation at a busbar.

Return type
float

GetReactiveGenerationMVAr() → **float**

Returns the total MVAr of generation at a busbar.

Returns
The total MVAr of generation at a busbar.

Return type
float

GetRealInductionMW() → **float**

Returns the total MW of induction machines at a busbar.

Returns
The total MW of induction machines at a busbar.

Return type
float

GetReactiveInductionMVAr() → **float**

Returns the total MVAr of induction machines at a busbar.

Returns
The total MVAr of induction machines at a busbar.

Return type
float

GetRealLoadMW() → **float**

Returns the total MW of static load at a busbar.

Returns
The total MW of static load at a busbar.

Return type
float

GetReactiveLoadMVAr() → **float**

Returns the total MVAr of static load at a busbar.

Returns
The total MVAr of static load at a busbar.

Return type
float

GetRedVoltageMagnitudePU() → **float**

Returns the red phase voltage magnitude in per-unit.

Returns
The red phase voltage magnitude in per-unit.

Return type
float

GetRedVoltageMagnitudekV() → **float**

Returns the red phase voltage magnitude in kV.

Returns
The red phase voltage magnitude in kV.

Return type
float

GetRedVoltageAngleDeg() → **float**

Returns the red phase voltage angle in degrees.

Returns
The red phase voltage angle in degrees.

Return type
float

GetYellowVoltageMagnitudePU() → **float**

Returns the yellow phase voltage magnitude in per-unit.

Returns
The yellow phase voltage magnitude in per-unit.

Return type
float

GetYellowVoltageMagnitudekV() → **float**

Returns the yellow phase voltage magnitude in kV.

Returns
The yellow phase voltage magnitude in kV.

Return type
float

GetYellowVoltageAngleDeg() → **float**

Returns the yellow phase voltage angle in degrees.

Returns
The yellow phase voltage angle in degrees.

Return type
float

GetBlueVoltageMagnitudePU() → **float**

Returns the blue phase voltage magnitude in per-unit.

Returns
The blue phase voltage magnitude in per-unit.

Return type
float

GetBlueVoltageMagnitudekV() → **float**

Returns the blue phase voltage magnitude in kV.

Returns
The blue phase voltage magnitude in kV.

Return type
float

GetBlueVoltageAngleDeg() → **float**

Returns the blue phase voltage angle in degrees.

Returns
The blue phase voltage angle in degrees.

Return type
float

GetFaultACComponentMVA() → **float**

Returns the AC component of fault level in MVA.

Returns
The AC component of fault level in MVA.

Return type
float

GetFaultDCComponentMVA() → **float**

Returns the DC component of fault level in MVA.

Returns
The DC component of fault level in MVA.

Return type
float

GetFault2HComponentMVA() → **float**

Returns the second harmonic component of fault level in MVA.

Returns
The second harmonic component of fault level in MVA.

Return type
float

GetFaultDCTheveninX() → **float**

Returns the inductive/capacitive component of the DC X/R ratio.

Returns
The inductive/capacitive component of the DC X/R ratio.

Return type
float

GetFaultDCTheveninR() → **float**

Returns the resistive component of the DC X/R ratio.

Returns
The resistive component of the DC X/R ratio.

Return type
float

GetFaultRedComponentMVA() → **float**

Returns the red phase component of fault level in MVA.

Returns
The red phase component of fault level in MVA.

Return type
float

GetFaultRedComponentAngleDeg() → **float**

Returns the red phase component of fault angle in degrees.

Returns
The red phase component of fault angle in degrees.

Return type
float

GetFaultYellowComponentMVA() → **float**

Returns the yellow phase component of fault level in MVA.

Returns
The yellow phase component of fault level in MVA.

Return type
float

GetFaultYellowComponentAngleDeg() → **float**

Returns the yellow phase component of fault angle in degrees.

Returns
The yellow phase component of fault angle in degrees.

Return type
float

GetFaultBlueComponentMVA() → **float**

Returns the blue phase component of fault level in MVA.

Returns
The blue phase component of fault level in MVA.

Return type
float

GetFaultBlueComponentAngleDeg() → **float**

Returns the blue phase component of fault angle in degrees.

Returns
The blue phase component of fault angle in degrees.

Return type
float

GetFaultPositiveComponentMVA() → **float**

Returns the positive sequence component of fault level in MVA.

Returns
The positive sequence component of fault level in MVA.

Return type
float

GetFaultPositiveComponentAngleDeg() → **float**

Returns the positive sequence component of fault angle in degrees.

Returns
The positive sequence component of fault angle in degrees.

Return type
float

GetFaultNegativeComponentMVA() → **float**

Returns the negative sequence component of fault level in MVA.

Returns
The negative sequence component of fault level in MVA.

Return type
float

GetFaultNegativeComponentAngleDeg() → **float**

Returns the negative sequence component of fault angle in degrees.

Returns
The negative sequence component of fault angle in degrees.

Return type
float

GetFaultZeroComponentMVA() → **float**

Returns the zero sequence component of fault level in MVA.

Returns
The zero sequence component of fault level in MVA.

Return type
float

GetFaultZeroComponentAngleDeg() → **float**

Returns the zero sequence component of fault angle in degrees.

Returns
The zero sequence component of fault angle in degrees.

Return type
float

GetFaultACComponentkA() → **float**

Returns the AC component of fault level in kA.

Returns
The AC component of fault level in kA.

Return type
float

GetFaultDCComponentkA() → **float**

Returns the DC component of fault level in kA.

Returns
The DC component of fault level in kA.

Return type
float

GetFault2HComponentkA() → **float**

Returns the second harmonic component of fault level in kA.

Returns
The second harmonic component of fault level in kA.

Return type
float

GetFaultRedComponentkA() → **float**

Returns the red phase component of fault level in kA.

Returns
The red phase component of fault level in kA.

Return type
float

GetFaultYellowComponentkA() → **float**

Returns the yellow phase component of fault level in kA.

Returns
The yellow phase component of fault level in kA.

Return type
float

GetFaultBlueComponentkA() → **float**

Returns the blue phase component of fault level in kA.

Returns
The blue phase component of fault level in kA.

Return type
float

GetFaultPositiveComponentkA() → **float**

Returns the positive sequence component of fault level in kA.

Returns
The positive sequence component of fault level in kA.

Return type
float

GetFaultNegativeComponentkA() → **float**

Returns the negative sequence component of fault level in kA.

Returns
The negative sequence component of fault level in kA.

Return type
float

GetFaultZeroComponentkA() → **float**

Returns the zero sequence component of fault level in kA.

Returns
The zero sequence component of fault level in kA.

Return type
float

GetFaultRedVoltagePU() → **float**

Returns the red phase fault voltage in per unit.

Returns
The red phase fault voltage in per unit.

Return type
float

GetFaultRedVoltageAngleDeg() → **float**

Returns the red phase fault voltage angle in degrees.

Returns
The red phase fault voltage angle in degrees.

Return type
float

GetFaultYellowVoltagePU() → **float**

Returns the yellow phase fault voltage in per unit.

Returns
The yellow phase fault voltage in per unit.

Return type
float

GetFaultYellowVoltageAngleDeg() → **float**

Returns the yellow phase fault voltage angle in degrees.

Returns
The yellow phase fault voltage angle in degrees.

Return type
float

GetFaultBlueVoltagePU() → **float**

Returns the blue phase fault voltage in per unit.

Returns
The blue phase fault voltage in per unit.

Return type
float

GetFaultBlueVoltageAngleDeg() → **float**

Returns the blue phase fault voltage angle in degrees.

Returns
The blue phase fault voltage angle in degrees.

Return type
float

GetFaultPositiveVoltagePU() → **float**

Returns the positive sequence component of fault voltage in per unit.

Returns
The positive sequence component of fault voltage in per unit.

Return type
float

GetFaultPositiveVoltageAngleDeg() → **float**

Returns the positive sequence component of fault voltage angle in degrees.

Returns
The positive sequence component of fault voltage angle in degrees.

Return type
float

GetFaultNegativeVoltagePU() → **float**

Returns the negative sequence component of fault voltage in per unit.

Returns
The negative sequence component of fault voltage in per unit.

Return type
float

GetFaultNegativeVoltageAngleDeg() → **float**

Returns the negative sequence component of fault voltage angle in degrees.

Returns
The negative sequence component of fault voltage angle in degrees.

Return type
float

GetFaultZeroVoltagePU() → **float**

Returns the zero sequence component of fault voltage in per unit.

Returns
The zero sequence component of fault voltage in per unit.

Return type
float

GetFaultZeroVoltageAngleDeg() → **float**

Returns the zero sequence component of fault voltage angle in degrees.

Returns
The zero sequence component of fault voltage angle in degrees.

Return type
float

GetFaultIEC909InitialSymRMSMVA() → **float**

Returns the initial symmetrical RMS fault level in MVA for IEC60909 analysis.

Returns
The initial symmetrical RMS fault level in MVA for IEC60909 analysis.

Return type
float

GetFaultIEC909PeakMVA() → **float**

Returns the peak fault level in MVA for IEC60909 analysis.

Returns
The peak fault level in MVA for IEC60909 analysis.

Return type
float

GetFaultIEC909AsymmetricBreakMVA() → **float**

Returns the asymmetric break fault level in MVA for IEC60909 analysis.

Returns
The asymmetric break fault level in MVA for IEC60909 analysis.

Return type
float

GetFaultIEC909SymmetricBreakMVA() → **float**

Returns the symmetric break fault level in MVA for IEC60909 analysis.

Returns
The symmetric break fault level in MVA for IEC60909 analysis.

Return type
float

GetFaultIEC909DCMagnitudeMVA() → **float**

Returns the DC fault level magnitude in MVA for IEC60909 analysis.

Returns
The DC fault level magnitude in MVA for IEC60909 analysis.

Return type
float

GetFaultIEC909SteadyStateMVA() → **float**

Returns the steady state fault level in MVA for IEC60909 analysis.

Returns
The steady state fault level in MVA for IEC60909 analysis.

Return type
float

GetFaultIEC909DCXoverR() → **float**

Returns the X/R ratio for IEC60909 analysis.

Returns
The X/R ratio for IEC60909 analysis.

Return type
float

GetFaultIEC909DCXoverRBreak() → **float**

Returns the X/R ratio at break time for IEC60909 analysis.

Returns
The X/R ratio at break time for IEC60909 analysis.

Return type
float

GetFaultIEC909InitialSymRMSkA() → float

Returns the initial symmetrical RMS fault level in kA for IEC60909 analysis.

Returns
The initial symmetrical RMS fault level in kA for IEC60909 analysis.

Return type
float

GetFaultIEC909PeakkA() → float

Returns the peak fault level in kA for IEC60909 analysis.

Returns
The peak fault level in kA for IEC60909 analysis.

Return type
float

GetFaultIEC909AsymmetricBreakkA() → float

Returns the asymmetric break fault level in kA for IEC60909 analysis.

Returns
The asymmetric break fault level in kA for IEC60909 analysis.

Return type
float

GetFaultIEC909SymmetricBreakkA() → float

Returns the symmetric break fault level in kA for IEC60909 analysis.

Returns
The symmetric break fault level in kA for IEC60909 analysis.

Return type
float

GetFaultIEC909DCMagnitudekA() → float

Returns the DC fault level magnitude in kA for IEC60909 analysis.

Returns
The DC fault level magnitude in kA for IEC60909 analysis.

Return type
float

GetFaultIEC909SteadyStatekA() → float

Returns the steady state fault level in kA for IEC60909 analysis.

Returns
The steady state fault level in kA for IEC60909 analysis.

Return type
float

GetVoltageOrders() → List[float]

Returns a list of all harmonic orders at a busbar. These harmonic orders can then be used to access busbar results at a specific harmonic order.

Returns
All harmonic orders at a busbar.

Return type
list(float)

GetVoltageMagnitudePC(dOrder: float) → float

Returns the harmonic voltage magnitude in percent for harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The harmonic voltage magnitude in percent.

Return type
float

GetVoltageAngle(dOrder: float) → float

Returns the harmonic voltage angle in radians for harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The harmonic voltage angle in radians.

Return type
float

GetImpedanceOrders() → List[float]

Returns a list of all harmonic impedance orders at a busbar. These harmonic orders can then be used to access busbar results at a specific harmonic order.

Returns
All harmonic impedance orders at a busbar.

Return type
list[float]

GetImpedanceMagnitude(dOrder: float) → float

Returns the harmonic impedance magnitude in per unit for harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The harmonic impedance magnitude in per unit.

Return type
float

GetImpedanceAngle(dOrder: float) → float

Returns the harmonic impedance angle in radians for harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The harmonic impedance angle in radians.

Return type
float

GetImpedanceReal(dOrder: float) → float

Returns the real part of the harmonic impedance in per unit for harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The real part of the harmonic impedance in per unit.

Return type
float

GetImpedanceImaginary(dOrder: float) → float

Returns the imaginary part of the harmonic impedance in per unit for harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The imaginary part of the harmonic impedance in per unit.

Return type
float

GetTotalHarmonicDistortion() → float

Returns the total harmonic distortion at a busbar in percent.

Returns
The total harmonic distortion at a busbar in percent.

Return type
float

GetHarmonicDistortion(dOrder: float) → float

Returns the harmonic distortion at a busbar in percent for order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The harmonic distortion at a busbar in percent.

Return type
float

GetMaximumDistortion() → List[float]

Returns a list of reals for harmonic order with the highest distortion. The distortion is in percent.

Returns
A list of reals for harmonic order with the highest distortion.

Return type
list[float]

GetResonances() → List[float]

Returns a list containing all the resonances found at a busbar. Each list gives the lower and upper resonance orders for each resonance found.

Returns
A list containing all the resonances found at a busbar.

Return type
list[float]

GetVoltageSum() → float

Returns the arithmetic sum of all harmonic voltages at a busbar in per unit.

Returns
The arithmetic sum of all harmonic voltages at a busbar in per unit.

Return type
float

GetAverageInterruptionHours() → float

Returns the average interruption time in hours from the reliability study results.

Returns
The average interruption time in hours from the reliability study results.

Return type
float

GetAnnualInterruptionHours() → **float**

Returns the total annual interruption time in hours from the reliability study results.

Returns
The total annual interruption time in hours from the reliability study results.

Return type
float

GetAnnualInterruptionFrequency() → **float**

Returns the number of interruptions per year from the reliability study results.

Returns
The number of interruptions per year from the reliability study results.

Return type
float

GetProfileMinimumVoltagePU() → **float**

Returns the minimum voltage in per unit from the profile study results.

Returns
The minimum voltage in per unit from the profile study results.

Return type
float

GetProfileMaximumVoltagePU() → **float**

Returns the maximum voltage in per unit from the profile study results.

Returns
The maximum voltage in per unit from the profile study results.

Return type
float

GetProfileMedianVoltagePU() → **float**

Returns the median of the voltage in per unit from the profile study results.

Returns
The median of the voltage in per unit from the profile study results.

Return type
float

GetMinimumProfileIndex() → **int**

Returns the category index which identifies the minimum busbar voltage result from the profile study results.

Returns
The minimum category index.

Return type
int

GetMaximumProfileIndex() → **int**

Returns the category index which identifies the maximum busbar voltage result from the profile study results.

Returns
The maximum category index.

Return type
int

GetDCLFVoltageAngleDeg() → **float**

Returns the voltage angle in degrees.

Returns
The voltage angle in degrees.

Return type
float

GetDCLFVoltageAngleRad() → **float**

Returns the voltage angle in radians.

Returns
The voltage angle in radians.

Return type
float

GetDCLFRealGenerationMW() → **float**

Returns the total MW of generation at a busbar.

Returns
The total MW of generation at a busbar.

Return type
float

GetDCLFRealGenerationkW() → **float**

Returns the total kW of generation at a busbar.

Returns
The total kW of generation at a busbar.

Return type
float

GetDCLFRealLoadMW() → **float**

Returns the total MW of static load at a busbar.

Returns
The total MW of static load at a busbar.

Return type
float

GetDCLFRealLoadkW() → **float**

Returns the total kW of static load at a busbar.

Returns
The total kW of static load at a busbar.

Return type
float

GetDCLFTransmissionLossFactor() → **float**

Returns transmission losses factor for the busbar.

Returns
Transmission losses factor for the busbar.

Return type
float

1.11 IscBranch

The *IscBranch* class provides access to an IPSA branch, to set and get data values and to retrieve analysis results.

Note that the branch rating sets are defined in the *IscNetwork* class.

1.11.1 Field Values

Table 9: **IscBranch Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique component ID for the “From” busbar.
Integer	ToUID	Gets the unique component ID for the “To” busbar.

continues on next page

Table 9 – continued from previous page

Type	Field Name	Description
String	FromBusName	Gets the sending busbar name.
String	ToBusName	Gets the receiving busbar name.
String	Name	Gets the branch name.
Boolean	HideLabel	*True* if the branch label (usually the name and any results) should be hidden on the diagram.
Integer	Type	Gets the branch/line type as defined below. • 0 = Unset • 1 = Overhead lines • 2 = Cable • 3 = Ducted • 4 = Mixed
Integer	Status	Line status as defined below: • 0 = Switched in. • 1 = Switched in, sending end will be opened in transient stability. • 2 = Switched in, receiving end will be opened in transient stability. • 3 = Switched in, both ends will be opened in transient stability. • -1 = Switched out, sending end will be closed in transient stability. • -2 = Switched out, receiving end will be closed in transient stability. • -3 = Switched out, both ends will be closed in transient stability.
Float	ResistancePU	Positive sequence resistance.
Float	MinResistancePU	Positive sequence minimum resistance.
Float	ReactancePU	Positive sequence reactance.
Float	SusceptancePU	Positive sequence susceptance.
Float	ZSResistancePU	Zero sequence resistance.
Float	ZSReactancePU	Zero sequence reactance.
Boolean	ZeroImpedance	*True* if treated as a zero impedance line.
Boolean	ZeroSequence	*True* if treated as a zero sequence only line.
Float	SwitchTime1Sec	Line switching time 1.
Float	SwitchTime2Sec	Line switching time 2.

continues on next page

Table 9 – continued from previous page

Type	Field Name	Description
Float	HarmRC0 HarmRC12 HarmRC1 HarmRC2 HarmRC3	Harmonic polynomial constants RC0, RC12, RC1, RC2 and RC3 in: $R_h = R[RC0+RC12.h^{0.5}0+RC1.h+RC2.h^2+RC3.h^3]$
Float	HarmXC0 HarmXC1 HarmXC2 HarmXC3 HarmXCEX HarmXEX	Harmonic polynomial constants XC0, XC1, XC2, XC3, XCEX and XEX in: $Xh = X[XC0 + XC1.h + XC2.h^2 + XC3.h^3] + XCEX.X.h^{XEX}$
Float	FailureRateYr	Branch failure rate per annum.
Float	RepairTimeHr	Branch repair time in hours.
String	DbType1	Branch database type. For representing the cable at the From end of the transformer.
String	DbType2	Second cable database type representing the cable at the To end of the transformer.
Float	DbLength1 *or* LengthKm	First cable database length.
Float	DbLength2	Second cable database length (for transformers only).
Integer	DbPar1	Gets the number of lines of database type 1 in parallel.
Integer	DbPar2	Gets the number of lines of database type 2 in parallel.
String	DbTranType	Gets the transformer database type (only for transformers).
Integer	DbTranPar	Gets the number of transformers in parallel (database only and only for transformers).
String	UdmID	Gets the UDM ID.
Integer	UdmDevEnd	Gets the device end.
Integer	UdmCtrlType	Gets the UDM type.
Integer	UdmCtrlUID	Gets the UDM control ID.
String	PluginID	Plugin Name, empty string means no plugin is assigned.

1.11.2 IscBranch Class

class *ipsa.***IscBranch**

Provides access to the IPSA branch.

SetName*(strName: **str**)* → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

AddSections*(nSections: **int**)* → **None**

Add sections to the branch. All branches start with one section.

Parameters

nSections (***int***) – The number of sections.

GetSections*()* → **int**

Returns the number of sections in the branch. All branches have at least one section.

Returns

The number of sections in the branch.

Return type

int

GetIValue*(nFieldIndex: **int**)* → **int**

GetIValue*(nSection: **int**, nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters

- **nSection** (***int***) – The index of the section.
- **nFieldIndex** (***int***) – The field index.

Returns

The integer value.

Return type

int

GetDValue*(nFieldIndex: **int**)* → **float**

GetDValue(*nSection: **int**, nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters

- **nSection** (***int***) – The index of the section.
- **nFieldIndex** (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

GetSValue(*nSection: **int**, nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters

- **nSection** (***int***) – The index of the section.
- **nFieldIndex** (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

GetBValue(*nSection: **int**, nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters

- **nSection** (***int***) – The index of the section.
- **nFieldIndex** (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

SetIValue(*nSection: **int**, nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nSection** (***int***) – The index of the section.

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(nFieldIndex: int, dValue: float) → bool

SetDValue(nSection: int, nFieldIndex: int, dValue: float) → bool

Sets the value for the enumerated field from a double.

Parameters

- **nSection** (*int*) – The index of the section.
- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(nFieldIndex: int, strValue: int) → bool

SetSValue(nSection: int, nFieldIndex: int, strValue: int) → bool

Sets the value for the enumerated field from a string.

Parameters

- **nSection** (*int*) – The index of the section.
- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(nFieldIndex: int, bValue: bool) → bool

SetBValue(nSection: int, nFieldIndex: int, bValue: bool) → bool

Sets the value for the enumerated field from boolean.

Parameters

- **nSection** (*int*) – The index of the section.
- **nFieldIndex** (*int*) – The field index.

- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

GetRatingMVA(nRatingIndex: int) → **float**

GetRatingMVA(nSection: int, nRatingIndex: int) → **float**

Returns the MVA rating associated with the rating set given by the rating index. Set 0 for details of branch rating indices.

Parameters

- **nSection** (*int*) – The index of the section.
- **nRatingIndex** (*int*) – The rating index.

Returns
The MVA rating associated with the rating set.

Return type
float

GetRatingSendkA(nRatingIndex: int) → **float**

GetRatingSendkA(nSection: int, nRatingIndex: int) → **float**

Returns the send end kA rating associated with the rating set given by the rating index. Set 0 for details of branch rating indices.

Parameters

- **nSection** (*int*) – The index of the section.
- **nRatingIndex** (*int*) – The rating index.

Returns
The send end kA rating associated with the rating set.

Return type
float

GetRatingReceivekA(nRatingIndex: int) → **float**

GetRatingReceivekA(nSection: int, nRatingIndex: int) → **float**

Returns the receiving end kA rating associated with the rating set given by the rating index. Set 0 for details of branch rating indices.

Parameters

- **nSection** (*int*) – The index of the section.
- **nRatingIndex** (*int*) – The rating index.

Returns

The receiving end kA rating associated with the rating set.

Return type

float

SetRatingMVA(nRatingIndex: int, dRatingMVA: float) → None

SetRatingMVA(nSection: int, nRatingIndex: int, dRatingMVA: float) → None

Sets the MVA rating to the specified rating MVA for the rating set given by the rating index.

Parameters

- **nSection** (*int*) – The index of the section.
- **nRatingIndex** (*int*) – The rating index.
- **dRatingMVA** (*float*) – The rating MVA.

SetRatingkA(nRatingIndex: int, dRatingkA: float) → None

SetRatingkA(nSection: int, nRatingIndex: int, dRatingkA: float) → None

Sets the kA rating to the specified rating kA for the rating set given by the rating index.

Parameters

- **nSection** (*int*) – The index of the section.
- **nRatingIndex** (*int*) – The rating index.
- **dRatingkA** (*float*) – The rating kA.

SetRatingSendkA(nRatingIndex: int, dRatingkA: float) → None

SetRatingSendkA(nSection: int, nRatingIndex: int, dRatingkA: float) → None

Sets the send end kA rating to the specified rating kA for the rating set given by the rating index.

Parameters

- **nSection** (*int*) – The index of the section.
- **nRatingIndex** (*int*) – The rating index.
- **dRatingkA** (*float*) – The rating kA.

SetRatingReceivekA(nRatingIndex: int, dRatingkA: float) → None

SetRatingReceivekA(nSection: int, nRatingIndex: int, dRatingkA: float) → None

Sets the receiving end kA rating to the specified rating kA for the rating set given by the rating index.

Parameters

- **nSection** (*int*) – The index of the section.

- **nRatingIndex** (*int*) – The rating index.
- **dRatingkA** (*float*) – The rating kA.

PopulateByDBEntry(strLineDataName: str, dLength: float, nParallel: int) → **bool**

Populates the object data with database information from the first database that was loaded.

Parameters

- **strLineDataName** (*str*) – The name of the branch.
- **dLength** (*float*) – The length of the branch.
- **nParallel** (*int*) – The number of parallel components.

Returns
Returns True if successful.

Return type
bool

GetSendPowerMagnitudeMVA() → **float**

Returns the branch sending end power in MVA.

Returns
The branch sending end power in MVA.

Return type
float

GetSendPowerMagnitudekVA() → **float**

Returns the branch sending end power in kVA.

Returns
The branch sending end power in kVA.

Return type
float

GetSendRealPowerMW() → **float**

Returns the branch sending end power in MW.

Returns
The branch sending end power in MW.

Return type
float

GetSendReactivePowerMVAr() → **float**

Returns the branch sending end power in MVAr.

Returns
The branch sending end power in MVAr.

Return type
float

GetSendRealPowerkW() → **float**

Returns the branch sending end power in kW.

Returns
The branch sending end power in kW.

Return type
float

GetSendReactivePowerkVAr() → **float**

Returns the branch sending end power in kVAr.

Returns
The branch sending end power in kVAr.

Return type
float

GetReceivePowerMagnitudeMVA() → **float**

Returns the branch receiving end power in MVA.

Returns
The branch receiving end power in MVA.

Return type
float

GetReceivePowerMagnitudekVA() → **float**

Returns the branch receiving end power in kVA.

Returns
The branch receiving end power in kVA.

Return type
float

GetReceiveRealPowerMW() → **float**

Returns the branch receiving end power in MW.

Returns
The branch receiving end power in MW.

Return type
float

GetReceiveReactivePowerMVAr() → **float**

Returns the branch receiving end power in MVAr.

Returns
The branch receiving end power in MVAr.

Return type
float

GetReceiveRealPowerkW() → **float**

Returns the branch receiving end power in kW.

Returns
The branch receiving end power in kW.

Return type
float

GetReceiveReactivePowerkVAr() → **float**

Returns the branch receiving end power in kVAr.

Returns
The branch receiving end power in kVAr.

Return type
float

GetLargestPowerMagnitudeMVA() → **float**

GetLargestPowerMagnitudeMVA*(nStudyUID: **int**)* → **float**

Returns the highest branch power in MVA.

Parameters
nStudyUID (***int***) – If supplied, the automation or contingency study UID which the results are for

Returns
The highest branch power in MVA.

Return type
float

GetLargestPowerMagnitudekVA() → **float**

Returns the highest branch power in kVA.

Returns
The highest branch power in kVA.

Return type
float

GetLargestRealPowerMW() → **float**

Returns the highest branch power in MW.

Returns
The highest branch power in MW.

Return type
float

GetLargestReactivePowerMVAr() → **float**

Returns the highest branch power in MVAr.

Returns
The highest branch power in MVAr.

Return type
float

GetLargestRealPowerkW() → **float**

Returns the highest branch power in kW.

Returns
The highest branch power in kW.

Return type
float

GetLargestReactivePowerkVAr() → **float**

Returns the highest branch power in kVAr.

Returns
The highest branch power in kVAr.

Return type
float

GetLossesMW() → **float**

Returns the branch losses in MW.

Returns
The branch losses in MW.

Return type
float

GetLossesMVAr() → **float**

Returns the branch losses in MVAr.

Returns
The branch losses in MVAr.

Return type
float

GetLosseskW() → **float**

Returns the branch losses in kW.

Returns
The branch losses in kW.

Return type
float

GetLosseskVAr() → float

Returns the branch losses in kVAr.

Returns
The branch losses in kVAr.

Return type
float

GetFaultRedComponentMVA() → float

Returns the red phase level component in MVA.

Returns
The red phase level component in MVA.

Return type
float

GetFaultYellowComponentMVA() → float

Returns the yellow phase fault level component in MVA.

Returns
The yellow phase fault level component in MVA.

Return type
float

GetFaultBlueComponentMVA() → float

Returns the blue phase fault level component in MVA.

Returns
The blue phase fault level component in MVA.

Return type
float

GetFaultPositiveComponentMVA() → float

Returns the positive sequence fault level component in MVA.

Returns
The positive sequence fault level component in MVA.

Return type
float

GetFaultNegativeComponentMVA() → **float**

Returns the negative sequence fault level component in MVA.

Returns
The negative sequence fault level component in MVA.

Return type
float

GetFaultZeroComponentMVA() → **float**

Returns the zero sequence fault level component in MVA.

Returns
The zero sequence fault level component in MVA.

Return type
float

GetFaultRedComponentkA() → **float**

Returns the red phase component of fault current in kA.

Returns
The red phase component of fault current in kA.

Return type
float

GetFaultYellowComponentkA() → **float**

Returns the yellow phase component of fault current in kA.

Returns
The yellow phase component of fault current in kA.

Return type
float

GetFaultBlueComponentkA() → **float**

Returns the blue phase component of fault current in kA.

Returns
The blue phase component of fault current in kA.

Return type
float

GetFaultPositiveComponentkA() → **float**

Returns the positive sequence component of fault current in kA.

Returns
The positive sequence component of fault current in kA.

Return type
float

GetFaultNegativeComponentkA() → **float**

Returns the negative sequence component of fault current in kA.

Returns
The negative sequence component of fault current in kA.

Return type
float

GetFaultZeroComponentkA() → **float**

Returns the zero sequence component of fault current in kA.

Returns
The zero sequence component of fault current in kA.

Return type
float

GetFaultRedComponentAngleDeg() → **float**

Returns the red phase component of fault angle in degrees.

Returns
The red phase component of fault angle in degrees.

Return type
float

GetFaultYellowComponentAngleDeg() → **float**

Returns the yellow phase component of fault angle in degrees.

Returns
The yellow phase component of fault angle in degrees.

Return type
float

GetFaultBlueComponentAngleDeg() → **float**

Returns the blue phase component of fault angle in degrees.

Returns
The blue phase component of fault angle in degrees.

Return type
float

GetFaultPositiveComponentAngleDeg() → **float**

Returns the positive sequence component of fault angle in degrees.

Returns
The positive sequence component of fault angle in degrees.

Return type
float

GetFaultNegativeComponentAngleDeg() → **float**

Returns the negative sequence component of fault angle in degrees.

Returns
The negative sequence component of fault angle in degrees.

Return type
float

GetFaultZeroComponentAngleDeg() → **float**

Returns the zero sequence component of fault angle in degrees.

Returns
The zero sequence component of fault angle in degrees.

Return type
float

GetCurrentMagnitude(*dOrder: **float***) → **float**

Returns the current magnitude in per unit on the network base for the harmonic order.

Parameters
dOrder (***float***) – The harmonic order.

Returns
The current magnitude in per unit.

Return type
float

GetCurrentAngle(*dOrder: **float***) → **float**

Returns the current angle in radians for the harmonic order.

Parameters
dOrder (***float***) – The harmonic order.

Returns
The current angle in radians.

Return type
float

GetResistance(*dOrder: **float***) → **float**

Returns the branch harmonic resistance in per unit on the network base for the harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The branch harmonic resistance in per unit.

Return type
float

GetReactance(dOrder: float) → float

Returns the branch harmonic reactance in per unit on the network base for the harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The branch harmonic reactance in per unit.

Return type
float

GetSusceptance(dOrder: float) → float

Returns the branch harmonic susceptance in per unit on the network base for the harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The branch harmonic susceptance in per unit.

Return type
float

GetProfileMinimumFlowMVA() → float

Returns the minimum branch flow in MVA from the profile study results.

Returns
The minimum branch flow in MVA from the profile study results.

Return type
float

GetProfileMinimumFlowkA() → float

Returns the minimum branch flow in kA from the profile study results.

Returns
The minimum branch flow in kA from the profile study results.

Return type
float

GetProfileMaximumFlowMVA() → **float**

Returns the maximum branch flow in MVA from the profile study results.

Returns
The maximum branch flow in MVA from the profile study results.

Return type
float

GetProfileMaximumFlowkA() → **float**

Returns the maximum branch flow in kA from the profile study results.

Returns
The maximum branch flow in kA from the profile study results.

Return type
float

GetProfileMedianFlowMVA() → **float**

Returns the median of the branch flow in MVA from the profile study results.

Returns
The median of the branch flow in MVA from the profile study results.

Return type
float

GetProfileMedianFlowkA() → **float**

Returns the median of the branch flow in kA from the profile study results.

Returns
The median of the branch flow in kA from the profile study results.

Return type
float

GetMinimumProfileIndex() → **int**

Returns the category index which identifies the minimum branch flow from the profile study results.

Returns
The minimum category index.

Return type
int

GetMaximumProfileIndex() → **int**

Returns the category index which identifies the maximum branch flow from the profile study results.

Returns
The maximum category index.

Return type
int

GetDCLFSendPowerMagnitudeMVA() → **float**

Returns the branch sending end power in MVA.

Returns
The branch sending end power in MVA.

Return type
float

GetDCLFSendPowerMagnitudekVA() → **float**

Returns the branch sending end power in kVA.

Returns
The branch sending end power in kVA.

Return type
float

GetDCLFSendRealPowerMW() → **float**

Returns the branch sending end power in MW.

Returns
The branch sending end power in MW.

Return type
float

GetDCLFSendRealPowerkW() → **float**

Returns the branch sending end power in kW.

Returns
The branch sending end power in kW.

Return type
float

GetDCLFReceivePowerMagnitudeMVA() → **float**

Returns the branch receiving end power in MVA.

Returns
The branch receiving end power in MVA.

Return type
float

GetDCLFReceivePowerMagnitudekVA() → **float**

Returns the branch receiving end power in kVA.

Returns
The branch receiving end power in kVA.

Return type
float

GetDCLFReceiveRealPowerMW() → **float**

Returns the branch receiving end power in MW.

Returns
The branch receiving end power in MW.

Return type
float

GetDCLFReceiveRealPowerkW() → **float**

Returns the branch receiving end power in kW.

Returns
The branch receiving end power in kW.

Return type
float

GetDCLFLargestPowerMagnitudeMVA() → **float**

Returns the highest branch power in MVA.

Returns
The highest branch power in MVA.

Return type
float

GetDCLFLargestPowerMagnitudekVA() → **float**

Returns the highest branch power in kVA.

Returns
The highest branch power in kVA.

Return type
float

GetDCLFLargestRealPowerMW() → **float**

Returns the highest branch power in MW.

Returns
The highest branch power in MW.

Return type
float

GetDCLFLargestRealPowerkW() → **float**

Returns the highest branch power in kW.

Returns
The highest branch power in kW.

Return type
float

GetDCLFLossesMW() → **float**

Returns the branch losses in MW.

Returns
The branch losses in MW.

Return type
float

GetDCLFLosseskW() → **float**

Returns the branch losses in kW.

Returns
The branch losses in kW.

Return type
float

1.12 IscTransformer

The *IscTransformer* class provides access to an IPSA transformer, to set and get data values and to retrieve load flow and fault level results. **Note that in IPSA a transformer is modelled as a combination of a branch and a tap changer. Therefore the transformer impedance data is stored in a branch instance and functions such as** *GetLineDValue()* **are used to access branch type data.**

1.12.1 Field Values

Table 10: **IscTransformer Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID of the sending busbar.
Integer	ToUID	Gets the unique ID of the receiving busbar.
String	FromBusName	Gets the sending busbar name.
String	ToBusName	Gets the receiving busbar name.

continues on next page

Table 10 – continued from previous page

Type	Field Name	Description
String	Name	Gets the transformer name.
Integer	Type	Specifies the transformer type as follows: • 0 = Unknown • 1 = Ground Mounted • 2 = Pole Mounted • 3 = Bulk Supply • 4 = Grid Supply • 5 = Super Grid • 6 = Primary Distribution • 7 = Secondary Distribution
Integer	Winding	Transformer winding type connection as follows: • 1 = none • 2 = Xx0 • 3 = Yy0 • 4 = Dd0 • 5 = Xy0 • 6 = Yx0 • 7 = Dx11 • 8 = Dy11 • 9 = Xd11 • 10 = Yd11 • 11 = Dx1 • 12 = Dy1 • 13 = Xd1 • 14 = Yd1 • 15 = Xy0, zero sequence current passing • 16 = Yx0, zero sequence current passing • 17 = Dz0 • 18 = Zd0 where: • X = Earthed star • Y = Unearthed star • D = Delta • Z = Zig-zag
Float	NEResistanceW1PU	Winding 1 neutral earth resistance in per unit.

continues on next page

Table 10 – continued from previous page

Type	Field Name	Description
Float	NEReactanceW1PU	Winding 1 neutral earth reactance in per unit.
Float	NEResistanceW2PU	Winding 2 neutral earth resistance in per unit.
Float	NEReactanceW2PU	Winding 2 neutral earth reactance in per unit.
Float	TapNominalPC	Nominal tap position, optionally used in a flat start.
Float	TapStartPC	Present tap position, used as a starting point for the next load flow.
Float	MinTapPC	Minimum tap position, normally negative or zero.
Float	TapStepPC	Tap increment. This defaults to 0.01 if left blank.
Float	MaxTapPC	Maximum tap position, normally positive or zero.
Float	DxDTap	Changes in reactance with tap change. This value is used in compounding only.
Boolean	LockTap	Sets the flag to lock the transformer tap changer. Use *True* to lock, *False* to unlock.
Float	SpecVPU	Target voltage in per unit. Positive means control 'to' busbar, negative means control 'from' busbar. Magnitudes of less than 0.5 pu mean fixed tap operation.
Float	RBWidthPC	Full bandwidth of the voltage sensing relay. This should be larger than tap step size.
Float	CompRPC	Line drop compensation resistance in percentage on the compensation rating base.
Float	CompXPC	Line drop compensation reactance in percentage on the compensation rating base.
Float	RatingMVA	Rating used for line drop compensation impedances. This can be a different value from the branch rating used for overloads.
Float	PhShiftDeg	Phase shift angle. A positive value makes the receiving end voltage lead the sending end voltage.
Float	SpecPowerMW	Quad Booster target power in MW - can be specified as zero.
Boolean	SpecPowerAtSend	Control the power at the "from" side of the transformer.
Float	MinPhShiftDeg	Min phase shift angle - both angle limits are required for Power control.
Float	MaxPhShiftDeg	Max phase shift angle - both angle limits are required for Power control.
Float	PhShiftStepDeg	Phase shift step - default value is 0.01 degrees.
String	DbType	Gets the transformer database type including both tap and impedance information.

continues on next page

Table 10 – continued from previous page

Type	Field Name	Description
Integer	DbParallel	Gets the number of transformers in parallel. This is only used for database transformers.
String	PluginID	Gets and sets the plugin name associated with this transformer.
Float	VoltFactorPt	Sets the voltage factor for use in IEC60909 fault calculations.
Integer	RemoteCtlBusbarUID	Specifies the UID of the remote busbar which is used as the basis for the transformer voltage control.

1.12.2 IscTransformer Class

class** ipsa.**IscTransformer

Provides access to an IPSA transformer.

***SetName**(strName: **str**)* → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

***GetIValue**(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The integer value.

Return type

int

***GetDValue**(nFieldIndex: **int**)* → **float**

Returns a double value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex:* ***int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex:* ***int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex:* ***int***, *nValue:* ***int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex:* ***int***, *dValue:* ***float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: int*, *strValue: int*) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: int*, *bValue: bool*) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

GetLineIValue(*nFieldIndex: int*) → **int**

Returns an integer value for the field index for the line associated with this transformer.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The integer value for the field index for the line associated with this transformer.

Return type
int

GetLineDValue(*nFieldIndex: int*) → **float**

Returns a double value for the field index for the line associated with this transformer.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The double value for the field index for the line associated with this transformer.

Return type
float

GetLineSValue(nFieldIndex: int) → **str**

Returns a string value for the field index for the line associated with this transformer.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The string value for the field index for the line associated with this transformer.

Return type
str

SetLineIValue(nFieldIndex: int, nValue: int) → **bool**

Sets the value for the field index from an integer for the line associated with this transformer.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The given integer value.

Returns
True if successful.

Return type
bool

SetLineDValue(nFieldIndex: int, dValue: float) → **bool**

Sets the value for the field index from a double for the line associated with this transformer.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetLineSValue(*nFieldIndex: **int**, strValue: **int***) → **bool**

Sets the value for the field index from a string for the line associated with this transformer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

SetRatingskA(*nRatingIndex: **int**, dSendRatingkA: **float**, dRecieveRatingkA: **float***) → **None**

Sets the sending and receiving end current ratings in kA for the transformer.

Parameters

- **nRatingIndex** (***int***) – Specifies which rating set the data is applied to.
- **dSendRatingkA** (***float***) – The sending end current rating in kA for the transformer.
- **dRecieveRatingkA** (***float***) – The receiving end current rating in kA for the transformer.

SetRatingMVA(*nRatingIndex: **int**, dRatingMVA: **float***) → **None**

Sets the MVA rating for the transformer.

Parameters

- **nRatingIndex** (***int***) – Specifies which rating set the data is applied to.
- **dRatingMVA** (***float***) – The MVA rating for the transformer.

GetRatingSendkA(*nRatingIndex: **int***) → **float**

Returns the sending end current ratings in kA for the transformer.

Parameters
nRatingIndex (***int***) – Specifies which rating set the data is applied to.

Returns
The sending end current ratings in kA for the transformer.

Return type
float

GetRatingReceivekA(*nRatingIndex: **int***) → **float**

Returns the receiving end current ratings in kA for the transformer.

Parameters
nRatingIndex (***int***) – Specifies which rating set the data is applied to.

Returns
The receiving end current ratings in kA for the transformer.

Return type
float

GetRatingMVA(*nRatingIndex: **int***) → **float**

Returns the MVA rating for the transformer.

Parameters
nRatingIndex (***int***) – Specifies which rating set the data is applied to.

Returns
The MVA rating for the transformer.

Return type
float

GetControlledBusbarName() → **str**

Returns the name of the busbar whose voltage is controlled by the transformer.

Returns
The name of the busbar whose voltage is controlled by the transformer.

Return type
str

PopulateByDBEntry(*strTransformerDataName: **str**, strLine1DataName: **str**, strLine2DataName: **str**, nParallel: **int**, nParallelFrom: **int**, nParallelTo: **int**, dlengthFrom: **float**, dLengthTo: **float***) → **bool**

Populates the object data with database information from the first database that was loaded.

Parameters

- **strTransformerDataName** (***str***) – The name of the transformer.
- **strLine1DataName** (***str***) – The name of the From branch.
- **strLine2DataName** (***str***) – The name of the To branch.
- **nParallel** (***int***) – The number of parallel components.

- **nParallelFrom** (*int*) – The number of parallel components for the From branch.
- **nParallelTo** (*int*) – The number of parallel components for the To branch.
- **dlengthFrom** (*float*) – The length of the From branch.
- **dLengthTo** (*float*) – The length of the To branch.

Returns
True if successful.

Return type
bool

GetSendPowerMagnitudeMVA() → **float**

Returns the transformer sending end power in MVA.

Returns
The transformer sending end power in MVA.

Return type
float

GetSendPowerMagnitudekVA() → **float**

Returns the transformer sending end power in kVA.

Returns
The transformer sending end power in kVA.

Return type
float

GetSendRealPowerMW() → **float**

Returns the transformer sending end power in MW.

Returns
The transformer sending end power in MW.

Return type
float

GetSendReactivePowerMVAr() → **float**

Returns the transformer sending end power in MVAr.

Returns
The transformer sending end power in MVAr.

Return type
float

GetSendRealPowerkW() → **float**

Returns the transformer sending end power in kW.

Returns

The transformer sending end power in kW.

Return type

float

GetSendReactivePowerkVAr() → **float**

Returns the transformer sending end power in kVAr.

Returns

The transformer sending end power in kVAr.

Return type

float

GetReceivePowerMagnitudeMVA() → **float**

Returns the transformer receiving end power in MVA.

Returns

The transformer receiving end power in MVA.

Return type

float

GetReceivePowerMagnitudekVA() → **float**

Returns the transformer receiving end power in kVA.

Returns

The transformer receiving end power in kVA.

Return type

float

GetReceiveRealPowerMW() → **float**

Returns the transformer receiving end power in MW.

Returns

The transformer receiving end power in MW.

Return type

float

GetReceiveReactivePowerMVAr() → **float**

Returns the transformer receiving end power in MVAr.

Returns

The transformer receiving end power in MVAr.

Return type
float

GetReceiveRealPowerkW() → **float**

Returns the transformer receiving end power in kW.

Returns
The transformer receiving end power in kW.

Return type
float

GetReceiveReactivePowerkVAr() → **float**

Returns the transformer receiving end power in kVAr.

Returns
The transformer receiving end power in kVAr.

Return type
float

GetLargestPowerMagnitudeMVA() → **float**

GetLargestPowerMagnitudeMVA(*nStudyUID: **int***) → **float**

Returns the highest transformer power in MVA.

Parameters
nStudyUID (***int***) – If supplied, the automation or contingency study UID which the results belong to.

Returns
The highest transformer power in MVA.

Return type
float

GetLargestPowerMagnitudekVA() → **float**

Returns the highest transformer power in kVA.

Returns
The highest transformer power in kVA.

Return type
float

GetLargestRealPowerMW() → **float**

Returns the highest transformer power in MW.

Returns
The highest transformer power in MW.

Return type
float

GetLargestReactivePowerMVAr() → **float**

Returns the highest transformer power in MVAr.

Returns
The highest transformer power in MVAr.

Return type
float

GetLargestRealPowerkW() → **float**

Returns the highest transformer power in kW.

Returns
The highest transformer power in kW.

Return type
float

GetLargestReactivePowerkVAr() → **float**

Returns the highest transformer power in kVAr.

Returns
The highest transformer power in kVAr.

Return type
float

GetLossesMW() → **float**

Returns the transformer losses in MW.

Returns
The transformer losses in MW.

Return type
float

GetLossesMVAr() → **float**

Returns the transformer losses in MVAr.

Returns
The transformer losses in MVAr.

Return type
float

GetLosseskW() → **float**

Returns the transformer losses in kW.

Returns
The transformer losses in kW.

Return type
float

GetLosseskVAr() → **float**

Returns the transformer losses in kVAr.

Returns
The transformer losses in kVAr.

Return type
float

GetSpecVoltagePU() → **float**

Returns the target busbar voltage in per unit.

Returns
The target busbar voltage in per unit.

Return type
float

GetActualVoltagePU() → **float**

Returns the actual busbar voltage in per unit.

Returns
The actual busbar voltage in per unit.

Return type
float

GetTapPC() → **float**

Returns the current tap position in percentage.

Returns
The current tap position in percentage.

Return type
float

GetMinTapPC() → **float**

Returns the minimum tap position in percentage.

Returns
The minimum tap position in percentage.

Return type
float

GetMaxTapPC() → **float**

Returns the maximum tap position in percentage.

Returns
The maximum tap position in percentage.

Return type
float

GetPhShiftDeg() → float

Returns the current phase shift in degrees.

Returns
The current phase shift in degrees.

Return type
float

GetPhShiftRad() → float

Returns the current phase shift in radians.

Returns
The current phase shift in radians.

Return type
float

GetHasCompounding() → bool

Returns True if the transformer has compounding, False otherwise.

Returns
True if the transformer has compounding, False otherwise.

Return type
bool

GetFaultRedComponentFromMVA() → float

Returns the red phase fault level component in MVA at the "From" end of the transformer.

Returns
The red phase fault level component in MVA at the "From" end of the transformer.

Return type
float

GetFaultRedComponentToMVA() → float

Returns the red phase fault level component in MVA at the "To" end of the transformer.

Returns
The red phase fault level component in MVA at the "To" end of the transformer.

Return type
float

GetFaultYellowComponentFromMVA() → **float**

Returns the yellow phase fault level component in MVA at the "From" end of the transformer.

Returns
The yellow phase fault level component in MVA at the "From" end of the transformer.

Return type
float

GetFaultYellowComponentToMVA() → **float**

Returns the yellow phase fault level component in MVA at the "To" end of the transformer.

Returns
The yellow phase fault level component in MVA at the "To" end of the transformer.

Return type
float

GetFaultBlueComponentFromMVA() → **float**

Returns the blue phase fault level component in MVA at the "From" end of the transformer.

Returns
The blue phase fault level component in MVA at the "From" end of the transformer.

Return type
float

GetFaultBlueComponentToMVA() → **float**

Returns the blue phase fault level component in MVA at the "To" end of the transformer.

Returns
The blue phase fault level component in MVA at the "To" end of the transformer.

Return type
float

GetFaultRedComponentFromkA() → **float**

Returns the red phase fault level component in kA at the "From" end of the transformer.

Returns
The red phase fault level component in kA at the "From" end of the transformer.

Return type
float

GetFaultRedComponentTokA() → **float**

Returns the red phase fault level component in kA at the "To" end of the transformer.

Returns
The red phase fault level component in kA at the "To" end of the transformer.

Return type
float

GetFaultYellowComponentFromkA() → **float**

Returns the yellow phase fault level component in kA at the "From" end of the transformer.

Returns
The yellow phase fault level component in kA at the "From" end of the transformer.

Return type
float

GetFaultYellowComponentTokA() → **float**

Returns the yellow phase fault level component in kA at the "To" end of the transformer.

Returns
The yellow phase fault level component in kA at the "To" end of the transformer.

Return type
float

GetFaultBlueComponentFromkA() → **float**

Returns the blue phase fault level component in kA at the "From" end of the transformer.

Returns
The blue phase fault level component in kA at the "From" end of the transformer.

Return type
float

GetFaultBlueComponentTokA() → **float**

Returns the blue phase fault level component in kA at the "To" end of the transformer.

Returns
The blue phase fault level component in kA at the "To" end of the transformer.

Return type
float

GetFaultPositiveComponentFromMVA() → **float**

Returns the positive sequence fault level component in MVA at the "From" end of the transformer.

Returns
The positive sequence fault level component in MVA at the "From" end of the transformer.

Return type
float

GetFaultPositiveComponentToMVA() → **float**

Returns the positive sequence fault level component in MVA at the "To" end of the transformer.

Returns
The positive sequence fault level component in MVA at the "To" end of the transformer.

Return type
float

GetFaultNegativeComponentFromMVA() → **float**

Returns the negative sequence fault level component in MVA at the "From" end of the transformer.

Returns
The negative sequence fault level component in MVA at the "From" end of the transformer.

Return type
float

GetFaultNegativeComponentToMVA() → **float**

Returns the negative sequence fault level component in MVA at the "To" end of the transformer.

Returns
The negative sequence fault level component in MVA at the "To" end

of the transformer.

Return type
float

GetFaultZeroComponentFromMVA() → **float**

Returns the zero sequence fault level component in MVA at the "From" end of the transformer.

Returns
The zero sequence fault level component in MVA at the "From" end of the transformer.

Return type
float

GetFaultZeroComponentToMVA() → **float**

Returns the zero sequence fault level component in MVA at the "To" end of the transformer.

Returns
The zero sequence fault level component in MVA at the "To" end of the transformer.

Return type
float

GetFaultPositiveComponentFromkA() → **float**

Returns the positive sequence fault level component in kA at the "From" end of the transformer.

Returns
The positive sequence fault level component in kA at the "From" end of the transformer.

Return type
float

GetFaultPositiveComponentTokA() → **float**

Returns the positive sequence fault level component in kA at the "To" end of the transformer.

Returns
The positive sequence fault level component in kA at the "To" end of the transformer.

Return type
float

GetFaultNegativeComponentFromkA() → **float**

Returns the negative sequence fault level component in kA at the "From" end of the transformer.

Returns
The negative sequence fault level component in kA at the "From" end of the transformer.

Return type
float

GetFaultNegativeComponentTokA() → **float**

Returns the negative sequence fault level component in kA at the "To" end of the transformer.

Returns
The negative sequence fault level component in kA at the "To" end of the transformer.

Return type
float

GetFaultZeroComponentFromkA() → **float**

Returns the zero sequence fault level component in kA at the "From" end of the transformer.

Returns
The zero sequence fault level component in kA at the "From" end of the transformer.

Return type
float

GetFaultZeroComponentTokA() → **float**

Returns the zero sequence fault level component in kA at the "To" end of the transformer.

Returns
The zero sequence fault level component in kA at the "To" end of the transformer.

Return type
float

GetFaultRedComponentFromAngleDeg() → **float**

Returns the red phase component of fault angle in degrees at the "From" end of the transformer.

Returns
The red phase component of fault angle in degrees at the "From" end

of the transformer.

Return type
float

GetFaultRedComponentToAngleDeg() → **float**

Returns the red phase component of fault angle in degrees at the "To" end of the transformer.

Returns
The red phase component of fault angle in degrees at the "To" end of the transformer.

Return type
float

GetFaultYellowComponentFromAngleDeg() → **float**

Returns the yellow phase component of fault angle in degrees at the "From" end of the transformer.

Returns
The yellow phase component of fault angle in degrees at the "From" end of the transformer.

Return type
float

GetFaultYellowComponentToAngleDeg() → **float**

Returns the yellow phase component of fault angle in degrees at the "To" end of the transformer.

Returns
The yellow phase component of fault angle in degrees at the "To" end of the transformer.

Return type
float

GetFaultBlueComponentFromAngleDeg() → **float**

Returns the blue phase component of fault angle in degrees at the "From" end of the transformer.

Returns
The blue phase component of fault angle in degrees at the "From" end of the transformer.

Return type
float

GetFaultBlueComponentToAngleDeg() → **float**

Returns the blue phase component of fault angle in degrees at the “To” end of the transformer.

Returns
The blue phase component of fault angle in degrees at the “To” end of the transformer.

Return type
float

GetFaultPositiveComponentFromAngleDeg() → **float**

Returns the positive sequence component of fault angle in degrees at the “From” end of the transformer.

Returns
The positive sequence component of fault angle in degrees at the “From” end of the transformer.

Return type
float

GetFaultPositiveComponentToAngleDeg() → **float**

Returns the positive sequence component of fault angle in degrees at the “To” end of the transformer.

Returns
The positive sequence component of fault angle in degrees at the “To” end of the transformer.

Return type
float

GetFaultNegativeComponentFromAngleDeg() → **float**

Returns the negative sequence component of fault angle in degrees at the “From” end of the transformer.

Returns
The negative sequence component of fault angle in degrees at the “From” end of the transformer.

Return type
float

GetFaultNegativeComponentToAngleDeg() → **float**

Returns the negative sequence component of fault angle in degrees at the “To” end of the transformer.

Returns
The negative sequence component of fault angle in degrees at the

"To" end of the transformer.

Return type
float

GetFaultZeroComponentFromAngleDeg() → **float**

Returns the zero sequence component of fault angle in degrees at the "From" end of the transformer.

Returns
The zero sequence component of fault angle in degrees at the "From" end of the transformer.

Return type
float

GetFaultZeroComponentToAngleDeg() → **float**

Returns the zero sequence component of fault angle in degrees at the "To" end of the transformer.

Returns
The zero sequence component of fault angle in degrees at the "To" end of the transformer.

Return type
float

GetCurrentMagnitude*(dOrder: **float**)* → **float**

Returns the current magnitude in per unit on the network base for the harmonic order.

Parameters
dOrder (***float***) – The harmonic order.

Returns
The current magnitude in per unit.

Return type
float

GetCurrentAngle*(dOrder: **float**)* → **float**

Returns the current angle in radians for the harmonic order.

Parameters
dOrder (***float***) – The harmonic order.

Returns
The current angle in radians.

Return type
float

*GetResistance(dOrder: **float**)* → **float**

Returns the transformer harmonic resistance in per unit on the network base for the harmonic order.

Parameters

dOrder (***float***) – The harmonic order.

Returns

The transformer harmonic resistance in per unit.

Return type

float

*GetReactance(dOrder: **float**)* → **float**

Returns the transformer harmonic reactance in per unit on the network base for the harmonic order.

Parameters

dOrder (***float***) – The harmonic order.

Returns

The transformer harmonic reactance in per unit.

Return type

float

*GetShuntResistance(dOrder: **float**)* → **float**

Returns the transformer harmonic shunt resistance in per unit on the network base for the harmonic order.

Parameters

dOrder (***float***) – The harmonic order.

Returns

The transformer shunt resistance in per unit.

Return type

float

*GetShuntReactance(dOrder: **float**)* → **float**

Returns the transformer harmonic shunt reactance in per unit on the network base for the harmonic order.

Parameters

dOrder (***float***) – The harmonic order.

Returns

The transformer shunt reactance in per unit.

Return type

float

GetProfileMinimumFlowMVA() → **float**

Returns the minimum branch flow in MVA from the profile study results.

Returns
The minimum branch flow in MVA from the profile study results.

Return type
float

GetProfileMinimumFlowkA() → **float**

Returns the minimum branch flow in kA from the profile study results.

Returns
The minimum branch flow in kA from the profile study results.

Return type
float

GetProfileMaximumFlowMVA() → **float**

Returns the maximum branch flow in MVA from the profile study results.

Returns
The maximum branch flow in MVA from the profile study results.

Return type
float

GetProfileMaximumFlowkA() → **float**

Returns the maximum branch flow in kA from the profile study results.

Returns
The maximum branch flow in kA from the profile study results.

Return type
float

GetProfileMedianFlowMVA() → **float**

Returns the median of the branch flow in MVA from the profile study results.

Returns
The median of the branch flow in MVA from the profile study results.

Return type
float

GetProfileMedianFlowkA() → **float**

Returns the median of the branch flow in kA from the profile study results.

Returns
The median of the branch flow in kA from the profile study results.

Return type
float

GetMinimumProfileIndex() → **int**

Returns the category index which identifies the minimum branch flow from the profile study results.

Returns
The minimum category index.

Return type
int

GetMaximumProfileIndex() → **int**

Returns the category index which identifies the maximum branch flow from the profile study results.

Returns
The maximum category index.

Return type
int

GetDCLFSendPowerMagnitudeMVA() → **float**

Returns the transformer sending end power in MVA.

Returns
The transformer sending end power in MVA.

Return type
float

GetDCLFSendPowerMagnitudekVA() → **float**

Returns the transformer sending end power in kVA.

Returns
The transformer sending end power in kVA.

Return type
float

GetDCLFSendRealPowerMW() → **float**

Returns the transformer sending end power in MW.

Returns
The transformer sending end power in MW.

Return type
float

GetDCLFSendRealPowerkW() → **float**

Returns the transformer sending end power in kW.

Returns

The transformer sending end power in kW.

Return type

float

GetDCLFReceivePowerMagnitudeMVA() → float

Returns the transformer receiving end power in MVA.

Returns

The transformer receiving end power in MVA.

Return type

float

GetDCLFReceivePowerMagnitudekVA() → float

Returns the transformer receiving end power in kVA.

Returns

The transformer receiving end power in kVA.

Return type

float

GetDCLFReceiveRealPowerMW() → float

Returns the transformer receiving end power in MW.

Returns

The transformer receiving end power in MW.

Return type

float

GetDCLFReceiveRealPowerkW() → float

Returns the transformer receiving end power in kW.

Returns

The transformer receiving end power in kW.

Return type

float

GetDCLFReceiveReactivePowerkVAr() → float

Returns the transformer receiving end power in kVAr.

Returns

The transformer receiving end power in kVAr.

Return type

float

GetDCLFLargestPowerMagnitudeMVA() → **float**

Returns the highest transformer end power in MVA.

Returns
The highest transformer end power in MVA.

Return type
float

GetDCLFLargestPowerMagnitudekVA() → **float**

Returns the highest transformer end power in kVA.

Returns
The highest transformer end power in kVA.

Return type
float

GetDCLFLargestRealPowerMW() → **float**

Returns the highest transformer end power in MW.

Returns
The highest transformer end power in MW.

Return type
float

GetDCLFLargestRealPowerkW() → **float**

Returns the highest transformer end power in kW.

Returns
The highest transformer end power in kW.

Return type
float

GetDCLFLossesMW() → **float**

Returns the transformer losses in MW.

Returns
The transformer losses in MW.

Return type
float

GetDCLFLosseskW() → **float**

Returns the transformer losses in kW.

Returns
The transformer losses in kW.

Return type
float

GetDCLFPhShiftDeg() → **float**

Returns the transformer phase shift in degrees.

Returns
The transformer phase shift in degrees.

Return type
float

GetDCLFPhShiftRad() → **float**

Returns the transformer phase shift in radians.

Returns
The transformer phase shift in radians.

Return type
float

1.13 Isc3WTransformer

The *Isc3WTransformer* class provides access to an IPSA 3-winding transformer, to set and get data values and to retrieve load flow and fault level results. In the following functions and field values the following conventions are used;

- Primary winding = winding 1. Winding number *nWinding = 1*
- Secondary winding = winding 2. Winding number *nWinding = 2*
- Tertiary winding = winding 3. Winding number *nWinding = 3*

1.13.1 Field Values

Table 11: **Isc3WTransformer Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID of the primary winding busbar.
Integer	ToUID	Gets the unique ID of the secondary winding busbar.
Integer	ThreeUID	Gets the unique ID of the tertiary winding busbar.
String	FromBusName	Gets the primary winding busbar name.
String	ToBusName	Gets the secondary winding busbar name.
String	ThreeBusName	Gets the tertiary winding busbar name.

continues on next page

Table 11 – continued from previous page

Type	Field Name	Description
String	Name	Gets the 3-winding transformer name.
Integer	Status	Status: • 0 = All windings switched in • -1 = All windings switched out

continues on next page

Table 11 – continued from previous page

Type	Field Name	Description
Integer	Winding / Vector-Group	Transformer winding/VG type connection as follows: • 1 = none • 2 = xd1d1 • 3 = xd1d11 • 4 = xd11d1 • 5 = xd11d11 • 6 = xxd1 • 7 = xxd11 • 8 = xd1x • 9 = xd11x • 10 = xyd1 • 11 = xyd11 • 12 = xd1y • 13 = xd11y • 14 = ddx1 • 15 = ddx11 • 16 = dx1d • 17 = dx11d • 18 = ddy1 • 19 = ddy11 • 20 = dy1d • 21 = dy11d • 22 = dx1x1 • 23 = dx11x11 • 24 = dy1y1 • 25 = dy11y11 • 26 = dx1y1 • 27 = dx11y11 • 28 = dy1x1 • 29 = dy11x11 • 30 = yd1d1 • 31 = yd1d11 • 32 = yd11d1 • 33 = yd11d11 • 34 = yyd1 • 35 = yyd11 • 36 = yd1y • 37 = yd11y • 38 = yxd1 • 39 = yxd11
		• 40 = yd1x • 41 = yd11x • 42 = xxy • 43 = xyx

Table 11 – continued from previous page

Type	Field Name	Description
Float	W1W2ResistancePU	Gets and sets the winding 1 to winding 2 resistance in per unit.
Float	W1W2ReactancePU	Gets and sets the winding 1 to winding 2 reactance in per unit.
Float	W1W3ResistancePU	Gets and sets the winding 1 to winding 3 resistance in per unit.
Float	W1W3ReactancePU	Gets and sets the winding 1 to winding 3 reactance in per unit.
Float	W2W3ResistancePU	Gets and sets the winding 2 to winding 3 reactance in per unit.
Float	W2W3ReactancePU	Gets and sets the winding 2 to winding 3 reactance in per unit.
Float	W1W2ZSResistanceP	Gets and sets the winding 1 to winding 2 zero sequence resistance in per unit.
Float	W1W2ZSReactancePU	Gets and sets the winding 1 to winding 2 zero sequence reactance in per unit.
Float	W1W3ZSResistanceP	Gets and sets the winding 1 to winding 3 zero sequence resistance in per unit.
Float	W1W3ZSReactancePU	Gets and sets the winding 1 to winding 3 zero sequence reactance in per unit.
Float	W2W3ZSResistanceP	Gets and sets the winding 2 to winding 3 zero sequence resistance in per unit.
Float	W2W3ZSReactancePU	Gets and sets the winding 2 to winding 3 zero sequence reactance in per unit.
Float	W1NEResistancePU	Gets and sets the winding 1 neutral earth resistance in per unit.
Float	W1NEReactancePU	Gets and sets the winding 1 neutral earth reactance in per unit.
Float	W2NEResistancePU	Gets and sets the winding 2 neutral earth resistance in per unit.
Float	W2NEReactancePU	Gets and sets the winding 2 neutral earth reactance in per unit.
Float	W3NEResistancePU	Gets and sets the winding 3 neutral earth resistance in per unit.
Float	W3NEReactancePU	Gets and sets the winding 3 neutral earth reactance in per unit.
Boolean	LockTap	Gets and sets the flag to lock the tap changer on the primary winding.

continues on next page

Table 11 – continued from previous page

Type	Field Name	Description
Float	W1TapNominalPC	Gets and sets the winding 1 nominal tap position in percent, optionally used in a flat start.
Float	W2TapNominalPC	Gets and sets the winding 2 nominal tap position in percent, optionally used in a flat start.
Float	W3TapNominalPC	Gets and sets the winding 3 nominal tap position in percent, optionally used in a flat start.
Float	W1TapStartPC	Gets and sets the winding 1 tap position in percent, used as a starting point for the next load flow.
Float	W2TapStartPC	Gets and sets the winding 2 tap position in percent, used as a starting point for the next load flow.
Float	W3TapStartPC	Gets and sets the winding 3 tap position in percent, used as a starting point for the next load flow.
Float	W1MinTapPC	Gets and sets the winding 1 minimum tap position in percent, normally negative or zero.
Float	W2MinTapPC	Gets and sets the winding 2 minimum tap position in percent, normally negative or zero.
Float	W3MinTapPC	Gets and sets the winding 3 minimum tap position in percent, normally negative or zero.
Float	W1TapStepPC	Gets and sets the winding 1 tap increment in percent. This defaults to 0.01 if left blank.
Float	W2TapStepPC	Gets and sets the winding 2 tap increment in percent. This defaults to 0.01 if left blank.
Float	W3TapStepPC	Gets and sets the winding 3 tap increment in percent. This defaults to 0.01 if left blank.
Float	W1MaxTapPC	Gets and sets the winding 1 maximum tap position in percent, normally positive or zero.
Float	W2MaxTapPC	Gets and sets the winding 2 maximum tap position in percent, normally positive or zero.
Float	W3MaxTapPC	Gets and sets the winding 3 maximum tap position in percent, normally positive or zero.
Float	W1SpecVPU	Gets and sets the winding 1 target voltage in per unit. Positive values only. Magnitudes of less than 0.5 pu mean fixed tap operation.
Integer	W1SpecVWinding	Specifies the busbar whose voltage is controlled by the tap changer on winding 1.
Float	W1RBWidthPC	Full bandwidth of the winding 1 voltage sensing relay. This should be larger than tap step size.
Float	W1CompRPC	Winding 1 line drop compensation resistance in percentage on the compensation rating base.

continues on next page

Table 11 – continued from previous page

Type	Field Name	Description
Float	W1CompXPC	Winding 1 line drop compensation reactance in percentage on the compensation rating base.
Float	W1CompRatingMVA	Winding 1 line drop compensation rating in MVA used to provide load compensation.
Float	VoltFactorPt	Sets the voltage factor for use in IEC60909 fault calculations.
Integer	HarmonicModel	Transformer harmonic model. One of the following: • 0 = Polynomial resistance mode • 1 = Resistance square root model • 2 = Constant X/R model
Float	HarmRC0 HarmRC12 HarmRC1 HarmRC2 HarmRC3	Harmonic polynomial constants RC0, RC12, RC1, RC2 and RC3 in: $R_h = R[RC0{+}RC12.h^{0.5}0{+}RC1.h{+}RC2.h^2{+}RC3.h^3]$
Float	FailureRateYr	3W transformer failure rate per annum.
Float	RepairTimeHr	3W transformer repair time in hours.

1.13.2 Isc3WTransformer Class

class *ipsa.**Isc3WTransformer***

Provides access to an IPSA 3-winding transformer.

SetName(*strName: **str***) → **bool**

Sets the name as a string.

Parameters
strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue(*nFieldIndex: **int***) → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: **int**, strValue: **int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: **int**, bValue: **bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns
True if successful.

Return type
bool

GetWindingRatingMVA(*nWinding: **int**, nRatingIndex: **int***) → **float**

Returns the MVA rating for the 3-winding transformer for the specified rating set.

Parameters

- **nWinding** (***int***) – The winding number.

- **nRatingIndex** (*int*) – The specified rating index.

Returns
The MVA rating for the 3-winding transformer.

Return type
float

SetWindingRatingMVA(nSection: int, nRatingIndex: int, dRatingMVA: float) → None

Sets the MVA rating to dRatingMVA for the specified rating set.

Parameters

- **nSection** (*int*) – The number of sections.
- **nRatingIndex** (*int*) – The specified rating index.
- **dRatingMVA** (*float*) – The MVA rating that is set.

GetLargestPowerMagnitudeMVA() → float

Returns the highest 3-winding transformer end power in MVA.

Returns
The highest 3-winding transformer end power in MVA.

Return type
float

GetLargestPowerMagnitudekVA() → float

Returns the highest 3-winding transformer end power in kVA.

Returns
The highest 3-winding transformer end power in kVA.

Return type
float

GetLargestRealPowerMW() → float

Returns the highest 3-winding transformer end power in MW.

Returns
The highest 3-winding transformer end power in MW.

Return type
float

GetLargestReactivePowerMVAr() → float

Returns the highest 3-winding transformer end power in MVAr.

Returns
The highest 3-winding transformer end power in MVAr.

Return type
float

GetLargestRealPowerkW() → **float**

Returns the highest 3-winding transformer end power in kW.

Returns
The highest 3-winding transformer end power in kW.

Return type
float

GetLargestReactivePowerkVAr() → **float**

Returns the highest 3-winding transformer end power in kVAr.

Returns
The highest 3-winding transformer end power in kVAr.

Return type
float

GetLossesMW() → **float**

Returns the 3-winding transformer losses in MW.

Returns
The 3-winding transformer losses in MW.

Return type
float

GetLossesMVAr() → **float**

Returns the 3-winding transformer losses in MVAr.

Returns
The 3-winding transformer losses in MVAr.

Return type
float

GetLosseskW() → **float**

Returns the 3-winding transformer losses in kW.

Returns
The 3-winding transformer losses in kW.

Return type
float

GetLosseskVAr() → **float**

Returns the 3-winding transformer losses in kVAr.

Returns
The 3-winding transformer losses in kVAr.

Return type
float

GetWindingPowerMagnitudeMVA(nWinding: int) → **float**

Returns the MVA power flow in the specified winding for the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The MVA power flow in the specified winding for the 3-winding transformer.

Return type
float

GetWindingPowerMagnitudekVA(nWinding: int) → **float**

Returns the kVA power flow in the specified winding for the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The kVA power flow in the specified winding for the 3-winding transformer.

Return type
float

GetWindingRealPowerMW(nWinding: int) → **float**

Returns the MW power flow in the specified winding for the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The MW power flow in the specified winding for the 3-winding transformer.

Return type
float

GetWindingReactivePowerMVAr(nWinding: int) → **float**

Returns the MVAr power flow in the specified winding for the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The MVAr power flow in the specified winding for the 3-winding transformer.

Return type
float

GetWindingRealPowerkW(*nWinding: **int***) → **float**

Returns the kW power flow in the specified winding for the 3-winding transformer.

Parameters
nWinding (***int***) – The winding number.

Returns
The kW power flow in the specified winding for the 3-winding transformer.

Return type
float

GetWindingReactivePowerkVAr(*nWinding: **int***) → **float**

Returns the kVAr power flow in the specified winding for the 3-winding transformer.

Parameters
nWinding (***int***) – The winding number.

Returns
The kVAr power flow in the specified winding for the 3-winding transformer.

Return type
float

GetFaultRedComponentMVA(*nWinding: **int***) → **float**

Returns the red phase fault level component in MVA for the specified winding of the 3-winding transformer.

Parameters
nWinding (***int***) – The winding number.

Returns
The red phase fault level component in MVA for the specified winding of the 3-winding transformer.

Return type
float

GetFaultYellowComponentMVA(*nWinding: **int***) → **float**

Returns the yellow phase fault level component in MVA for the specified winding of the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The yellow phase fault level component in MVA for the specified winding of the 3-winding transformer.

Return type
float

GetFaultBlueComponentMVA(*nWinding: int*) → **float**

Returns the blue phase fault level component in MVA for the specified winding of the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The blue phase fault level component in MVA for the specified winding of the 3-winding transformer.

Return type
float

GetFaultPositiveComponentMVA(*nWinding: int*) → **float**

Returns the positive sequence fault level component in MVA for the specified winding of the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The positive sequence fault level component in MVA for the specified winding of the 3-winding transformer.

Return type
float

GetFaultNegativeComponentMVA(*nWinding: int*) → **float**

Returns the negative sequence fault level component in MVA for the specified winding of the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The negative sequence fault level component in MVA for the specified

winding of the 3-winding transformer.

Return type
float

GetFaultZeroComponentMVA(*nWinding: int*) → **float**

Returns the zero sequence fault level component in MVA for the specified winding of the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The zero sequence fault level component in MVA for the specified winding of the 3-winding transformer.

Return type
float

GetCurrentMagnitude(*nWinding: int, dOrder: float*) → **float**

Returns the current magnitude for the specified winding in per unit on the network base for the harmonic order.

Parameters

- **nWinding** (*int*) – The winding number.
- **dOrder** (*float*) – The harmonic order.

Returns
The current magnitude in per unit.

Return type
float

GetCurrentAngle(*nWinding: int, dOrder: float*) → **float**

Returns the current angle magnitude for the specified winding in radians for the harmonic order.

Parameters

- **nWinding** (*int*) – The winding number.
- **dOrder** (*float*) – The harmonic order.

Returns
The current angle magnitude in radians.

Return type
float

*GetResistance(nWinding: **int**, dOrder: **float**)* → **float**

Returns the transformer harmonic resistance for the specified winding in per unit on the network base for the harmonic order.

Parameters

- **nWinding** (*int*) – The winding number.
- **dOrder** (*float*) – The harmonic order.

Returns

The transformer harmonic resistance in per unit.

Return type

float

*GetReactance(nWinding: **int**, dOrder: **float**)* → **float**

Returns the transformer harmonic reactance for the specified winding in per unit on the network base for the harmonic order.

Parameters

- **nWinding** (*int*) – The winding number.
- **dOrder** (*float*) – The harmonic order.

Returns

The transformer harmonic reactance in per unit.

Return type

float

GetDCLFLargestPowerMagnitudeMVA() → **float**

Returns the highest 3-winding transformer end power in MVA.

Returns

The highest 3-winding transformer end power in MVA.

Return type

float

GetDCLFLargestPowerMagnitudekVA() → **float**

Returns the highest 3-winding transformer end power in kVA.

Returns

The highest 3-winding transformer end power in kVA.

Return type

float

GetDCLFLargestRealPowerMW() → **float**

Returns the highest 3-winding transformer end power in MW.

Returns
The highest 3-winding transformer end power in MW.

Return type
float

GetDCLFLargestRealPowerkW() → **float**

Returns the highest 3-winding transformer end power in kW.

Returns
The highest 3-winding transformer end power in kW.

Return type
float

GetDCLFLossesMW() → **float**

Returns the 3-winding transformer losses in MW.

Returns
The 3-winding transformer losses in MW.

Return type
float

GetDCLFLosseskW() → **float**

Returns the 3-winding transformer losses in kW.

Returns
The 3-winding transformer losses in kW.

Return type
float

GetDCLFWindingPowerMagnitudeMVA(nWinding: int) → **float**

Returns the MVA power flow in winding for the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The MVA power flow in winding for the 3-winding transformer.

Return type
float

GetDCLFWindingPowerMagnitudekVA(nWinding: int) → **float**

Returns the kVA power flow in winding for the 3-winding transformer.

Parameters
nWinding (*int*) – The winding number.

Returns
The kVA power flow in winding for the 3-winding transformer.

Return type
float

GetDCLFWindingRealPowerMW(*nWinding*: ***int***) → **float**

Returns the MW power flow in winding for the 3-winding transformer.

Parameters
nWinding (***int***) – The winding number.

Returns
The MW power flow in winding for the 3-winding transformer.

Return type
float

GetDCLFWindingRealPowerkW(*nWinding*: ***int***) → **float**

Returns the kW power flow in winding for the 3-winding transformer.

Parameters
nWinding (***int***) – The winding number.

Returns
The kW power flow in winding for the 3-winding transformer.

Return type
float

1.14 IscLoad

The *IscLoad* class provides access to an IPSA load, to set and get data values and to retrieve load flow and fault level results.

1.14.1 Field Values

Table 12: **IscLoad Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the load name.

continues on next page

Table 12 – continued from previous page

Type	Field Name	Description
Integer	Status	Status: • 0 = Switched in • 1 = Switched out
Float	RealMW	Gets and sets the real power output in MW.
Float	ReactiveMVAr	Gets and sets the reactive power output in MVAr.
Integer	ProfileUID	Sets and gets the load profile UID for the load.
Integer	Customers	Sets and gets the number of customers that this load represents. Used for reliability analysis.
Integer	CustomerType	Sets and gets the customer type that this load represents.
Boolean	IsEquivalent	If *True* then the load is an equivalent load.
Integer	LoadPlanningStage	The stage the load is currently at: • 0 = Proposed • 1 = Accepted • 2 = Completed • 3 = Energised (default, in service)

1.14.2 IscLoad Class

class *ipsa.***IscLoad**

Provides access to an IPSA load.

SetName(*strName: **str***) → **bool**

Sets the name as a string.

Parameters
strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue(*nFieldIndex: **int***) → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

*SetDValue(nFieldIndex: **int**, dValue: **float**)* → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

*SetSValue(nFieldIndex: **int**, strValue: **int**)* → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

*SetBValue(nFieldIndex: **int**, bValue: **bool**)* → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

GetPowerMagnitudeMVA() → **float**

Returns the load in MVA.

Returns
The load in MVA.

Return type
float

GetPowerMagnitudekVA() → **float**

Returns the load in kVA.

Returns
The load in kVA.

Return type
float

GetRealPowerMW() → **float**

Returns the load in MW.

Returns
The load in MW.

Return type
float

GetReactivePowerMVAr() → **float**

Returns the load in MVAr.

Returns
The load in MVAr.

Return type
float

GetRealPowerkW() → **float**

Returns the load in kW.

Returns
The load in kW.

Return type
float

GetReactivePowerkVAr() → **float**

Returns the load in kVAr.

Returns
The load in kVAr.

Return type
float

*GetCurrentMagnitude(dOrder: **float**)* → **float**

Returns the current magnitude in per unit on the network base for the harmonic order.

Parameters
dOrder (***float***) – The harmonic order.

Returns
The current magnitude in per unit.

Return type
float

GetCurrentAngle*(dOrder:* ***float****)* → **float**

Returns the current angle in radians for the harmonic order.

Parameters
dOrder (***float***) – The harmonic order.

Returns
The current angle in radians.

Return type
float

GetDCLFPowerMagnitudeMVA() → **float**

Returns the load in MVA.

Returns
The load in MVA.

Return type
float

GetDCLFPowerMagnitudekVA() → **float**

Returns the load in kVA.

Returns
The load in kVA.

Return type
float

GetDCLFRealPowerMW() → **float**

Returns the load in MW.

Returns
The load in MW.

Return type
float

GetDCLFRealPowerkW() → **float**

Returns the load in kW.

Returns
The load in kW.

Return type
float

1.15 IscCircuitBreaker

The *IscCircuitBreaker* class provides access to an IPSA circuit breaker, to set and get data values. There are no analysis results associated with circuit breakers in this version.

1.15.1 Field Values

Table 13: **IscCircuitBreaker Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name nearest to the circuit breaker.
String	BranchName	Gets the branch name the circuit breaker is located on.
String	Name	Gets the circuit breaker name.
Integer	Status	Status: • 0 = Switched in • -1 = Switched out
Boolean	NOP	If *True* then the circuit breaker is normally open.
Float	MakePeakkA	Sets and gets the peak rating in kA.
Float	BreakRMSkA	Sets and gets the symmetrical break rating in kA.
Float	BreakDCPC	Sets and gets the rated percentage DC component of the device.
Float	BreakTimemS	Sets and gets the time for the break rating in milliseconds.

continues on next page

Table 13 – continued from previous page

Type	Field Name	Description
Integer	BreakerType	Sets and gets the circuit breaker type: • 0 = Circuit breaker • 1 = Isolator • 2 = Disconnector • 3 = Recloser (reliability) • 4 = Remote control switch (reliability) • 5 = Fuse (reliability)
Float	SwitchTimeHr	Sets and gets the switch time in hours, used for reliability analysis.
Float	NomCurrentkA	Sets and gets the nominal current rating in kA
String	DbType	Gets the database type.

1.15.2 IscCircuitBreaker Class

class** ipsa.**IscCircuitBreaker

Provides access to an IPSA circuit breaker.

GetBranchUID() → **int**

Returns the UID of the branch which the breaker is located on.

Returns
The UID of the branch which the breaker is located on.

Return type
int

GetBusbarUID() → **int**

Returns the UID of the busbar that the breaker is connected to. If the breaker is located on the sending end of a branch, then the UID of the sending end busbar is returned.

Returns
The UID of the busbar that the breaker is connected to.

Return type
int

***SetName**(strName: **str**)* → **bool**

Sets the name as a string.

Parameters
strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue(nFieldIndex: int) → int

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The integer value.

Return type
int

GetDValue(nFieldIndex: int) → float

Returns a double value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The double value.

Return type
float

GetSValue(nFieldIndex: int) → str

Returns a string value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The string value.

Return type
str

GetBValue(nFieldIndex: int) → bool

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: **int**, strValue: **int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: **int**, bValue: **bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns
True if successful.

Return type

bool

PopulateByDBEntry(*strBreakerDataName:* ***str***) → **bool**

Populates the object data with database information from the first database that was loaded.

Parameters

strBreakerDataName (***str***) – The break data name.

Returns

True if successful.

Return type

bool

1.16 IscIndMachine

The *IscIndMachine* class provides access to an IPSA induction machine, to set and get data values and to retrieve load flow and fault level results.

1.16.1 Field Values

Table 14: **IscIndMachine Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the induction machine name.
Integer	Status	Status: • 0 = Switched in • 1 = Switched in, but can be switched out during transient studies • -1 = Switched out, but can be switched in during transient studies
Float	MechPowerMW	Mechanical power output of motor. Use negative values for induction generators.
Float	SlipPC	Slip in %.
Float	MagnetXPU	Magnetising reactance.
Float	StatorRPU	Stator resistance.
Float	StatorXPU	Stator reactance.

continues on next page

Table 14 – continued from previous page

Type	Field Name	Description
Integer	Model	The motor model used: • 0 = Running - standstill • 1 = Inner - outer • 2 = Not used • 3 = Not used • 4 = Running - standstill DFIG with slip control • 5 = Inner - outer DFIG with slip control • 6 = Running - standstill DFIG with slip and power factor control • 7 = Inner - outer DFIG with slip and power factor control
Float	RotorRPU	Inner cage or running rotor resistance.
Float	RotorXPU	Inner cage or running rotor reactance.
Float	StandRPU	Outer cage or standstill rotor resistance.
Float	StartXPU	Outer cage or standstill rotor reactance.
Float	TSlipB	Load torque-speed coefficient B.
Float	TSlipC	Load torque-speed squared coefficient C.
Float	InertiaSec	Inertia constant in kWs / kVA.
Float	DropoffVPU	If the voltage falls below this value disconnection will begin.
Float	DropPickUpDelay-Sec	The time taken to disconnect the machine.
Float	LockTimeSec	If a machine is disconnected for this length of time it will not be reconnected.
Float	UnderspeedPU	Under speed setting in per unit.
Float	OverspeedPU	Overspeed setting in per unit.
Float	PickUpTimeSec	The time taken to reconnect the machine.
Float	PickUpVoltagePU	If the voltage in per unit rises above this value reconnection will begin.
Float	SwitchOut1Sec	Time for the first switch-out operation. If the machine is already switched out, leave this entry blank.
Float	SwitchIn1Sec	Time for first switch-in.
Float	SwitchOut2Sec	Time for second switch-out.
Float	SwitchIn2Sec	Time for second switch-in.
Integer	MaxSwitchOp	Maximum number of automatic switching operations before the machine is locked out.
Integer	DFFeedBusbar	Feed busbar UID for doubly-fed model.

continues on next page

Table 14 – continued from previous page

Type	Field Name	Description
Float	DFPowerFactor	Power factor for doubly-fed model.
Boolean	DFExportQ	If *True* then reactive power is exported by the machine, else it is imported.
Integer	DFRotorReference-Frame	Rotor reference frame, either: • 0 = Direct-Quadrature aligned to stator voltage • 1 = Real-Imaginary aligned to stator voltage
Integer	DFFaultSwitch-Mode	How the DFIG behaves when a fault is detected: • 0 = Turn off rotor, stator and turbine. • 1 = Turn off rotor, switch permanently to induction generator mode. • 2 = Turn off rotor, switch to induction generator mode, reset on time. • 3 = Turn off rotor, switch to induction generator mode, reset on current.
Float	DFFaultRotorCur-rentLimit	When DFIG rotor current exceeds this value the DFIG alters its behaviour as determined by the fault switch mode.
Float	DFFaultCrowbar-Limit	The effect of this parameter is determined by the fault switch mode. See the online help manual for details.
Float	DFFaultCrowbar-ResPU	Crowbar fault resistance (per unit on system base and rotor voltage base).
String	DbType	Gets the database type.
Integer	DbPar	Gets the number of motors in parallel.
Integer	ControlModelType	Gets the Dynamic Model type. • 0 = Built in • 1 = Plugin • 2 = UDM
Float	RatedMW	The rated MW power of the machine. Only used in IEC60909 fault analysis.
Float	RatedMVA	The rated MVA power of the machine. Only used in IEC60909 fault analysis.
Integer	PolePairs	The number of pole pairs of the machine. Only used in IEC60909 fault analysis.

continues on next page

Table 14 – continued from previous page

Type	Field Name	Description
String	ControlPluginID	Gets the control plugin ID.

1.16.2 IscIndMachine Class

class** ipsa.**IscIndMachine

Provides access to an IPSA induction machine.

***SetName**(strName: **str**)* → **bool**

Sets the name as a string.

Parameters
strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

***GetIValue**(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

***GetDValue**(nFieldIndex: **int**)* → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

***GetSValue**(nFieldIndex: **int**)* → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: **int**, strValue: **int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(nFieldIndex: int, bValue: bool) → bool

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

PopulateByDBEntry(strIMachineDataName: str, nParallel: int) → bool

Populates the object data with database information from the first database that was loaded.

Parameters

- **strIMachineDataName** (*str*) – The name of the induction machine.
- **nParallel** (*int*) – The number of parallel components.

Returns
True if successful.

Return type
bool

GetStatorPowerMagnitudeMVA() → float

Returns stator power in MVA.

Returns
The stator power in MVA.

Return type
float

GetStatorPowerMagnitudekVA() → float

Returns stator power in kVA.

Returns
The stator power in kVA.

Return type
float

GetStatorRealPowerMW() → **float**

Returns stator power in MW.

Returns
The stator power in MW.

Return type
float

GetStatorReactivePowerMVAr() → **float**

Returns stator power in MVAr.

Returns
The stator power in MVAr.

Return type
float

GetStatorRealPowerkW() → **float**

Returns stator power in kW.

Returns
The stator power in kW.

Return type
float

GetStatorReactivePowerkVAr() → **float**

Returns stator power in kVAr.

Returns
The stator power in kVAr.

Return type
float

GetRotorPowerMagnitudeMVA() → **float**

Returns rotor power in MVA.

Returns
The rotor power in MVA.

Return type
float

GetRotorPowerMagnitudekVA() → **float**

Returns rotor power in kVA.

Returns
The rotor power in kVA.

Return type
float

GetRotorRealPowerMW() → **float**

Returns rotor power in MW.

Returns
The rotor power in MW.

Return type
float

GetRotorReactivePowerMVAr() → **float**

Returns rotor power in MVAr.

Returns
The rotor power in MVAr.

Return type
float

GetRotorRealPowerkW() → **float**

Returns rotor power in kW.

Returns
The rotor power in kW.

Return type
float

GetRotorReactivePowerkVAr() → **float**

Returns rotor power in kVAr.

Returns
The rotor power in kVAr.

Return type
float

GetMechanicalRealPowerMW() → **float**

Returns mechanical shaft power in MW.

Returns
The mechanical shaft power in MW.

Return type
float

GetMechanicalRealPowerkW() → **float**

Returns mechanical shaft power in kW.

Returns
The mechanical shaft power in kW.

Return type
float

GetSlipPU() → **float**

Returns the motor slip in per unit where 0.0 is synchronous speed, -1.0 if stationary.

Returns
The motor slip in per unit.

Return type
float

GetSlipPC() → **float**

Returns the motor slip in percent where 0% is synchronous speed, -100% if stationary.

Returns
The motor slip in percent.

Return type
float

GetEfficiencyPC() → **float**

Returns the motor efficiency in percent.

Returns
The motor efficiency in percent.

Return type
float

GetPowerFactor() → **float**

Returns the operating power factor.

Returns
The operating power factor.

Return type
float

GetCurrentkA() → **float**

Returns the total current in kA.

Returns
The total current in kA.

Return type
float

GetCurrentA() → **float**

Returns the total current in Amps.

Returns
The total current in Amps.

Return type
float

GetTorqueMNm() → **float**

Returns the shaft torque in MNm.

Returns
The shaft torque in MNm.

Return type
float

GetTorquekNm() → **float**

Returns the shaft torque in kNm.

Returns
The shaft torque in kNm.

Return type
float

GetFaultRedComponentMVA() → **float**

Returns the red phase component of fault level in MVA.

Returns
The red phase component of fault level in MVA.

Return type
float

GetFaultYellowComponentMVA() → **float**

Returns the yellow phase component of fault level in MVA.

Returns
The yellow phase component of fault level in MVA.

Return type
float

GetFaultBlueComponentMVA() → **float**

Returns the blue phase component of fault level in MVA.

Returns
The blue phase component of fault level in MVA.

Return type
float

GetFaultPositiveComponentMVA() → **float**

Returns the positive sequence component of fault level in MVA.

Returns
The positive sequence component of fault level in MVA.

Return type
float

GetFaultNegativeComponentMVA() → **float**

Returns the negative sequence component of fault level in MVA.

Returns
The negative sequence component of fault level in MVA.

Return type
float

GetFaultZeroComponentMVA() → **float**

Returns the zero sequence component of fault level in MVA.

Returns
The zero sequence component of fault level in MVA.

Return type
float

*GetCurrentMagnitude(dOrder: **float**)* → **float**

Returns the current magnitude in per unit on the network base for the harmonic order.

Parameters
dOrder (***float***) – The harmonic order.

Returns
The current magnitude in per unit.

Return type
float

*GetCurrentAngle(dOrder: **float**)* → **float**

Returns the current angle in radians for the harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The current angle in radians.

Return type
float

GetDCLFStatorPowerMagnitudeMVA() → float

Returns stator power in MVA.

Returns
The stator power in MVA.

Return type
float

GetDCLFStatorPowerMagnitudekVA() → float

Returns stator power in kVA.

Returns
The stator power in kVA.

Return type
float

GetDCLFStatorRealPowerMW() → float

Returns stator power in MW.

Returns
The stator power in MW.

Return type
float

GetDCLFStatorRealPowerkW() → float

Returns stator power in kW.

Returns
The stator power in kW.

Return type
float

GetDCLFEfficiencyPC() → float

Returns the motor efficiency in percent.

Returns
The motor efficiency in percent.

Return type
float

GetDCLFCurrentkA() → **float**

Returns the total current in kA.

Returns
The total current in kA.

Return type
float

GetDCLFCurrentA() → **float**

Returns the total current in Amps.

Returns
The total current in Amps.

Return type
float

1.17 IscSynMachine

The *IscSynMachine* class provides access to an IPSA generator (or more specifically, a synchronous machine), to set and get data values and to retrieve load flow and fault level results.

1.17.1 Field Values

Table 15: **IscSynMachine Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the synchronous machine name.
Integer	Status	Status: • 0 = Switched in • -1 = Switched out
Float	VoltPU	Per unit voltage target.
Float	VoltBandwidthPC	Bandwidth of acceptable busbar voltage.
Integer	CtlBusbar	UID of controlled busbar.
Float	GenMW	Generated real power.

continues on next page

Table 15 – continued from previous page

Type	Field Name	Description
Float	GenMVAr	Generated reactive power.
Float	GenMVArMax	Maximum reactive power limit for PV control.
Float	GenMVArMin	Minimum reactive power limit for PV control.
Float	GenRatedMW	Generator rated MW.
Float	GenRatedMVA	Generator rated MVA.
Integer	ProfileUID	Gets and sets the UID identifying the profile to be applied to the synchronous machine.
Float	SynResistancePU	Positive sequence or armature resistance.
Float	SynReactancePU	Positive sequence or d-axis synchronous reactance.
Float	ZSResistancePU	Zero sequence resistance.
Float	ZSReactancePU	Zero sequence reactance.
Float	NEResistancePU	Neutral earthing resistance.
Float	NEReactancePU	Neutral earthing reactance.
Integer	WindingEarthing	Neutral earthing connection type: • 0 = Star wound, unearthed • 1 = Star wound, neutral earthed
Float	DAxisTrXPU	D-axis transient reactance.
Float	DAxisTrTCSec	D-axis transient open-circuit time constant.
Float	DAxisStrXPU	D-axis sub transient reactance.
Float	DAxisStrTCSec	D-axis sub transient open-circuit time constant.
Float	QAxisXPU	Q-axis synchronous reactance.
Float	QAxisTrXPU	Q-axis transient reactance.
Float	QAxisTrTCSec	Q-axis transient open-circuit time constant.
Float	QAxisStrXPU	Q-axis sub transient reactance.
Float	QAxisStrTCSec	Q-axis sub transient open-circuit time constant.
Float	InertiaSec	Inertia constant.
Float	DampFactor	Damping factor.
Float	PotierXPU	Potier reactance (required only if a saturation factor is entered).
Float	SaturationFact	Per unit field current required to generate 1.2 per unit voltage in open circuit.
Integer	TID	Gets the ID for two generators to share the same prime mover.
Float	PMaxMW	Maximum machine real power.
Float	SMaxMVA	Maximum machine apparent power.
Float	SatDAxisXPU	Saturated d-axis synchronous reactance.
Float	SatDAxisTrXPU	Saturated d-axis transient reactance.

continues on next page

Table 15 – continued from previous page

Type	Field Name	Description
Float	SatDAxisTrTCSec	Saturated d-axis transient open-circuit time constant.
Float	SatDAxisStTrXPU	Saturated d-axis sub transient reactance.
Float	SatDAxisStrTCSec	Saturated d-axis sub transient open-circuit time constant.
Float	SatQAxisStrXPU	Saturated q-axis sub transient reactance.
String	DbGenType	Gets the database type.
Integer	DbGenPar	Gets the number of database generators in parallel.
Boolean	EnhancedModelling	*True* to indicate if rotor field current, calculated from the leakage reactance is modelled in transient stability. *False* if the leakage reactance is not used.
Float	LeakageReactance	The leakage reactance in per unit, required for extended field modelling.
Float	VoltageFactorPg	The voltage factor (P_g) of the machine, only for use in IEC60909 fault calculations.
Float	DispPMaxPC	Maximum economic dispatch as a percentage of the machine maximum power.
Float	DispPMinPC	Minimum economic dispatch as a percentage of the machine maximum power.
Integer	GenTechnology	The specific type of generator that can be categorized: • 0 = Synchronous machine (default) • 1 = Energy storage • 2 = Solar • 3 = Wind • 4 = Hydroelectric • 5 = Nuclear • 6 = Gas • 7 = Coal • 8 = Diesel • 9 = Geothermal • 10 = Tidal • 11 = Future generation (TBC)

continues on next page

Table 15 – continued from previous page

Type	Field Name	Description
Integer	GenStage	The stage at which the generation production/planning is situated: • 0 = Proposed • 1 = Accepted • 2 = Completed • 3 = Energized (default, in service)
Float	StorageIERatio	For energy storage components, this is the ratio between where a storage unit behaves as an import or an export. If the storage is flipped from export to import, the real power is multiplied by this ratio. Default is 1.

1.17.2 IscSynMachine Class

class *ipsa.**IscSynMachine***

Provides access to an IPSA generator (or more specifically, a synchronous machine).

SetName(*strName*: ***str***) → **bool**

Sets the name as a string.

Parameters
strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue(*nFieldIndex*: ***int***) → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex:* ***int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex:* ***int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex:* ***int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex:* ***int***, *nValue:* ***int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex:* ***int***, *dValue:* ***float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(nFieldIndex: int, strValue: int) → bool

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(nFieldIndex: int, bValue: bool) → bool

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

PopulateByDBEntry(strGeneratorDataName: str, nParallel: int) → bool

Populates the object data with database information from the first database that was loaded.

Parameters

- **strGeneratorDataName** (*str*) – The name of the generator.
- **nParallel** (*int*) – The number of parallel components.

Returns
True if successful.

Return type
bool

GetVoltageMagnitudePU() → **float**

Returns the generator voltage magnitude in per unit.

Returns
The generator voltage magnitude in per unit.

Return type
float

GetVoltageAngleRad() → **float**

Returns the voltage angle in radians.

Returns
The voltage angle.

Return type
float

GetVoltageAngleDeg() → **float**

Returns the voltage angle in degrees.

Returns
The voltage angle.

Return type
float

GetPowerMagnitudeMVA() → **float**

Returns the generator output in MVA.

Returns
The generator output in MVA.

Return type
float

GetPowerMagnitudekVA() → **float**

Returns the generator output in kVA.

Returns
The generator output in kVA.

Return type
float

GetRealPowerMW() → **float**

Returns the generator output in MW.

Returns
The generator output in MW.

Return type
float

GetReactivePowerMVAr() → **float**

Returns the generator output in MVAr.

Returns
The generator output in MVAr.

Return type
float

GetRealPowerkW() → **float**

Returns the generator output in kW.

Returns
The generator output in kW.

Return type
float

GetReactivePowerkVAr() → **float**

Returns the generator output in kVAr.

Returns
The generator output in kVAr.

Return type
float

GetFaultRedComponentMVA() → **float**

Returns the red phase component of fault level in MVA.

Returns
The red phase component of fault level in MVA.

Return type
float

GetFaultYellowComponentMVA() → **float**

Returns the yellow phase component of fault level in MVA.

Returns
The yellow phase component of fault level in MVA.

Return type
float

GetFaultBlueComponentMVA() → **float**

Returns the blue phase component of fault level in MVA.

Returns
The blue phase component of fault level in MVA.

Return type
float

GetFaultPositiveComponentMVA() → **float**

Returns the positive sequence component of fault level in MVA.

Returns
The positive sequence component of fault level in MVA.

Return type
float

GetFaultNegativeComponentMVA() → **float**

Returns the negative sequence component of fault level in MVA.

Returns
The negative sequence component of fault level in MVA.

Return type
float

GetFaultZeroComponentMVA() → **float**

Returns the zero sequence component of fault level in MVA.

Returns
The zero sequence component of fault level in MVA.

Return type
float

*GetCurrentMagnitude(dOrder: **float**)* → **float**

Returns the current magnitude in per unit on the network base for the harmonic order.

Parameters
dOrder (***float***) – The harmonic order.

Returns
The current magnitude in per unit.

Return type
float

*GetCurrentAngle(dOrder: **float**)* → **float**

Returns the current angle in radians for the harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The current angle in radians.

Return type
float

GetImpedanceMagnitude(*dOrder: float*) → **float**

Returns the impedance magnitude in per unit on the network base for the harmonic order.

Parameters
dOrder (*float*) – The harmonic order.

Returns
The impedance magnitude in per unit.

Return type
float

GetDCLFRealPowerMW() → **float**

Returns the generator output in MW.

Returns
The generator output in MW.

Return type
float

GetDCLFRealPowerkW() → **float**

Returns the generator output in kW.

Returns
The generator output in kW.

Return type
float

1.18 IscGridInfeed

The *IscGridInfeed* class provides access to an IPSA grid infeed, to set and get data values and to retrieve load flow and fault level results.

1.18.1 Field Values

Table 16: **IscGridInfeed Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the grid infeed name.
Integer	Status	Status: • 0 = Switched in • -1 = Switched out
Float	VoltPU	Per unit voltage target.
Float	GenMW	Generated real power.
Float	GenMVAr	Generated reactive power.
Integer	ProfileUID	Gets or sets the profile, *ProfileUID*, to be applied to the universal machine.
Float	PeakLLL	Peak asymmetric three phase fault contribution in MVA.
Float	RMSLLL	RMS three phase fault contribution in MVA at the break time specified by *Tbreak*.
Float	X2RLLL	Three phase X / R ratio determining the DC decay. A single exponential decay is modelled.
Float	PeakLG	Peak asymmetric single phase to ground fault contribution in MVA.
Float	RMSLG	RMS single phase to ground contribution in MVA at the break time specified by *Tbreak*.
Float	Tac	AC decay time constant for three phase faults, in seconds.
Float	Tbreak	RMS fault time in seconds.
Integer	GridPlanningStage	The stage the load is currently at: • 0 = Proposed • 1 = Accepted • 2 = Completed • 3 = Energised (default, in service)

1.18.2 IscGridInfeed Class

*class ipsa.***IscGridInfeed**

Provides access to an IPSA grid infeed.

SetName(*strName: **str***) → **bool**

Sets the name as a string.

Parameters
strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue(*nFieldIndex: **int***) → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The boolean value.

Return type

bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns

True if successful.

Return type

bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns

True if successful.

Return type

bool

SetSValue(*nFieldIndex: **int**, strValue: **int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns

True if successful.

Return type
bool

SetBValue(*nFieldIndex: **int**, bValue: **bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns
True if successful.

Return type
bool

SetHarmonicR(*dictHarmonicData: **Dict[float, float]***) → **None**

Sets the values for the harmonic resistance of the grid infeed.

Parameters
dictHarmonicData (***dict(float,float)***) – Dictionary in the following format: **{double dHarmonicOrder:double dHarmonicImpedance, …}** where dHarmonicImpedance is a value specifying the harmonic resistance at the frequency given by the harmonic order dHarmonicOrder. Up to 120 different orders may be specified in each grid infeed.

SetHarmonicX(*dictHarmonicData: **Dict[float, float]***) → **None**

Sets the values for the harmonic reactance of the grid infeed.

Parameters
dictHarmonicData (***dict(float,float)***) – Dictionary in the following format: **{double dHarmonicOrder:double dHarmonicImpedance, …}** where dHarmonicImpedance is a value specifying the harmonic resistance at the frequency given by the harmonic order dHarmonicOrder. Up to 120 different orders may be specified in each grid infeed.

GetVoltageMagnitudePU() → **float**

Returns the generator voltage magnitude in per unit.

Returns
The generator voltage magnitude in per unit.

Return type
float

GetVoltageAngleRad() → **float**

Returns the voltage angle in radians.

Returns
The voltage angle.

Return type
float

GetVoltageAngleDeg() → **float**

Returns the voltage angle in degrees.

Returns
The voltage angle.

Return type
float

GetPowerMagnitudeMVA() → **float**

Returns the generator output in MVA.

Returns
The generator output in MVA.

Return type
float

GetPowerMagnitudekVA() → **float**

Returns the generator output in kVA.

Returns
The generator output in kVA.

Return type
float

GetRealPowerMW() → **float**

Returns the generator output in MW.

Returns
The generator output in MW.

Return type
float

GetReactivePowerMVAr() → **float**

Returns the generator output in MVAr.

Returns
The generator output in MVAr.

Return type
float

GetRealPowerkW() → **float**

Returns the generator output in kW.

Returns
The generator output in kW.

Return type
float

GetReactivePowerkVAr() → **float**

Returns the generator output in kVAr.

Returns
The generator output in kVAr.

Return type
float

GetFaultRedComponentMVA() → **float**

Returns the red phase component of fault level in MVA.

Returns
The red phase component of fault level in MVA.

Return type
float

GetFaultYellowComponentMVA() → **float**

Returns the yellow phase component of fault level in MVA.

Returns
The yellow phase component of fault level in MVA.

Return type
float

GetFaultBlueComponentMVA() → **float**

Returns the blue phase component of fault level in MVA.

Returns
The blue phase component of fault level in MVA.

Return type
float

GetFaultPositiveComponentMVA() → **float**

Returns the positive sequence component of fault level in MVA.

Returns
The positive sequence component of fault level in MVA.

Return type
float

GetFaultNegativeComponentMVA() → **float**

Returns the negative sequence component of fault level in MVA.

Returns
The negative sequence component of fault level in MVA.

Return type
float

GetFaultZeroComponentMVA() → **float**

Returns the zero sequence component of fault level in MVA.

Returns
The zero sequence component of fault level in MVA.

Return type
float

GetDCLFRealPowerMW() → **float**

Returns the generator output in MW.

Returns
The generator output in MW.

Return type
float

GetDCLFRealPowerkW() → **float**

Returns the generator output in kW.

Returns
The generator output in kW.

Return type
float

1.19 IscFilter

The *IscFilter* class provides access to an IPSA harmonic filter, to set and get data values and to retrieve load flow and fault level results.

1.19.1 Field Values

Table 17: **IscFilter Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the synchronous machine name.
Integer	Status	Status: • 0 = Switched in • -1 = Switched out
Integer	FilterType	Filter Type: • 1 = Single tuned • 2 = Single tuned high pass • 3 = Double tuned • 4 = C Type
Boolean	Ground	Get or set the grounded status of the filter. *True* if grounded , *False* if isolated.
Float	Data1	Get or set the resistance R1 in per unit on the system MVA.
Float	Data2	Get or set the reactance XL1 in per unit on the system MVA.
Float	Data3	Get or set the capacitance XC1 in per unit on the system MVA.
Float	Data4	Get or set the capacitance XC2 in per unit on the system MVA - double tuned and C type only.
Float	Data5	Get or set the reactance XL2 in per unit on the system MVA - double tuned only.

1.19.2 IscFilter Class

class *ipsa.***IscFilter**

Provides access to an IPSA harmonic filter.

SetName(*strName: **str***) → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue(*nFieldIndex*: ***int***) → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex*: ***int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex*: ***int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex*: ***int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns

True if successful.

Return type

bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns

True if successful.

Return type

bool

SetSValue(*nFieldIndex: **int**, strValue: **int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns

True if successful.

Return type

bool

SetBValue(*nFieldIndex: **int**, bValue: **bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns

True if successful.

Return type
bool

GetPowerMagnitudeMVA() → **float**

Returns the filter absorbed power in MVA.

Returns
The filter absorbed power in MVA.

Return type
float

GetPowerMagnitudekVA() → **float**

Returns the filter absorbed power in kVA.

Returns
The filter absorbed power in kVA.

Return type
float

GetRealPowerMW() → **float**

Returns the filter absorbed power in MW.

Returns
The filter absorbed power in MW.

Return type
float

GetReactivePowerMVAr() → **float**

Returns the filter absorbed power in MVAr.

Returns
The filter absorbed power in MVAr.

Return type
float

GetRealPowerkW() → **float**

Returns the filter absorbed power in kW.

Returns
The filter absorbed power in kW.

Return type
float

GetReactivePowerkVAr() → **float**

Returns the filter absorbed power in kVAr.

Returns
The filter absorbed power in kVAr.

Return type
float

GetCurrentMagnitude(*dOrder: **float***) → **float**

Returns the current magnitude in per unit on the network base for the harmonic order.

Parameters
dOrder (***float***) – The harmonic order.

Returns
The current magnitude in per unit.

Return type
float

GetCurrentAngle(*dOrder: **float***) → **float**

Returns the current angle in radians for the harmonic order.

Parameters
dOrder (***float***) – The harmonic order.

Returns
The current angle in radians.

Return type
float

GetDCLFRealPowerMW() → **float**

Returns the generator output in MW.

Returns
The generator output in MW.

Return type
float

GetDCLFRealPowerkW() → **float**

Returns the generator output in kW.

Returns
The generator output in kW.

Return type
float

1.20 IscHarmonic

The *IscHarmonic* class provides access to an IPSA harmonic source. Individual harmonic orders are indexed using an integer number. This corresponds to a specific harmonic order which is a float, meaning that harmonic orders may be any value as shown below:

Order Index	Harmonic Order
1	2
2	2.5
3	3.75
4	15

1.20.1 Field Values

Table 18: **IscHarmonic Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the harmonic source name.
Integer	Status	Status: • 0 = Switched in • -1 = Switched out
Integer	InjectionType	Sets and gets the harmonic injection type which is defined as follows: • 0 = Current injection • 1 = Voltage injection
Integer	ImpedanceType	Sets and gets the harmonic impedance type which is defined as follows: • 0 = Not specified • 1 = Ideal impedance • 2 = Single R and X value for all harmonic orders • 3 = User defined R and X values for each harmonic order
String	VoltageImpedanceR	Sets and gets the resistance for the harmonic impedance if *ImpedanceType* is 2.

continues on next page

Table 18 – continued from previous page

Type	Field Name	Description
String	VoltageImpedanceX	Sets and gets the reactance for the harmonic impedance if *ImpedanceType* is 2.

1.20.2 IscHarmonic Class

*class ipsa.***IscHarmonic**

Provides access to an IPSA harmonic source.

SetName(*strName: str*) → **bool**

Sets the name as a string.

Parameters
strName (*str*) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue(*nFieldIndex: int*) → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex: int*) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex: int*) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The string value.

Return type
str

GetBValue(nFieldIndex: int) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(nFieldIndex: int, nValue: int) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(nFieldIndex: int, dValue: float) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(nFieldIndex: int, strValue: int) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(nFieldIndex: int, bValue: bool) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

SetOrder(nOrderIndex: int, h: float) → **None**

Sets the harmonic order index to the selected harmonic order.

Parameters

- **nOrderIndex** (*float*) – The order index.
- **nOrderIndex** – The selected harmonic order.

GetOrder(nOrderIndex: int) → **float**

Returns the harmonic order for the order index.

Parameters
nOrderIndex (*int*) – The order index.

Returns
The harmonic order.

Return type
float

GetMagnitudePU(nOrderIndex: int) → **float**

Gets the current or voltage magnitude for the order index.

Parameters
nOrderIndex (*int*) – The order index.

Returns
The current or voltage magnitude.

Return type
float

SetMagnitudePU(nOrderIndex: int, dMagnitude: float) → None

Sets the current or voltage magnitude for the order index.

Parameters

- **nOrderIndex** (*int*) – The order index.
- **dMagnitude** (*float*) – The current or voltage magnitude.

GetAngleDeg(nOrderIndex: int) → float

Gets the current or voltage angle in degrees for the order index.

Parameters
nOrderIndex (*int*) – The order index.

Returns
The current or voltage angle in degrees.

Return type
float

SetAngleDeg(nOrderIndex: int, dAngleDeg: float) → None

Sets the current or voltage angle in degrees for the order index.

Parameters

- **nOrderIndex** (*int*) – The order index.
- **dAngleDeg** (*float*) – The current or voltage angle in degrees.

GetAngleRad(nOrderIndex: int) → float

Gets the current or voltage angle in radians for the order index.

Parameters
nOrderIndex (*int*) – The order index.

Returns
The current or voltage angle in radians.

Return type
float

SetAngleRad(nOrderIndex: int, dAngleRad: float) → None

Sets the current or voltage angle in radians for the order index.

Parameters

- **nOrderIndex** (*int*) – The order index.

- **dAngleRad** (*float*) – The current or voltage angle in radians.

GetHarmonicR() → **Dict**[**int**, **float**]

Returns a dictionary of harmonic resistances. The dictionary key values are the order indexes and the values are the harmonic resistances in per unit.

Returns
A dictionary of harmonic resistances.

Return type
dict(**int**, **float**)

GetHarmonicX() → **Dict**[**int**, **float**]

Returns a dictionary of harmonic reactances. The dictionary key values are the order indexes and the values are the harmonic reactances in per unit.

Returns
A dictionary of harmonic reactances.

Return type
dict(**int**, **float**)

SetHarmonicR(*dicHarmonic: **Dict[int, float]***) → **None**

Sets the harmonic resistances from a dictionary. The dictionary key values are the order indexes and the values are the harmonic resistances in per unit.

Parameters
dicHarmonic (***dict(int,float)***) – The harmonic resistances.

SetHarmonicX(*dicHarmonic: **Dict[int, float]***) → **None**

Sets the harmonic reactances from a dictionary. The dictionary key values are the order indexes and the values are the harmonic reactances in per unit.

Parameters
dicHarmonic (***dict(int,float)***) – The harmonic reactances.

1.21 IscStaticVC

The *IscStaticVC* class provides access to an IPSA Static VAR Compensator (SVC), to set and get data values and to retrieve load flow results.

1.21.1 Field Values

Table 19: **IscStaticVC Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the Compensator name.
Integer	Status	Status: • 0 = Switched in • -1 = Switched out
Float	QMinMVAr	Gets and sets the minimum reactive SVC output in MVAr. This corresponds to the maximum voltage setting.
Float	QMaxMVAr	Gets and sets the maximum reactive SVC output in MVAr. This corresponds to the minimum voltage setting.
Float	VminPU	Gets and sets the minimum voltage setting in per unit. This corresponds to the maximum reactive SVC output.
Float	VmaxPU	Gets and sets the maximum voltage setting in per unit. This corresponds to the minimum reactive SVC output.
Boolean	IsStatcom	*True* if the SVC is a STATCOM.

1.21.2 IscStaticVC Class

class *ipsa.***IscStaticVC**

Provides access to an IPSA Static VAR Compensator (SVC).

SetName(*strName: **str***) → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

***GetIValue**(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

***GetDValue**(nFieldIndex: **int**)* → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

***GetSValue**(nFieldIndex: **int**)* → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

***GetBValue**(nFieldIndex: **int**)* → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

***SetIValue**(nFieldIndex: **int**, nValue: **int**)* → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(nFieldIndex: int, dValue: float) → bool

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(nFieldIndex: int, strValue: int) → bool

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(nFieldIndex: int, bValue: bool) → bool

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

GetReactivePowerMVAr() → **float**

Returns the SVC produced power in MVAr.

Returns
The SVC produced power in MVAr.

Return type
float

GetReactivePowerkVAr() → **float**

Returns the SVC produced power in kVAr.

Returns
The SVC produced power in kVAr.

Return type
float

GetTotalPowerMVA() → **float**

Returns the SVC produced total power in MVA.

Returns
The SVC produced total power in MVA.

Return type
float

GetTotalPowerkVA() → **float**

Returns the SVC produced total power in kVA.

Returns
The SVC produced total power in kVA.

Return type
float

GetCurrentkA() → **float**

Returns the SVC injected current in kA.

Returns
The SVC injected current in kA.

Return type
float

1.22 IscUMachine

The *IscUMachine* class provides access to an IPSA universal machine, to set and get data values and to retrieve load flow results.

1.22.1 Field Values

Table 20: **IscUMachine Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the universal machine name.
Integer	Status	Status: • 0 = Switched in • -1 = Switched out
Float	RealMW	Gets and sets the real power output in MW.
Float	ReactiveMVAr	Gets and sets the reactive power output in MVAr.
Float	RatingMVA	Gets and sets the apparent power generated by the machine.
Integer	ProfileUID	Gets and sets the UID of the profile applied to the universal machine. Set to 0 to not use any profiles.
Integer	PluginID	Gets and sets the ID of the plugin applied to the universal machine. Set to 0 to not use any profiles.
Boolean	ConverterDriven-Plant	*True* if the universal machine is being used as a Converter Driven Plant (G74/2)
Integer	CDPMethodType	The CDP current-output mode • 0 = Simple • 1 = Advanced
Integer	CDPVoltageInter-polation	The CDP voltage interpolation scheme • 0 = Linear • 1 = Cubic
Float	CDPKFactor	The K factor co-efficient that determines the strength of the current injection contributions (only valid between 0 and 10).
Float	CDPMaxISync	Maximum synchronous value for the current injected given the time domains.

continues on next page

Table 20 – continued from previous page

Type	Field Name	Description
Float	CDPMaxITrans	Maximum transient value for the current injected given the time domains.
Float	CDPMaxISubTrans	Maximum subtransient value for the current injected given the time domains.
Float	CDPTimeConstant-TransientMs	Time constant value in ms for the transient window duration.
Float	CDPTimeCon-stantSubTran-sientMs	Time constant value in ms for the subtransient window duration.
Boolean	CDPPhaseCorrec-tions	Switch for the CDP functionality of the universal machine that forces the phase correction of the injected current to be in quadrature with the pre-fault voltage. This 'prioritises' reactive power injection at the CDP injection site. In advanced mode, when this is disabled it will adopt the phase of the active-reactive current phasor. In simple mode, when this is disabled it will be in phase with the retained voltage.

1.22.2 IscUMachine Class

class *ipsa.****IscUMachine***

Provides access to an IPSA universal machine.

SetName*(strName: **str**)* → **bool**

Sets the name as a string.

Parameters
strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue*(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex*: ***int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex*: ***int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex*: ***int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex*: ***int***, *nValue*: ***int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: int, dValue: float*) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: int, strValue: int*) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: int, bValue: bool*) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

GetRealPowerMW() → **float**

Returns the universal machine output in MW.

Returns
The universal machine output in MW.

Return type
float

GetReactivePowerMVAr() → **float**

Returns the universal machine output in MVAr.

Returns
The universal machine output in MVAr.

Return type
float

GetRealPowerkW() → **float**

Returns the universal machine output in kW.

Returns
The universal machine output in kW.

Return type
float

GetReactivePowerkVAr() → **float**

Returns the universal machine output in kVAr.

Returns
The universal machine output in kVAr.

Return type
float

GetTotalPowerMVA() → **float**

Returns the universal machine produced total power in MVA.

Returns
The universal machine produced total power in MVA.

Return type
float

GetTotalPowerkVA() → **float**

Returns the universal machine produced total power in kVA.

Returns
The universal machine produced total power in kVA.

Return type
float

GetPowerFactor() → **float**

Returns the universal machine power factor.

Returns
The universal machine power factor.

Return type
float

GetCurrentkA() → float

Returns the universal machine injected current in kA.

Returns
The universal machine injected current in kA.

Return type
float

GetDCLFRealPowerMW() → float

Returns the universal machine output in MW.

Returns
The universal machine output in MW.

Return type
float

GetDCLFRealPowerkW() → float

Returns the universal machine output in kW.

Returns
The universal machine output in kW.

Return type
float

GetDCLFTotalPowerMVA() → float

Returns the universal machine produced total power in MVA.

Returns
The universal machine produced total power in MVA.

Return type
float

GetDCLFTotalPowerkVA() → float

Returns the universal machine produced total power in kVA.

Returns
The universal machine produced total power in kVA.

Return type
float

GetDCLFCurrentkA() → **float**

Returns the universal machine injected current in kA.

Returns
The universal machine injected current in kA.

Return type
float

TransformCDPParameters(dMachineMVA: float) → **bool**

Transforms the given CDP parametrisation based on the ratio between the machine and system base. Note this function should only be used if the user has the CDP parameters in machine base.

Parameters
dMachineMVA (*float*) – Machine base in MVA

Returns
True if successful.

Return type
bool

ActivateCDP() → **bool**

Switches the CDP functionality for the given Universal Machine on

Returns
True if successful.

Return type
bool

DeactivateCDP() → **bool**

Switches the CDP functionality for the given Universal Machine off

Returns
True if successful.

Return type
bool

GetCDPVoltagePU() → **List**[**float**]

Returns the synchronous region voltages for the CDP advanced mode

Returns
List of voltage entries (PU)

Return type
list(**float**)

GetCDPVoltageTransientPU() → **List**[**float**]

Returns the transient region voltages for the CDP advanced mode

Returns
List of voltage entries (PU)

Return type
list(**float**)

GetCDPVoltageSubTransientPU() → **List**[**float**]

Returns the subtransient region voltages for the CDP advanced mode

Returns
List of voltage entries (PU)

Return type
list(**float**)

GetCDPRealCurrentPU() → **List**[**float**]

Returns the synchronous real current values for the CDP advanced mode

Returns
List of real current entries (PU)

Return type
list(**float**)

GetCDPRealCurrentTransientPU() → **List**[**float**]

Returns the transient real current values for the CDP advanced mode

Returns
List of real current entries (PU)

Return type
list(**float**)

GetCDPRealCurrentSubTransientPU() → **List**[**float**]

Returns the subtransient real current values for the CDP advanced mode

Returns
List of real current entries (PU)

Return type
list(**float**)

GetCDPReactiveCurrentPU() → **List**[**float**]

Returns the synchronous reactive current values for the CDP advanced mode

Returns
List of reactive current entries (PU)

Return type
list(**float**)

GetCDPReactiveCurrentTransientPU() → **List**[**float**]

Returns the transient reactive current values for the CDP advanced mode

Returns
List of reactive current entries (PU)

Return type
list(**float**)

GetCDPReactiveCurrentSubTransientPU() → **List**[**float**]

Returns the subtransient reactive current values for the CDP advanced mode

Returns
List of reactive current entries (PU)

Return type
list(**float**)

*SetCDPVoltagePU(lVoltage: **List**[**float**])* → **bool**

Sets the synchronous region voltages for the CDP advanced mode

Param
List of voltage entries (PU)

Type
list(**float**)

Returns
True is successful

Return type
bool

*SetCDPVoltageTransientPU(lVoltage: **List**[**float**])* → **bool**

Sets the transient region voltages for the CDP advanced mode

Param
List of voltage entries (PU)

Type
list(**float**)

Returns
True is successful

Return type
bool

*SetCDPVoltageSubTransientPU(lVoltage: **List**[**float**])* → **bool**

Sets the subtransient region voltages for the CDP advanced mode

Param
List of voltage entries (PU)

Type
list(**float**)

Returns
True is successful

Return type
bool

SetCDPRealCurrentPU(*lRealCurrent: **List[float]***) → **bool**

Sets the synchronous real current values for the CDP advanced mode

Param
List of real current entries (PU)

Type
list(**float**)

Returns
True is successful

Return type
bool

SetCDPRealCurrentTransientPU(*lRealCurrent: **List[float]***) → **bool**

Sets the transient real current values for the CDP advanced mode

Param
List of real current entries (PU)

Type
list(**float**)

Returns
True is successful

Return type
bool

SetCDPRealCurrentSubTransientPU(*lRealCurrent: **List[float]***) → **bool**

Sets the subtransient real current values for the CDP advanced mode

Param
List of real current entries (PU)

Type
list(**float**)

Returns
True is successful

Return type
bool

SetCDPReactiveCurrentPU(*lReactiveCurrent: **List[float]***) → **bool**

Sets the synchronous reactive current values for the CDP advanced mode

Param
List of reactive current entries (PU)

Type
list(**float**)

Returns
True is successful

Return type
bool

SetCDPReactiveCurrentTransientPU(*lReactiveCurrent: **List[float]***) → **bool**

Sets the transient reactive current values for the CDP advanced mode

Param
List of reactive current entries (PU)

Type
list(**float**)

Returns
True is successful

Return type
bool

SetCDPReactiveCurrentSubTransientPU(*lReactiveCurrent: **List[float]***) → **bool**

Sets the subtransient reactive current values for the CDP advanced mode

Param
List of reactive current entries (PU)

Type
list(**float**)

Returns
True is successful

Return type
bool

1.23 IscBattery

The IscBattery class provides access to an IPSA battery, to set and get data values and to retrieve load flow results.

1.23.1 Field Values

Table 21: **IscBattery Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the synchronous machine name.
Integer	Status	Status: • 0 = Switched in • -1 = Switched out
Integer	Model	Sets and gets the model type for the battery: • 0 = Voltage type • 1 = Current type
Float	VoltPU	Sets and gets the battery terminal voltage in per unit.
Float	EmfPU	Sets and gets the battery internal EMF in per unit.
Float	ResistancePU	Sets and gets the internal battery resistance in per unit.
Float	CurrentPU	Sets and gets the battery current in per unit.
Float	InductancePU	Sets and gets the internal battery inductance in per unit.

1.23.2 IscBattery Class

class** ipsa.**IscBattery

Provides access to an IPSA battery.

***SetName**(strName: **str**)* → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue(*nFieldIndex: **int***) → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex*: ***int***, *nValue*: ***int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex*: ***int***, *dValue*: ***float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex*: ***int***, *strValue*: ***int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex*: ***int***, *bValue*: ***bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns
True if successful.

Return type
bool

GetRealPowerMW() → **float**

Returns the battery output in MW.

Returns
The battery output in MW.

Return type
float

GetRealPowerkW() → **float**

Returns the battery output in kW.

Returns
The battery output in kW.

Return type
float

GetTotalPowerMVA() → **float**

Returns the battery produced total power in MVA.

Returns
The battery produced total power in MVA.

Return type
float

GetTotalPowerkVA() → **float**

Returns the battery produced total power in kVA.

Returns
The battery produced total power in kVA.

Return type
float

GetVoltagePU() → **float**

Returns the battery injected voltage in per unit.

Returns
The battery injected voltage in per unit.

Return type
float

GetCurrentPU() → **float**

Returns the battery injected current in per unit.

Returns
The battery injected current in per unit.

Return type
float

GetCurrentkA() → **float**

Returns the battery injected current in kA.

Returns
The battery injected current in kA.

Return type
float

1.24 IscDCMachine

The *IscDCMachine* class provides access to an IPSA DC machine, to set and get data values and to retrieve load flow results.

1.24.1 Field Values

Table 22: **IscDCMachine Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the DC machine name.
Integer	Status	Status: • 0 = Switched in • -1 = Switched out
Float	BusVoltagePU	Sets and gets the busbar voltage in per unit.
Float	MechPowerMW	Sets and gets the mechanical output power in MW.
Float	Efficiency	Sets and gets the machine efficiency in percent.
Float	Speed	Sets and gets the machine speed in per unit.
Float	ArmResistPU	Sets and gets the armature resistance in per unit.
Float	ShuntResisPU	Sets and gets the shunt resistance in per unit.
Float	ControlResisPU	Sets and gets the control field resistance in per unit.
Float	ShuntTRatio	Sets and gets the shunt field turns ratio.
Float	SeriesTRatio	Sets and gets the series field turns ratio.

continues on next page

Table 22 – continued from previous page

Type	Field Name	Description
Float	Compounding	Sets and gets the flag to indicate if the machine has a compound winding.
Float	SatFac75	Sets and gets the saturation factor for the MMF at 75% of flux.
Float	SatFac120	Sets and gets the saturation factor for the MMF at 120% of flux.
Float	ArmSelfIndPU	Sets and gets the armature field self-inductance in per unit.
Float	SeriesSelfIndPU	Sets and gets the series field self-inductance in per unit.
Float	ShuntSelfIndPU	Sets and gets the shunt field self-inductance in per unit.
Float	CtrlSelfIndPU	Sets and gets the control field self-inductance in per unit.
Float	LeakageIndPU	Sets and gets the shunt field leakage inductance in per unit.
Float	MechLossConst	Sets and gets the mechanical loss coefficient.
Float	InertiaSec	Sets and gets the machine inertia.
Float	TSlipB	Sets and gets the load torque slip coefficient B.
Float	TSlipC	Sets and gets the load torque slip coefficient C.

1.24.2 IscDCMachine Class

class** ipsa.**IscDCMachine

Provides access to an IPSA DC machine.

***SetName**(strName: **str**)* → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

***GetIValue**(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The integer value.

Return type
int

GetDValue(nFieldIndex: int) → float

Returns a double value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The double value.

Return type
float

GetSValue(nFieldIndex: int) → str

Returns a string value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The string value.

Return type
str

GetBValue(nFieldIndex: int) → bool

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(nFieldIndex: int, nValue: int) → bool

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue*(nFieldIndex: **int**, dValue: **float**)* → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue*(nFieldIndex: **int**, strValue: **int**)* → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue*(nFieldIndex: **int**, bValue: **bool**)* → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns
True if successful.

Return type
bool

GetRealMechanicalPowerMW() → **float**

Returns the mechanical output power of the DC machine in MW.

Returns
The mechanical output power of the DC machine in MW.

Return type
float

GetRealMechanicalPowerkW() → **float**

Returns the mechanical output power of the DC machine in kW.

Returns
The mechanical output power of the DC machine in kW.

Return type
float

GetRealElectricalPowerMW() → **float**

Returns the electrical output power of the DC machine in MW.

Returns
The electrical output power of the DC machine in MW.

Return type
float

GetRealElectricalPowerkW() → **float**

Returns the electrical output power of the DC machine in kW.

Returns
The electrical output power of the DC machine in kW.

Return type
float

GetTotalPowerMVA() → **float**

Returns the total output power of the DC machine in MVA.

Returns
The total output power of the DC machine in MVA.

Return type
float

GetTotalPowerkVA() → **float**

Returns the total output power of the DC machine in kVA.

Returns
The total output power of the DC machine in kVA.

Return type
float

GetPowerLossMW() → **float**

Returns the power loss of the DC machine in MW.

Returns
The power loss of the DC machine in MW.

Return type
float

GetPowerLosskW() → **float**

Returns the power loss of the DC machine in kW.

Returns
The power loss of the DC machine in kW.

Return type
float

GetCurrentkA() → **float**

Returns the DC machine injected current in kA.

Returns
The DC machine injected current in kA.

Return type
float

1.25 IscConverter

The *IscConverter* class provides access to an AC/DC Converter, to set and get data values and to retrieve load flow results.

1.25.1 Field Values

Table 23: **IscConverter Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID of the sending busbar.
Integer	ToUID	Gets the unique ID of the receiving busbar.
String	FromBusName	Gets the sending busbar name.
String	ToBusName	Gets the receiving busbar name.
String	Name	Gets the converter name.

continues on next page

Table 23 – continued from previous page

Type	Field Name	Description
Integer	Status	Status of converter: • 0 = Converter is switched in. • 1 = Converter is connected but only used for harmonic analysis. • -1 = Converter is switched out.
Integer	Type	The type of converter: • 0 = Thyristor or line commuted converter • 51 = PWM or voltage source converter
Float	DCPowerMW	Gets and sets the DC power in MW.
Float	DCVoltagePU	Gets and sets the DC terminal voltage in per unit.
Float	ACReactivePower-MVAr	Gets and sets the AC reactive power in MVAr.
Float	PowerFactor	Returns the operating power factor.
Float	TransEqReactan-cePU	Gets and sets the internal transformer reactance in per unit.
Float	TransOperate-TapPC	Gets and sets the operating tap position of the internal transformer in percent.
Float	MinTapPC	Gets and sets the minimum tap position of the internal transformer in percent.
Float	TapStepPC	Gets and sets the tap step of the internal transformer in percent.
Float	MaxTapPC	Gets and sets the maximum tap position of the internal transformer in percent.
Integer	PulseNumber	Gets and sets the pulse number of the converter, should be either 6, 12, 24 or 48.
Integer	NumParaBridges	Gets and sets the number of parallel bridges.
Float	CommutateRe-actPU	Gets and sets the commutation reactance in per unit.
Float	MaxACCurrentPU	Gets and sets the maximum AC current trip limit in per unit.
Float	VoltRatio	Gets and sets the voltage ratio of the internal converter transformer.
Float	FilterResisPU	Gets and sets the filter resistance in per unit.
Float	FilterInductPU	Gets and sets the filter inductance in per unit.
Float	DCEquivCapPU	Gets and sets the DC side capacitance in per unit.
Float	MinFireAngleDeg	Gets and sets the minimum firing angle in degrees.

continues on next page

Table 23 – continued from previous page

Type	Field Name	Description
Float	MaxDCCurrentPU	Gets and sets the maximum DC current limit in per unit.

1.25.2 IscConverter Class

class *ipsa.**IscConverter***

Provides access to an AC/DC Converter.

SetName(*strName:* ***str***) → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

GetIValue(*nFieldIndex:* ***int***) → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The integer value.

Return type

int

GetDValue(*nFieldIndex:* ***int***) → **float**

Returns a double value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The double value.

Return type

float

GetSValue(*nFieldIndex:* ***int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The string value.

Return type
str

GetBValue(nFieldIndex: int) → bool

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(nFieldIndex: int, nValue: int) → bool

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(nFieldIndex: int, dValue: float) → bool

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(nFieldIndex: int, strValue: int) → bool

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: **int**, bValue: **bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

GetACRealPowerMW() → **float**

Returns the AC real power output of the converter in MW.

Returns
The AC real power output of the converter in MW.

Return type
float

GetACRealPowerkW() → **float**

Returns the AC real power output of the converter in kW.

Returns
The AC real power output of the converter in kW.

Return type
float

GetACReactivePowerMVAr() → **float**

Returns the AC reactive power output of the converter in MVAr.

Returns
The AC reactive power output of the converter in MVAr.

Return type
float

GetACReactivePowerkVAr() → **float**

Returns the AC reactive power output of the converter in kVAr.

Returns
The AC reactive power output of the converter in kVAr.

Return type
float

GetACTotalPowerMVA() → **float**

Returns the total AC output power of the converter in MVA.

Returns
The total AC output power of the converter in MVA.

Return type
float

GetACTotalPowerkVA() → **float**

Returns the total AC output power of the converter in kVA.

Returns
The total AC output power of the converter in kVA.

Return type
float

GetACCurrentkA() → **float**

Returns the AC current of the converter in kA.

Returns
The AC current of the converter in kA.

Return type
float

GetDCRealPowerMW() → **float**

Returns the DC real power output of the converter in MW.

Returns
The DC real power output of the converter in MW.

Return type
float

GetDCRealPowerkW() → **float**

Returns the DC real power output of the converter in kW.

Returns
The DC real power output of the converter in kW.

Return type
float

GetDCTotalPowerMVA() → **float**

Returns the total DC output power of the converter in MVA.

Returns
The total DC output power of the converter in MVA.

Return type
float

GetDCTotalPowerkVA() → **float**

Returns the total DC output power of the converter in kVA.

Returns
The total DC output power of the converter in kVA.

Return type
float

GetDCCurrentkA() → **float**

Returns the DC current of the converter in kA.

Returns
The DC current of the converter in kA.

Return type
float

GetTapPC() → **float**

Returns the operating tap setting of the converter transformer in percentage of nominal.

Returns
The operating tap setting of the converter transformer in percentage of nominal.

Return type
float

GetFundamentalEMFMagnitude() → **float**

Returns the fundamental EMF magnitude of the converter.

Returns
The fundamental EMF magnitude of the converter.

Return type
float

GetFundamentalEMFAngle() → **float**

Returns the fundamental EMF angle of the converter.

Returns
The fundamental EMF angle of the converter.

Return type
float

GetModulationIndex() → **float**

Returns the modulation index of the converter.

Returns
The modulation index of the converter.

Return type
float

1.26 IscChopper

The *IscChopper* class provides access to a DC/DC Converter, to set and get data values and to retrieve load flow results. **Note that in IPSA, like the transformer, the chopper is modelled as a combination of a branch and a tap changer. Therefore some of the chopper data is stored in a branch instance and functions such as *GetLineDValue()* are used to access branch type data.**

1.26.1 Field Values

Table 24: **IscChopper Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID of the sending busbar.
Integer	ToUID	Gets the unique ID of the receiving busbar.
String	FromBusName	Gets the sending busbar name.
String	ToBusName	Gets the receiving busbar name.
String	Name	Gets the chopper name.
Integer	ChopperModel	The chopper gain calculation model: • 0 = Voltage gain amplification parameter. • 1 = Duty cycle parametrisation (buck-boost).
String	DispChopperModel	Gets the name of the chopper calculation model.
Float	VoltGainRatio	Gets and sets the voltage gain ratio for the ratio model.
Float	VoltGainDutyCycle	Gets and sets the duty cycle value for the duty model.

continues on next page

Table 24 – continued from previous page

Type	Field Name	Description
Float	ConductancePU	Gets and sets the per unit parallel conductive losses for the capacitor component of the chopper.
Float	ConverterEfficiencyPC	Gets and sets the efficiency of the chopper in percent.

1.26.2 IscChopper Class

class *ipsa.****IscChopper***

Provides access to a DC/DC Converter.

SetName(*strName: **str***) → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

GetIValue(*nFieldIndex: **int***) → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The integer value.

Return type

int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The double value.

Return type

float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex:* ***int***, *strValue:* ***int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex:* ***int***, *bValue:* ***bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns
True if successful.

Return type
bool

GetLineIValue(*nFieldIndex:* ***int***) → **int**

Returns an integer value for the line associated with this chopper.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The line associated with this chopper.

Return type
int

GetLineDValue(*nFieldIndex:* ***int***) → **float**

Returns a float value for the line associated with this chopper.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The line associated with this chopper.

Return type
float

GetLineSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the line associated with this chopper.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The line associated with this chopper.

Return type
str

SetLineIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets an integer value for the line associated with this chopper.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The integer value for the line.

Returns
True if successful.

Return type
bool

SetLineDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets a float value for the line associated with this chopper.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The float value for the line.

Returns
True if successful.

Return type
bool

SetLineSValue(*nFieldIndex: **int**, strValue: **str***) → **bool**

Sets a string value for the line associated with this chopper.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The string value for the line.

Returns
True if successful.

Return type
bool

SetRatingskA(*nRatingIndex:* ***int****, dSendRatingkA:* ***float****, dRecieveRatingkA:* ***float***) → **None**

Sets the sending and receiving end current ratings in kA for the chopper.

Parameters

- **nRatingIndex** (*int*) – Specifies which rating set the data is applied to.
- **dSendRatingkA** (*float*) – The sending end current ratings in kA.
- **dRecieveRatingkA** (*float*) – The receiving end current ratings in kA.

SetRatingMVA(*nRatingIndex:* ***int****, dRatingMVA:* ***float***) → **None**

Sets the MVA rating for the chopper.

Parameters

- **nRatingIndex** (*int*) – Specifies which rating set the data is applied to.
- **dRatingMVA** (*float*) – The MVA rating.

GetRatingSendkA(*nRatingIndex:* ***int***) → **float**

Returns the sending end current ratings in kA for the chopper.

Parameters
nRatingIndex (*int*) – Specifies which rating set the data is applied to.

Returns
The sending end current ratings in kA.

Return type
float

GetRatingReceivekA(*nRatingIndex:* ***int***) → **float**

Returns the receiving end current ratings in kA for the chopper.

Parameters
nRatingIndex (*int*) – Specifies which rating set the data is applied to.

Returns
The receiving end current ratings in kA.

Return type
float

GetRatingMVA(*nRatingIndex:* ***int***) → **float**

Returns the MVA rating for the chopper.

Parameters
nRatingIndex (*int*) – Specifies which rating set the data is applied to.

Returns
The MVA rating.

Return type
float

GetSendRealPowerMW() → float

Returns the chopper sending end power in MW.

Returns
The chopper sending end power in MW.

Return type
float

GetSendRealPowerkW() → float

Returns the chopper sending end power in kW.

Returns
The chopper sending end power in kW.

Return type
float

GetSendRealCurrentkA() → float

Returns the chopper sending end current in kA.

Returns
The chopper sending end current in kA.

Return type
float

GetReceiveRealPowerMW() → float

Returns the chopper receiving end power in MW.

Returns
The chopper receiving end power in MW.

Return type
float

GetReceiveRealPowerkW() → float

Returns the chopper receiving end power in kW.

Returns
The chopper receiving end power in kW.

Return type
float

GetReceiveRealCurrentkA() → **float**

Returns the chopper receiving end current in kA.

Returns
The chopper receiving end current in kA.

Return type
float

GetLargestRealPowerMW() → **float**

Returns the highest chopper end power in MW.

Returns
The highest chopper end power in MW.

Return type
float

GetLargestRealPowerkW() → **float**

Returns the highest chopper end power in kW.

Returns
The highest chopper end power in kW.

Return type
float

GetLargestRealCurrentkA() → **float**

Returns the highest chopper end current in kA.

Returns
The highest chopper end current in kA.

Return type
float

GetLossesMW() → **float**

Returns the chopper losses in MW.

Returns
The chopper losses in MW.

Return type
float

GetLosseskW() → **float**

Returns the chopper losses in kW.

Returns
The chopper losses in kW.

Return type
float

GetChopperEfficiency() → **float**

Returns the efficiency of the chopper in percent.

Returns
The efficiency of the chopper in percent.

Return type
float

GetLoadRatio() → **float**

Returns the ratio of the internal resistance to the load of the chopper for clearer visualization of buck-boost losses (fractional value).

Returns
The ratio of the internal resistance to the load of the chopper.

Return type
float

1.27 IscMGSet

The *IscMGSet* class provides access to an IPSA motor-generator set, to set and get data values and to retrieve load flow results.

1.27.1 Field Values

Table 25: **IscMGSet Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique component ID for the sending busbar.
Integer	ToUID	Gets the unique component ID for the receiving busbar.
String	FromBusName	Gets the sending busbar name.
String	ToBusName	Gets the receiving busbar name.
String	Name	Gets the MG set name.

continues on next page

Table 25 – continued from previous page

Type	Field Name	Description
Integer	Status	Status of MG set: • 0 = Switched in • -1 = Switched out

1.27.2 IscMGSet Class

class *ipsa.***IscMGSet**

Provides access to an IPSA motor-generator set.

SetName(*strName: **str***) → **bool**

Sets the name as a string.

Parameters
strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue(*nFieldIndex: **int***) → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The string value.

Return type

str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The boolean value.

Return type

bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns

True if successful.

Return type

bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns

True if successful.

Return type

bool

*SetSValue(nFieldIndex: **int**, strValue: **int**)* → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns

True if successful.

Return type

bool

*SetBValue(nFieldIndex: **int**, bValue: **bool**)* → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns

True if successful.

Return type

bool

GetACRealPowerMW() → **float**

Returns the AC real power output of the motor-generator set in MW.

Returns

The AC real power output of the motor-generator set in MW.

Return type

float

GetACRealPowerkW() → **float**

Returns the AC real power output of the motor-generator set in kW.

Returns

The AC real power output of the motor-generator set in kW.

Return type

float

GetACReactivePowerMVAr() → **float**

Returns the AC reactive power output of the motor-generator set in MVAr.

Returns

The AC reactive power output of the motor-generator set in MVAr.

Return type
float

GetACReactivePowerkVAr() → **float**

Returns the AC reactive power output of the motor-generator set in kVAr.

Returns
The AC reactive power output of the motor-generator set in kVAr.

Return type
float

GetACTotalPowerMVA() → **float**

Returns the total AC output power of the motor-generator set in MVA.

Returns
The total AC output power of the motor-generator set in MVA.

Return type
float

GetACTotalPowerkVA() → **float**

Returns the total AC output power of the motor-generator set in kVA.

Returns
The total AC output power of the motor-generator set in kVA.

Return type
float

GetACCurrentkA() → **float**

Returns the AC current of the motor-generator set in kA.

Returns
The AC current of the motor-generator set in kA.

Return type
float

GetDCRealPowerMW() → **float**

Returns the DC real power output of the motor-generator set in MW.

Returns
The DC real power output of the motor-generator set in MW.

Return type
float

GetDCRealPowerkW() → **float**

Returns the DC real power output of the motor-generator set in kW.

Returns
The DC real power output of the motor-generator set in kW.

Return type
float

GetDCTotalPowerMVA() → **float**

Returns the total DC output power of the motor-generator set in MVA.

Returns
The total DC output power of the motor-generator set in MVA.

Return type
float

GetDCTotalPowerkVA() → **float**

Returns the total DC output power of the motor-generator set in kVA.

Returns
The total DC output power of the motor-generator set in kVA.

Return type
float

GetDCCurrentkA() → **float**

Returns the DC current of the motor-generator set in kA.

Returns
The DC current of the motor-generator set in kA.

Return type
float

1.28 IscMechSwCapacitor

The *IscMechSwCapacitor* class provides access to an IPSA mechanical switched capacitor, to set and get data values and to retrieve load flow results.

1.28.1 Field Values

Table 26: **IscMechSwCapacitor Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID of the sending busbar.
String	BusName	Gets the busbar name.

continues on next page

Table 26 – continued from previous page

Type	Field Name	Description
String	Name	Gets the mechanical switched capacitor name.
Integer	Status	Status of mechanical switched capacitor: • 0 = Switched in • -1 = Switched out
Integer	ControlMode	Sets or returns the control mode of the mechanical switched capacitor: • 0 = Local voltage control • 1 = Remote busbar voltage control • 2 = Branch reactive power control entering the busbar • 3 = Branch reactive power control leaving the busbar
Integer	ControlSteps	Sets or returns the control mode of the mechanical switched capacitor: • 0 = Switch based on discrete steps • 1 = Continuous control based on min and max limits (see *MinContinuousMVAr* and *MaxContinuousMVAr*)
Integer	CapSteps	Sets or returns the number of capacitor steps.
Integer	IndSteps	Sets or returns the number of inductor steps.
Float	CapStepSizeMVAr	Sets or returns the capacitor step size in MVAr.
Float	IndStepSizeMVAr	Sets or returns the inductor step size in MVAr.
Float	TargetVoltagePU	Sets or returns the target voltage in per unit. Note when the MSC is branch reactive power controlled, this is the target power factor.
Float	BandwidthPC	Sets or returns the bandwidth of acceptable voltage in percentage.
Integer	InitPosition	Sets or returns the initial position of the mechanical switched capacitor. Positive values represent capacitor steps and negative values are inductive steps.
Float	MaxContinuousMVAr	Sets or returns the maximum MVAr output of the MSC when in continuous control mode.
Float	MinContinuousMVAr	Sets or returns the minimum MVAr output of the MSC when in continuous control mode.

continues on next page

Table 26 – continued from previous page

Type	Field Name	Description
Float	ContinuousOutput-MVAr	Sets or returns the operating output of the MSC when in continuous control mode.
Integer	ControlActive	Sets or returns the voltage or power factor control status of the MSC: • 0 = Voltage or power factor control off • 1 = Voltage or power factor control is active
Integer	ControlledUID	Sets or returns the busbar or branch UID for the remote busbar or branch whose voltage is being controlled.
Integer	SendEnd	Sets or returns the branch end that is controlled when in power factor control mode: • 0 = Control power factor at receiving end • 1 = Control power factor at send end

1.28.2 IscMechSwCapacitor Class

*class ipsa.***IscMechSwCapacitor**

Provides access to an IPSA mechanical switched capacitor.

SetName(*strName: **str***) → **bool**

Sets the name as a string.

Parameters
strName (***str***) – The selected string name.

Returns
True if successful.

Return type
bool

GetIValue(*nFieldIndex: **int***) → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

*SetDValue(nFieldIndex: **int**, dValue: **float**)* → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

*SetSValue(nFieldIndex: **int**, strValue: **int**)* → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

*SetBValue(nFieldIndex: **int**, bValue: **bool**)* → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

GetFinalPosition() → **int**

Returns the position of the MSC after a load flow. Positive values represent capacitor steps and negative values are inductive steps. See also **ContinuousOutputMVAr** for the output in continuous mode.

Returns
The position of the MSC after a load flow.

Return type
int

1.29 IscGroup

class *ipsa.***IscGroup**

The IscGroup class provides access to an IPSA group to set and get group members. Note the extension functions will only work for general groups and may not function for other groups e.g., areas, transformer groups.

GetUID() → **int**

Returns the UID of the group.

Returns
The group UID.

Return type
int

GetName() → **str**

Returns the user defined group name as a string.

Returns
The user defined group name.

Return type
str

SetName(*strName: **str***) → **None**

Sets the name as a string.

Parameters
strName (***str***) – The selected string name.

GetGroupType() → **int**

Returns the type of the group where:

- 0 = No group type
- 1 = Area type group – contains all busbars in an area
- 2 = Mixed item group
- 3 = Load scaling group
- 4 = Load transfer group
- 5 = Protection device group
- 8 = Generator scaling group

- 9 = Region group
- 10 = Transformer group (master slave operation)

Returns
The group type.

Return type
int

GetMembers() → **List[int]**

Returns a list containing the UIDs of the components in the group.

Returns
The UIDs of the components in the group.

Return type
list(int)

SetMembers*(nUIDs: List[int])* → **None**

Overwrites the current list of group members with the given list of component UIDs. This replaces any existing members with the supplied list of UIDs.

Parameters
nUIDs (*list(int)*) – List of component integers.

ClearMembers() → **None**

Sets the group members to an empty list. This clears any existing members.

AddMember*(nUID: int)* → **None**

Appends the component with the given UID to the list of component UIDs if the UID is not present. All existing group member UIDs are unaffected.

Parameters
nUID (*int*) – Component UID.

RemoveMember*(nUID: int)* → **None**

Removes the component with the given UID from the list of component UIDs if the UID is present. All other existing group member UIDs are unaffected.

Parameters
nUID (*int*) – Component UID.

IsMember*(nUID: int)* → **bool**

Checks whether the component with the given UID is present in the list of component UIDs. The list of group member UIDs will be unaffected.

Parameters
nUID (*int*) – Component UID.

Returns
True if nUID is present in list of member UIDs.

Return type
bool

CompareGroups(*nGroupUID*: ***int***, *bUseIntersection*: ***bool*** = *False*) → **List[int]**

Compares the current group with the group with UID given by nGroupUID. By default, will perform a difference operation returning a list of component UIDs present in the current group but not present in the group with UID given by nGroupUID. If bUseIntersection is True it will return a list of component UIDs present in both lists. Both lists of group member UIDs will be unaffected.

Parameters

- **nGroupUID** (***int***) – UID of the group to compare with.
- **bUseIntersection** (***bool***) – If True performs an intersection, if False a difference operation.

Returns
The list of UIDs that make up the difference (default) or intersection of the two groups.

Return type
list(**int**)

MergeGroups(*nGroupUID*: ***int***, *bDeleteGroup*: ***bool*** = *False*) → **None**

Appends the list of component UIDs from the group with the given UID onto the current group's UID list. By default the group with the given UID will be unaffected, unless bDeleteGroup is True, in which case it will be deleted.

Parameters

- **nGroupUID** (***int***) – UID of the group to merge with.
- **bDeleteGroup** (***bool***) – If True deletes the group with nGroupUID, otherwise the group is unaffected.

GetLoadScalingReal() → **float**

Returns the per unit scaling factor for the active power load.

Returns
The per unit scaling factor for the active power load.

Return type
float

GetLoadScalingReactive() → **float**

Returns the per unit scaling factor for the reactive power load.

Returns

The per unit scaling factor for the reactive power load.

Return type

float

SetLoadScaling(fMW: float, fMVAr: float) → **bool**

Sets the per unit scaling factors for the active and reactive parts of the load.

Parameters

- **fMW** (*float*) – The active part of the load.
- **fMVAr** (*float*) – The reactive part of the load.

Returns

True if successful.

Return type

bool

AddDataExtension(strName: str, default: int | float | str) → **int**

Adds an integer data field and returns the new field index. Sets the default value.

Note: The variable of the function is not called default.

You can use either nDefault, dDefault, or strDefault to specify the default value depending on the type of dta extension being added.

Parameters

- **strName** (*str*) – The name of the field.
- **nDefault** (*int*) – The integer default value.
- **dDefault** (*float*) – The float default value.
- **strDefault** (*str*) – The string default value.

Returns

The new field index.

Return type

int

AddListIntDataExtension(strName: str) → **int**

Adds a data field for a list of integers and returns the new field index. Sets the default value to an empty list.

Parameters

strName (*str*) – The name of the field.

Returns

The new field index.

Return type

int

AddListDblDataExtension(*strName: **str***) → **int**

Adds a data field for a list of doubles and returns the new field index. Sets the default value to an empty list.

Parameters

strName (***str***) – The name of the field.

Returns

The new field index.

Return type

int

AddListStrDataExtension(*strName: **str***) → **int**

Adds a data field for a list of strings and returns the new field index. Sets the default value to an empty list.

Parameters

strName (***str***) – The name of the field.

Returns

The new field index.

Return type

int

GetListIntExtensionValue(*nFieldIndex: **int**, nIndex: **int***) → **int**

Get a single integer value from the list within the given enumerated field.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nIndex** (***int***) – The index of the selected element.

Returns

The element value.

Return type

int

GetListDblExtensionValue(*nFieldIndex: **int**, nIndex: **int***) → **float**

Get a single float value from the list within the given enumerated field.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nIndex** (***int***) – The index of the selected element.

Returns
The element value.

Return type
float

GetListStrExtensionValue(*nFieldIndex: **int**, nIndex: **int***) → **str**

Get a single string value from the list within the given enumerated field.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nIndex** (***int***) – The index of the selected element.

Returns
The element value.

Return type
str

GetListIntSize(*nFieldIndex: **int***) → **int**

Gets the size of the list of integers for the given enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The size of the field list.

Return type
int

GetListDblSize(*nFieldIndex: **int***) → **int**

Gets the size of the list of doubles for the given enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The size of the field list.

Return type
int

GetListStrSize(*nFieldIndex: **int***) → **int**

Gets the size of the list of strings for the given enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The size of the field list.

Return type
int

SetListIntExtensionValue(*nFieldIndex*: **int**, *nIndex*: **int**, *nValue*: **int**) → **bool**

Sets the value of a specified element in a list of integers within the given enumerated field.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nIndex** (*int*) – The index of the selected element.
- **nValue** (*int*) – The selected value.

Returns
True if the operation was successful.

Return type
bool

SetListDblExtensionValue(*nFieldIndex*: **int**, *nIndex*: **int**, *dValue*: **float**) → **bool**

Sets the value of a specified element in a list of doubles within the given enumerated field.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nIndex** (*int*) – The index of the selected element.
- **dValue** (*float*) – The selected value.

Returns
True if the operation was successful.

Return type
bool

SetListStrExtensionValue(*nFieldIndex*: **int**, *nIndex*: **int**, *strValue*: **str**) → **bool**

Sets the value of a specific element in a list of strings within the given enumerated field.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nIndex** (*int*) – The index of the selected element.
- **strValue** (*str*) – The selected value.

Returns
True if the operation was successful.

Return type
bool

PushBackListIntExtensionValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Adds an item with the given value to the end of a list of integers within the given enumerated field.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The selected value.

Returns
True if the operation was successful.

Return type
bool

PushBackListDblExtensionValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Adds an item with the given value to the end of a list of doubles within the given enumerated field.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The selected value.

Returns
True if the operation was successful.

Return type
bool

PushBackListStrExtensionValue(*nFieldIndex: **int**, strValue: **str***) → **bool**

Adds an item with the given value to the end of a list of strings within the given enumerated field.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The selected value.

Returns
True if the operation was successful.

Return type
bool

GetExtensionFieldIndex(*strName: **str***) → **int**

Returns the field index for the extended data field of a specified name.

Parameters
strName (*str*) – The name of the extended data field.

Returns
The field index.

Return type
int

GetExtensionNames() → **Dict**[**int**, **str**]

Returns a dictionary of extension field indexes and field names. The dictionary keys are integers representing all the extended data fields. The dictionary values are the field names of the individual extended data fields. Each extended data field is therefore represented by {nIndex:strName}, where integer nIndex is the field index and string strName is the field name.

Returns
Dictionary of extension field indexes and field names.

Return type
dict(**int**, **str**)

1.30 IscPlugin

The *IscPlugin* class provides access to an IPSA plugin, to set and get data values and assign the plugin to a component. To use the functions in this section an *IscPlugin* plugin object must be created from the *CreatePlugin* function of the *IscNetwork* class. One such object should be created each time a plugin is to be assigned to a network component. The sequence of operations is as follows:

1. Create an *IscPlugin* from the *CreatePlugin* function of *IscNetwork*
    a. The plugin name should be obtained from the plugin documentation
2. Set the *ControlledUID* field value to the UID of the component that the plugin is to be assigned to
3. Set the *Plugin* field value of the component itself to the UID of the plugin created in step 1
4. The plugin parameters can now be set using the normal *SetIntParameterValue* function calls etc

a. Note that the *Set…*/*Get…* functions are used only to get and set *IscPlugin* **field values** such as *Name* and *Type*

Refer to the documentation provided with each plugin to determine the usage and parameter values available.

1.30.1 Field Values

Table 27: **IscPlugin Field Values**

Type	Field Name	Description
Integer	ControlledUID	Gets the unique ID for controlled plugin.
String	Name	Gets the plugin name.
Integer	Type	Returns the type of the plugin, defined as follows: • 1 = Synchronous Machine AVR • 2 = Synchronous Machine Governor • 3 = DC Machine AVR • 4 = DC Machine Governor • 5 = Induction Machine D Axis AVR • 6 = Induction Machine Q Axis AVR • 7 = Induction Machine Governor • 8 = Synchronous Machine • 9 = DC Machine • 10 = AC/DC Converter • 11 = AC Converter Controller • 12 = DC Converter Controller • 13 = DC Non – Linear Devive • 14 = Universal Machine Active Power Controller • 15 = Universal Machine Reactive Power Controller • 16 = Induction Machine • 17 = Universal Machine • 18 = MSC Controller • 19 = SVC Controller • 20 = Transformer AVR • 21 = Network Controller • 30 = Line Dynamic Rating • 31 = Transformer Dynamic Rating • 32 = Transformer Reverse Rating • 50 = Battery Dynamic Model
String	Model	Returns the model name of the plugin.

1.30.2 IscPlugin Class

class *ipsa.**IscPlugin***

Provides access to an IPSA plugin.

SetName(*strName:* ***str***) → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

GetIValue(*nFieldIndex:* ***int***) → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The integer value.

Return type

int

GetDValue(*nFieldIndex:* ***int***) → **float**

Returns a double value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The double value.

Return type

float

GetSValue(*nFieldIndex:* ***int***) → **str**

Returns a string value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The string value.

Return type

str

SetIValue(*nFieldIndex*: ***int***, *nValue*: ***int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex*: ***int***, *dValue*: ***float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex*: ***int***, *strValue*: ***int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

SetIntParameter(*nPluginIndex*: ***int***, *nValue*: ***int***) → **bool**

Sets the index of the specific plugin parameter for the field from an integer value. The parameters are specific for the plugin object.

Parameters

- **nPluginIndex** (***int***) – The index to the specific plugin parameter.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDoubleParameter(nPluginIndex: int, dValue: float) → bool

Sets the index of the specific plugin parameter for the field from a double value. The parameters are specific for the plugin object.

Parameters

- **nPluginIndex** (*int*) – The index to the specific plugin parameter.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetBoolParameter(nPluginIndex: int, strValue: int) → bool

Sets the index of the specific plugin parameter for the field from a boolean value. The parameters are specific for the plugin object.

Parameters

- **nPluginIndex** (*int*) – The index to the specific plugin parameter.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

GetIntParameter(nPluginIndex: int) → int

Returns an integer parameter for the enumerated field defined by the specific plugin parameter. The parameters are specific for the plugin object.

Parameters
nPluginIndex (*int*) – The index to the specific plugin parameter.

Returns
The integer value.

Return type
int

GetDoubleParameter(*nPluginIndex: **int***) → **float**

Returns a double parameter for the enumerated field defined by the specific plugin parameter. The parameters are specific for the plugin object.

Parameters

nPluginIndex (***int***) – The index to the specific plugin parameter.

Returns

The double value.

Return type

float

GetBoolParameter(*nPluginIndex: **int***) → **bool**

Returns a boolean parameter for the enumerated field defined by the specific plugin parameter. The parameters are specific for the plugin object.

Parameters

nPluginIndex (***int***) – The index to the specific plugin parameter.

Returns

The string value.

Return type

bool

GetIntOutput(*nFieldIndex: **int***) → **int**

Returns the integer output of the plugin itself for the field. The parameters are specific for the plugin object.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The integer value.

Return type

int

GetDoubleOutput(*nFieldIndex: **int***) → **float**

Returns the double output of the plugin itself for the field. The parameters are specific for the plugin object.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The double value.

Return type

float

GetBoolOutput(*nFieldIndex: int*) → **bool**

Returns the boolean output of the plugin itself for the field. The parameters are specific for the plugin object.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The string value.

Return type
bool

1.31 IscVoltageRegulator

The *IscVoltageRegulator* class provides access to a series voltage regulator to get and set data values.

1.31.1 Field Values

Table 28: **IscVoltageRegulator Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	FromBusName	Returns the busbar name at the From end of the branch the regulator is located on.
String	ToBusName	Returns the busbar name at the To end of the branch the regulator is located on.
String	Name	Name of the voltage regulator.
Integer	Status	Status of voltage regulator: • 0 = Switched in • -1 = Switched out
Float	ResistancePU	Gets or sets the resistance of the voltage regulator in per unit.
Float	ReactancePU	Gets or sets the reactance of the voltage regulator in per unit.
Float	TapStart	Present tap position, used as a starting point for the next load flow.
Float	MinTap	Minimum tap position, normally negative or zero.
Float	TapStep	Tap increment. This defaults to 0.01 if left blank.

continues on next page

Table 28 – continued from previous page

Type	Field Name	Description
Float	MaxTap	Maximum tap position, normally positive or zero.
Integer	ControlsUID	Returns the UID of the branch that the voltage regulator is located on.
Integer	ControlMode	Gets or sets the control mode of the voltage regulator as defined by: • 0 = Manual tap control • 1 = Forward locked mode • 2 = Reverse locked mode • 3 = Neutral reverse mode • 4 = Cogeneration mode • 5 = Normal bi-directional mode • 6 = Reactive bi-directional mode
Float	VoltageSetpoint-Forward	Gets or sets the target voltage in per unit when operating in the forward direction.
Float	CompensatingR-Forward	Gets or sets the compensating resistance in per unit when operating in the forward direction.
Float	CompensatingXForward	Gets or sets the compensating reactance in per unit when operating in the forward direction.
Float	VoltageSetpoint-Backward	Gets or sets the target voltage in per unit when operating in the reverse direction.
Float	CompensatingR-Backward	Gets or sets the compensating resistance in per unit when operating in the reverse direction.
Float	Compensat-ingXBackward	Gets or sets the compensating reactance in per unit when operating in the reverse direction.

1.31.2 IscVoltageRegulator Class

class *ipsa.**IscVoltageRegulator***

Provides access to a series voltage regulator.

SetName*(strName: **str**)* → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

GetIValue(*nFieldIndex: **int***) → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The integer value.

Return type
int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex*: ***int***, *dValue*: ***float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex*: ***int***, *strValue*: ***int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex*: ***int***, *bValue*: ***bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

GetBranchUID() → **int**

Returns the UID of the branch that the voltage regulator is located on.

Returns

The branch UID.

Return type

int

1.32 IscUnbalancedLine

The *IscUnbalancedLine* class provides access to the three phase unbalanced lines to get and set data values.

1.32.1 Field Values

Table 29: **IscUnbalancedLine Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique component ID for the "From" busbar.
Integer	ToUID	Gets the unique component ID for the "To" busbar.
String	FromBusName	Gets the sending busbar name.
String	ToBusName	Gets the receiving busbar name.
String	Name	Gets the branch name.
Integer	Type	Gets the branch/line type: • 0 = Unset • 1 = Overhead lines • 2 = Cable • 3 = Ducted • 4 = Mixed
Integer	Status	Line status: • 0 = Switched in. • -1 = Sending/From end switched out • -2 = Receiving/To end switched out • -3 = Both ends switched out
Boolean	HasPhaseA	Gets or sets if the line has the A phase connected. Set to *True* to enable the A phase.

continues on next page

Table 29 – continued from previous page

Type	Field Name	Description
Boolean	HasPhaseB	Gets or sets if the line has the B phase connected. Set to *True* to enable the B phase.
Boolean	HasPhaseC	Gets or sets if the line has the C phase connected. Set to *True* to enable the C phase.
Boolean	HasNeutral	Gets or sets if the line has the neutral conductor connected. Set to *True* to enable the neutral conductor.
Float	ResistancePhasePU	Gets or sets the positive sequence resistance in all phases.
Float	ReactancePhasePU	Gets or sets the positive sequence reactance in all phases.
Float	SusceptancePhasePU	Gets or sets the positive sequence susceptance in all phases.
Float	ResistanceNeutralPU	Gets or sets the neutral conductor resistance in all phases.
Float	ReactanceNeutralPU	Gets or sets the neutral conductor reactance in all phases.
Float	SusceptanceNeutralPU	Gets or sets the neutral conductor susceptance in all phases.
String	DbType	Gets the branch database type.
Float	DbLength	Gets the branch database length.
Integer	DbPar	Gets the branch database number in parallel.

1.32.2 IscUnbalancedLine Class

class** ipsa.**IscUnbalancedLine

Provides access to the three-phase unbalanced lines.

***SetName**(strName: **str**)* → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

***GetIValue**(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The integer value.

Return type
int

GetDValue(nFieldIndex: int) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The double value.

Return type
float

GetSValue(nFieldIndex: int) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The string value.

Return type
str

GetBValue(nFieldIndex: int) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (*int*) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(nFieldIndex: int, nValue: int) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **nValue** (*int*) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: int, dValue: float*) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **dValue** (*float*) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: int, strValue: int*) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **strValue** (*str*) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: int, bValue: bool*) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (*int*) – The field index.
- **bValue** (*bool*) – The given boolean value.

Returns
True if successful.

Return type
bool

AddSections(*nSections: int*) → **None**

Add sections to the unbalanced line. All unbalanced lines start with one section.

Parameters

nSections (*int*) – The number of sections.

GetSections() → **int**

Returns the number of sections in the unbalanced line. All unbalanced lines have at least one section.

Returns

The number of sections in the unbalanced line.

Return type

int

GetRatingMVA(nRatingIndex: int) → **float**

Returns the MVA rating associated with the rating set. The same rating is used for all phases.

Parameters

nRatingIndex (*int*) – Specifies which rating set the data is applied to.

Returns

The MVA rating for the transformer.

Return type

float

SetRatingkA(nRatingIndex: int, dRatingkA: float) → **None**

Sets the kA rating to given value for the rating set given by the rating index. The same rating is used for all phases.

Parameters

- **nRatingIndex** (*int*) – The rating index.
- **dRatingkA** (*float*) – The kA rating value.

SetRatingMVA(nRatingIndex: int, dRatingMVA: float) → **None**

Sets the MVA rating to given value for the rating set given by the rating index. The same rating is used for all phases.

Parameters

- **nRatingIndex** (*int*) – The rating index.
- **dRatingMVA** (*float*) – The MVA rating value.

GetRatingSendkA(nRatingIndex: int) → **float**

Returns the sending end kA rating associated with the rating set given by the rating index. The same rating is used for all phases.

Parameters

nRatingIndex (*int*) – The rating index.

Returns
The sending end kA rating.

Return type
float

GetRatingReceivekA(nRatingIndex: int) → **float**

Returns the receiving end kA rating associated with the rating set given by the rating index. The same rating is used for all phases.

Parameters
nRatingIndex (*int*) – The rating index.

Returns
The receiving end kA rating.

Return type
float

GetRealPowerSendAMW() → **float**

Returns the branch sending end real power in MW in the A phase.

Returns
The branch sending end real power in MW in the A phase.

Return type
float

GetRealPowerSendBMW() → **float**

Returns the branch sending end real power in MW in the B phase.

Returns
The branch sending end real power in MW in the B phase.

Return type
float

GetRealPowerSendCMW() → **float**

Returns the branch sending end real power in MW in the C phase.

Returns
The branch sending end real power in MW in the C phase.

Return type
float

GetRealPowerSendNMW() → **float**

Returns the branch sending end real power in MW in the N phase.

Returns
The branch sending end real power in MW in the N phase.

Return type
float

GetReactivePowerSendAMVAr() → **float**

Returns the branch sending end reactive power in MVAr in the A phase.

Returns
The branch sending end reactive power in MVAr in the A phase.

Return type
float

GetReactivePowerSendBMVAr() → **float**

Returns the branch sending end reactive power in MVAr in the B phase.

Returns
The branch sending end reactive power in MVAr in the B phase.

Return type
float

GetReactivePowerSendCMVAr() → **float**

Returns the branch sending end reactive power in MVAr in the C phase.

Returns
The branch sending end reactive power in MVAr in the C phase.

Return type
float

GetReactivePowerSendNMVAr() → **float**

Returns the branch sending end reactive power in MVAr in the N phase.

Returns
The branch sending end reactive power in MVAr in the N phase.

Return type
float

GetSendPowerAMVA() → **float**

Returns the branch sending end power in MVA in the A phase.

Returns
The branch sending end power in MVA in the A phase.

Return type
float

GetSendPowerBMVA() → **float**

Returns the branch sending end power in MVA in the B phase.

Returns
The branch sending end power in MVA in the B phase.

Return type
float

GetSendPowerCMVA() → **float**

Returns the branch sending end power in MVA in the C phase.

Returns
The branch sending end power in MVA in the C phase.

Return type
float

GetSendPowerNMVA() → **float**

Returns the branch sending end power in MVA in the N phase.

Returns
The branch sending end power in MVA in the N phase.

Return type
float

GetRealPowerSendAkW() → **float**

Returns the branch sending end real power in kW in the A phase.

Returns
The branch sending end real power in kW in the A phase.

Return type
float

GetRealPowerSendBkW() → **float**

Returns the branch sending end real power in kW in the B phase.

Returns
The branch sending end real power in kW in the B phase.

Return type
float

GetRealPowerSendCkW() → **float**

Returns the branch sending end real power in kW in the C phase.

Returns
The branch sending end real power in kW in the C phase.

Return type
float

GetRealPowerSendNkW() → **float**

Returns the branch sending end real power in kW in the N phase.

Returns
The branch sending end real power in kW in the N phase.

Return type
float

GetReactivePowerSendAkVAr() → **float**

Returns the branch sending end reactive power in kVAr in the A phase.

Returns
The branch sending end reactive power in kVAr in the A phase.

Return type
float

GetReactivePowerSendBkVAr() → **float**

Returns the branch sending end reactive power in kVAr in the B phase.

Returns
The branch sending end reactive power in kVAr in the B phase.

Return type
float

GetReactivePowerSendCkVAr() → **float**

Returns the branch sending end reactive power in kVAr in the C phase.

Returns
The branch sending end reactive power in kVAr in the C phase.

Return type
float

GetReactivePowerSendNkVAr() → **float**

Returns the branch sending end reactive power in kVAr in the N phase.

Returns
The branch sending end reactive power in kVAr in the N phase.

Return type
float

GetSendPowerAkVA() → **float**

Returns the branch sending end power in kVA in the A phase.

Returns
The branch sending end power in kVA in the A phase.

Return type
float

GetSendPowerBkVA() → **float**

Returns the branch sending end power in kVA in the B phase.

Returns
The branch sending end power in kVA in the B phase.

Return type
float

GetSendPowerCkVA() → **float**

Returns the branch sending end power in kVA in the C phase.

Returns
The branch sending end power in kVA in the C phase.

Return type
float

GetSendPowerNkVA() → **float**

Returns the branch sending end power in kVA in the N phase.

Returns
The branch sending end power in kVA in the N phase.

Return type
float

GetRealPowerRecvAMW() → **float**

Returns the branch receive end real power in MW in the A phase.

Returns
The branch receive end real power in MW in the A phase.

Return type
float

GetRealPowerRecvBMW() → **float**

Returns the branch receive end real power in MW in the B phase.

Returns
The branch receive end real power in MW in the B phase.

Return type
float

GetRealPowerRecvCMW() → **float**

Returns the branch receive end real power in MW in the C phase.

Returns
The branch receive end real power in MW in the C phase.

Return type
float

GetRealPowerRecvNMW() → float

Returns the branch receive end real power in MW in the N phase.

Returns
The branch receive end real power in MW in the N phase.

Return type
float

GetReactivePowerRecvAMVAr() → float

Returns the branch receive end reactive power in MVAr in the A phase.

Returns
The branch receive end reactive power in MVAr in the A phase.

Return type
float

GetReactivePowerRecvBMVAr() → float

Returns the branch receive end reactive power in MVAr in the B phase.

Returns
The branch receive end reactive power in MVAr in the B phase.

Return type
float

GetReactivePowerRecvCMVAr() → float

Returns the branch receive end reactive power in MVAr in the C phase.

Returns
The branch receive end reactive power in MVAr in the C phase.

Return type
float

GetReactivePowerRecvNMVAr() → float

Returns the branch receive end reactive power in MVAr in the N phase.

Returns
The branch receive end reactive power in MVAr in the N phase.

Return type
float

GetRecvPowerAMVA() → **float**

Returns the branch receive end power in MVA in the A phase.

Returns
The branch receive end power in MVA in the A phase.

Return type
float

GetRecvPowerBMVA() → **float**

Returns the branch receive end power in MVA in the B phase.

Returns
The branch receive end power in MVA in the B phase.

Return type
float

GetRecvPowerCMVA() → **float**

Returns the branch receive end power in MVA in the C phase.

Returns
The branch receive end power in MVA in the C phase.

Return type
float

GetRecvPowerNMVA() → **float**

Returns the branch receive end power in MVA in the N phase.

Returns
The branch receive end power in MVA in the N phase.

Return type
float

GetRealPowerRecvAkW() → **float**

Returns the branch receive end real power in kW in the A phase.

Returns
The branch receive end real power in kW in the A phase.

Return type
float

GetRealPowerRecvBkW() → **float**

Returns the branch receive end real power in kW in the B phase.

Returns
The branch receive end real power in kW in the B phase.

Return type
float

GetRealPowerRecvCkW() → **float**

Returns the branch receive end real power in kW in the C phase.

Returns
The branch receive end real power in kW in the C phase.

Return type
float

GetRealPowerRecvNkW() → **float**

Returns the branch receive end real power in kW in the N phase.

Returns
The branch receive end real power in kW in the N phase.

Return type
float

GetReactivePowerRecvAkVAr() → **float**

Returns the branch receive end reactive power in kVAr in the A phase.

Returns
The branch receive end reactive power in kVAr in the A phase.

Return type
float

GetReactivePowerRecvBkVAr() → **float**

Returns the branch receive end reactive power in kVAr in the B phase.

Returns
The branch receive end reactive power in kVAr in the B phase.

Return type
float

GetReactivePowerRecvCkVAr() → **float**

Returns the branch receive end reactive power in kVAr in the C phase.

Returns
The branch receive end reactive power in kVAr in the C phase.

Return type
float

GetReactivePowerRecvNkVAr() → **float**

Returns the branch receive end reactive power in kVAr in the N phase.

Returns
The branch receive end reactive power in kVAr in the N phase.

Return type
float

GetRecvPowerAkVA() → float

Returns the branch receive end power in kVA in the A phase.

Returns
The branch receive end power in kVA in the A phase.

Return type
float

GetRecvPowerBkVA() → float

Returns the branch receive end power in kVA in the B phase.

Returns
The branch receive end power in kVA in the B phase.

Return type
float

GetRecvPowerCkVA() → float

Returns the branch receive end power in kVA in the C phase.

Returns
The branch receive end power in kVA in the C phase.

Return type
float

GetRecvPowerNkVA() → float

Returns the branch receive end power in kVA in the N phase.

Returns
The branch receive end power in kVA in the N phase.

Return type
float

GetRealPowerSendMeanMW() → float

Returns the real power mean in MW of the three branch phase send end powers.

Returns
The real power mean in MW of the three branch phase send end powers.

Return type
float

GetReactivePowerSendMeanMVAr() → **float**

Returns the reactive power mean in MVAr of the three branch phase send end powers.

Returns
The real power mean in MVAr of the three branch phase send end powers.

Return type
float

GetSendPowerMeanMVA() → **float**

Returns the power mean in MVA of the three branch phase send end powers.

Returns
The power mean in MVA of the three branch phase send end powers.

Return type
float

GetRealPowerSendMeankW() → **float**

Returns the real power mean in kW of the three branch phase send end powers.

Returns
The real power mean in kW of the three branch phase send end powers.

Return type
float

GetReactivePowerSendMeankVAr() → **float**

Returns the reactive power mean in kVAr of the three branch phase send end powers.

Returns
The reactive power mean in kVAr of the three branch phase send end powers.

Return type
float

GetSendPowerMeankVA() → **float**

Returns the power mean in kVA of the three branch phase send end powers.

Returns
The power mean in kVA of the three branch phase send end powers.

Return type
float

GetRealPowerSendMaxMW() → **float**

Returns the highest real power of the three branch phase send end powers in MW.

Returns
The highest real power of the three branch phase send end powers in MW.

Return type
float

GetReactivePowerSendMaxMVAr() → **float**

Returns the highest reactive power of the three branch phase send end powers in MVAr.

Returns
The highest reactive power of the three branch phase send end powers in MVAr.

Return type
float

GetSendPowerMaxMVA() → **float**

Returns the highest power of the three branch phase send end powers in MVA.

Returns
The highest power of the three branch phase send end powers in MVA.

Return type
float

GetRealPowerSendMaxkW() → **float**

Returns the highest real power of the three branch phase send end powers in kW.

Returns
The highest real power of the three branch phase send end powers in kW.

Return type
float

GetReactivePowerSendMaxkVAr() → **float**

Returns the highest reactive power of the three branch phase send end powers in kVAr.

Returns
The highest reactive power of the three branch phase send end powers in kVAr.

Return type
float

GetSendPowerMaxkVA() → **float**

Returns the highest power of the three branch phase send end powers in kVA.

Returns
The highest power of the three branch phase send end powers in kVA.

Return type
float

GetRealPowerRecvMeanMW() → **float**

Returns the mean of the three branch phase receive end real powers in MW.

Returns
The mean of the three branch phase receive end real powers in MW.

Return type
float

GetReactivePowerRecvMeanMVAr() → **float**

Returns the mean of the three branch phase receive end reactive powers in MVAr.

Returns
The mean of the three branch phase receive end reactive powers in MVAr.

Return type
float

GetRecvPowerMeanMVA() → **float**

Returns the mean of the three branch phase receive end powers in MVA.

Returns
The mean of the three branch phase receive end powers in MVA.

Return type
float

GetRealPowerRecvMeankW() → **float**

Returns the mean of the three branch phase receive end real powers in kW.

Returns
The mean of the three branch phase receive end real powers in kW.

Return type
float

GetReactivePowerRecvMeankVAr() → **float**

Returns the mean of the three branch phase receive end reactive powers in kVAr.

Returns

The mean of the three branch phase receive end reactive powers in kVAr.

Return type

float

GetRecvPowerMeankVA() → **float**

Returns the mean of the three branch phase receive end powers in kVA.

Returns

The mean of the three branch phase receive end powers in kVA.

Return type

float

GetRealPowerRecvMaxMW() → **float**

Returns the highest of the three branch phase receive end real powers in MW.

Returns

The highest of the three branch phase receive end real powers in MW.

Return type

float

GetReactivePowerRecvMaxMVAr() → **float**

Returns the highest of the three branch phase receive end reactive powers in MVAr.

Returns

The highest of the three branch phase receive end reactive powers in MVAr.

Return type

float

GetRecvPowerMaxMVA() → **float**

Returns the highest of the three branch phase receive end powers in MVA.

Returns

The highest of the three branch phase receive end powers in MVA.

Return type

float

GetRealPowerRecvMaxkW() → **float**

Returns the highest of the three branch phase receive end real powers in kW.

Returns

The highest of the three branch phase receive end real powers in kW.

Return type
float

GetReactivePowerRecvMaxkVAr() → **float**

Returns the highest of the three branch phase receive end reactive powers in kVAr.

Returns
The highest of the three branch phase receive end reactive powers in kVAr.

Return type
float

GetRecvPowerMaxkVA() → **float**

Returns the highest of the three branch phase receive end powers in kVA.

Returns
The highest of the three branch phase receive end powers in kVA.

Return type
float

GetRealPowerSendPosMW() → **float**

Returns the positive branch phase sequence send end real power in MW.

Returns
The positive branch phase sequence send end real power in MW.

Return type
float

GetRealPowerSendNegMW() → **float**

Returns the negative branch phase sequence send end real power in MW.

Returns
The negative branch phase sequence send end real power in MW.

Return type
float

GetRealPowerSendZeroMW() → **float**

Returns the zero branch phase sequence send end real power in MW.

Returns
The zero branch phase sequence send end real power in MW.

Return type
float

GetReactivePowerSendPosMVAr() → **float**

Returns the positive branch phase sequence send end reactive power in MVAr.

Returns
The positive branch phase sequence send end reactive power in MVAr.

Return type
float

GetReactivePowerSendNegMVAr() → **float**

Returns the negative branch phase sequence send end reactive power in MVAr.

Returns
The negative branch phase sequence send end reactive power in MVAr.

Return type
float

GetReactivePowerSendZeroMVAr() → **float**

Returns the zero branch phase sequence send end reactive power in MVAr.

Returns
The zero branch phase sequence send end reactive power in MVAr.

Return type
float

GetSendPowerPosMVA() → **float**

Returns the positive branch phase sequence send end power in MVA.

Returns
The positive branch phase sequence send end power in MVA.

Return type
float

GetSendPowerNegMVA() → **float**

Returns the negative branch phase sequence send end power in MVA.

Returns
The negative branch phase sequence send end power in MVA.

Return type
float

GetSendPowerZeroMVA() → **float**

Returns the zero branch phase sequence send end power in MVA.

Returns
The zero branch phase sequence send end power in MVA.

Return type
float

GetRealPowerSendPoskW() → **float**

Returns the positive branch phase sequence send end real power in kW.

Returns
The positive branch phase sequence send end real power in kW.

Return type
float

GetRealPowerSendNegkW() → **float**

Returns the negative branch phase sequence send end real power in kW.

Returns
The negative branch phase sequence send end real power in kW.

Return type
float

GetRealPowerSendZerokW() → **float**

Returns the zero branch phase sequence send end real power in kW.

Returns
The zero branch phase sequence send end real power in kW.

Return type
float

GetReactivePowerSendPoskVAr() → **float**

Returns the positive branch phase sequence send end reactive power in kVAr.

Returns
The positive branch phase sequence send end reactive power in kVAr.

Return type
float

GetReactivePowerSendNegkVAr() → **float**

Returns the negative branch phase sequence send end reactive power in kVAr.

Returns
The negative branch phase sequence send end reactive power in kVAr.

Return type
float

GetReactivePowerSendZerokVAr() → **float**

Returns the zero branch phase sequence send end reactive power in kVAr.

Returns
The zero branch phase sequence send end reactive power in kVAr.

Return type
float

GetSendPowerPoskVA() → **float**

Returns the positive branch phase sequence send end power in kVA.

Returns
The positive branch phase sequence send end power in kVA.

Return type
float

GetSendPowerNegkVA() → **float**

Returns the negative branch phase sequence send end power in kVA.

Returns
The negative branch phase sequence send end power in kVA.

Return type
float

GetSendPowerZerokVA() → **float**

Returns the zero branch phase sequence send end power in kVA.

Returns
The zero branch phase sequence send end power in kVA.

Return type
float

GetRealPowerRecvPosMW() → **float**

Returns the positive branch phase sequence receive end real power in MW.

Returns
The positive branch phase sequence receive end real power in MW.

Return type
float

GetRealPowerRecvNegMW() → **float**

Returns the negative branch phase sequence receive end real power in MW.

Returns
The negative branch phase sequence receive end real power in MW.

Return type
float

GetRealPowerRecvZeroMW() → **float**

Returns the zero branch phase sequence receive end real power in MW.

Returns
The zero branch phase sequence receive end real power in MW.

Return type
float

GetReactivePowerRecvPosMVAr() → float

Returns the positive branch phase sequence receive end reactive power in MVAr.

Returns
The positive branch phase sequence receive end reactive power in MVAr.

Return type
float

GetReactivePowerRecvNegMVAr() → float

Returns the negative branch phase sequence receive end reactive power in MVAr.

Returns
The negative branch phase sequence receive end reactive power in MVAr.

Return type
float

GetReactivePowerRecvZeroMVAr() → float

Returns the zero branch phase sequence receive end reactive power in MVAr.

Returns
The zero branch phase sequence receive end reactive power in MVAr.

Return type
float

GetRecvPowerPosMVA() → float

Returns the positive branch phase sequence receive end power in MVA.

Returns
The positive branch phase sequence receive end power in MVA.

Return type
float

GetRecvPowerNegMVA() → float

Returns the negative branch phase sequence receive end power in MVA.

Returns
The negative branch phase sequence receive end power in MVA.

Return type
float

GetRecvPowerZeroMVA() → float

Returns the zero branch phase sequence receive end power in MVA.

Returns
The zero branch phase sequence receive end power in MVA.

Return type
float

GetRealPowerRecvPoskW() → float

Returns the positive branch phase sequence receive end real power in kW.

Returns
The positive branch phase sequence receive end real power in kW.

Return type
float

GetRealPowerRecvNegkW() → float

Returns the negative branch phase sequence receive end real power in kW.

Returns
The negative branch phase sequence receive end real power in kW.

Return type
float

GetRealPowerRecvZerokW() → float

Returns the zero branch phase sequence receive end real power in kW.

Returns
The zero branch phase sequence receive end real power in kW.

Return type
float

GetReactivePowerRecvPoskVAr() → float

Returns the positive branch phase sequence receive end reactive power in kVAr.

Returns
The positive branch phase sequence receive end reactive power in kVAr.

Return type
float

GetReactivePowerRecvNegkVAr() → float

Returns the negative branch phase sequence receive end reactive power in kVAr.

Returns

The negative branch phase sequence receive end reactive power in kVAr.

Return type

float

GetReactivePowerRecvZerokVAr() → **float**

Returns the zero branch phase sequence receive end reactive power in kVAr.

Returns

The zero branch phase sequence receive end reactive power in kVAr.

Return type

float

GetRecvPowerPoskVA() → **float**

Returns the positive branch phase sequence receive end power in kVA.

Returns

The positive branch phase sequence receive end power in kVA.

Return type

float

GetRecvPowerNegkVA() → **float**

Returns the negative branch phase sequence receive end power in kVA.

Returns

The negative branch phase sequence receive end power in kVA.

Return type

float

GetRecvPowerZerokVA() → **float**

Returns the zero branch phase sequence receive end power in kVA.

Returns

The zero branch phase sequence receive end power in kVA.

Return type

float

1.33 IscUnbalancedLoad

The *IscUnbalancedLoad* class provides access to the three phase unbalanced load components to get and set data values.

1.33.1 Field Values

Table 30: **IscUnbalancedLoad Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID for busbar.
String	BusName	Gets the busbar name.
String	Name	Gets the branch name.
Integer	Status	Line status: • 0 = Switched in • -1 = Switched out
Integer	Connection	Connection type: • 1 = Phase-ground • 2 = Phase-neutral • 3 = Phase-phase
Boolean	HasPhaseA	Gets or sets if the line has the A phase connected. Set to *True* to enable the A phase.
Boolean	HasPhaseB	Gets or sets if the line has the B phase connected. Set to *True* to enable the B phase.
Boolean	HasPhaseC	Gets or sets if the line has the C phase connected. Set to *True* to enable the C phase.
Float	RealPhaseAMW	Gets or sets the A phase power in MW.
Float	ReactivePhaseAM-VAr	Gets or sets the A phase power in MVAr.
Float	RealPhaseBMW	Gets or sets the B phase power in MW.
Float	ReactivePhaseBM-VAr	Gets or sets the B phase power in MVAr.
Float	RealPhaseCMW	Gets or sets the C phase power in MW.
Float	ReactivePhaseCM-VAr	Gets or sets the C phase power in MVAr.
Integer	ProfilePhaseAUID	Gets or sets the load profile UID applied to the A phase of this load.

continues on next page

Table 30 – continued from previous page

Type	Field Name	Description
Float	ProfilePhaseBUID	Gets or sets the load profile UID applied to the B phase of this load.
Integer	ProfilePhaseCUID	Gets or sets the load profile UID applied to the C phase of this load.

1.33.2 IscUnbalancedLoad Class

class *ipsa.**IscUnbalancedLoad***

Provides access to the three phase unbalanced load components.

SetName*(strName: **str**)* → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

GetIValue*(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The integer value.

Return type

int

GetDValue*(nFieldIndex: **int**)* → **float**

Returns a double value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The double value.

Return type

float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The string value.

Return type

str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The boolean value.

Return type

bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns

True if successful.

Return type

bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns

True if successful.

Return type

bool

SetSValue(*nFieldIndex: **int***, *strValue: **int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: **int***, *bValue: **bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns
True if successful.

Return type
bool

GetTotalMeanMVA() → **float**

Returns the mean load power across all 3 phases in MVA.

Returns
The mean load power across all 3 phases in MVA.

Return type
float

GetTotalMeankVA() → **float**

Returns the mean load power across all 3 phases in kVA.

Returns
The mean load power across all 3 phases in kVA.

Return type
float

GetRealMeanMW() → **float**

Returns the mean load power across all 3 phases in MW.

Returns
The mean load power across all 3 phases in MW.

Return type
float

GetRealMeankW() → **float**

Returns the mean load power across all 3 phases in kW.

Returns
The mean load power across all 3 phases in kW.

Return type
float

GetReactiveMeanMVAr() → **float**

Returns the mean load power across all 3 phases in MVAr.

Returns
The mean load power across all 3 phases in MVAr.

Return type
float

GetReactiveMeankVAr() → **float**

Returns the mean load power across all 3 phases in kVAr.

Returns
The mean load power across all 3 phases in kVAr.

Return type
float

GetTotalMaxMVA() → **float**

Returns the highest load power across all 3 phases in MVA.

Returns
The highest load power across all 3 phases in MVA.

Return type
float

GetTotalMaxkVA() → **float**

Returns the highest load power across all 3 phases in kVA.

Returns
The highest load power across all 3 phases in kVA.

Return type
float

GetRealMaxMW() → **float**

Returns the highest load power across all 3 phases in MW.

Returns
The highest load power across all 3 phases in MW.

Return type
float

GetRealMaxkW() → **float**

Returns the highest load power across all 3 phases in kW.

Returns
The highest load power across all 3 phases in kW.

Return type
float

GetReactiveMaxMVAr() → **float**

Returns the highest load power across all 3 phases in MVAr.

Returns
The highest load power across all 3 phases in MVAr.

Return type
float

GetReactiveMaxkVAr() → **float**

Returns the highest load power across all 3 phases in kVAr.

Returns
The highest load power across all 3 phases in kVAr.

Return type
float

GetRealPowerAMW() → **float**

Returns the A phase power for the load in MW.

Returns
The A phase power for the load in MW.

Return type
float

GetRealPowerBMW() → **float**

Returns the B phase power for the load in MW.

Returns
The B phase power for the load in MW.

Return type
float

GetRealPowerCMW() → **float**

Returns the C phase power for the load in MW.

Returns
The C phase power for the load in MW.

Return type
float

GetRealPowerAkW() → **float**

Returns the A phase power for the load in kW.

Returns
The A phase power for the load in kW.

Return type
float

GetRealPowerBkW() → **float**

Returns the B phase power for the load in kW.

Returns
The B phase power for the load in kW.

Return type
float

GetRealPowerCkW() → **float**

Returns the C phase power for the load in kW.

Returns
The C phase power for the load in kW.

Return type
float

GetReactivePowerAMVAr() → **float**

Returns the A phase power for the load in MVAr.

Returns
The A phase power for the load in MVAr.

Return type
float

GetReactivePowerBMVAr() → **float**

Returns the B phase power for the load in MVAr.

Returns
The B phase power for the load in MVAr.

Return type
float

GetReactivePowerCMVAr() → **float**

Returns the C phase power for the load in MVAr.

Returns
The C phase power for the load in MVAr.

Return type
float

GetReactivePowerAkVAr() → **float**

Returns the A phase power for the load in kVAr.

Returns
The A phase power for the load in kVAr.

Return type
float

GetReactivePowerBkVAr() → **float**

Returns the B phase power for the load in kVAr.

Returns
The B phase power for the load in kVAr.

Return type
float

GetReactivePowerCkVAr() → **float**

Returns the C phase power for the load in kVAr.

Returns
The C phase power for the load in kVAr.

Return type
float

1.34 IscUnbalancedTransformer

The *IscUnbalancedTransformer* class provides access to the three phase unbalanced transformer to get and set data values.

1.34.1 Field Values

Table 31: **IscUnbalancedTransformer Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique component ID for the "From" busbar.
Integer	ToUID	Gets the unique component ID for the "To" busbar.
String	FromBusName	Gets the sending busbar name.
String	ToBusName	Gets the receiving busbar name.
String	Name	Gets the transformer name.
Integer	Type	Specifies the transformer type: • 1 = Not set • 2 = Ground Mounted • 3 = Pole Mounted • 7 = Secondary Distribution
Integer	Winding/Vector-Group	Transformer winding type connection as follows: • 1 = XX • 2 = YY • 3 = DD • 4 = XD • 5 = YD where: • X = Earthed star • Y = Unearthed star • D = Delta
Integer	Status	Line status: • 0 = Switched in. • -1 = Sending/From end switched out • -2 = Receiving/To end switched out • -3 = Both ends switched out
Float	ResistancePhasePU	Gets or sets the positive sequence resistance in all phases.
Float	ReactancePhasePU	Gets or sets the positive sequence reactance in all phases.
Float	EarthPrimaryResistancePU	Gets or sets the primary winding earth resistance in all phases.

continues on next page

Table 31 – continued from previous page

Type	Field Name	Description
Float	EarthPrimaryReactancePU	Gets or sets the primary winding earth reactance in all phases.
Float	EarthSecondaryResistancePU	Gets or sets the secondary winding earth resistance in all phases.
Float	EarthSecondaryReactancePU	Gets or sets the secondary winding earth reactance in all phases.
Float	TapPrimaryNominalPC	Nominal tap position on the primary winding, optionally used in a flat start.
Float	TapPrimaryPositionPC	Present tap position on the primary winding, used as a starting point for the next load flow.
Float	MinTapPrimaryPC	Minimum tap position on the primary winding, normally negative or zero.
Float	TapPrimaryStepPC	Tap step or increment on the primary winding. This defaults to 0.01 if left blank.
Float	MaxTapPrimaryPC	Maximum tap position on the primary winding, normally positive or zero.
Float	TapSecondaryNominalPC	Nominal tap position on the secondary winding, optionally used in a flat start.
Float	TapSecondaryPositionPC	Present tap position on the secondary winding, used as a starting point for the next load flow.
Float	MinTapSecondaryPC	Minimum tap position on the secondary winding, normally negative or zero.
Float	TapSecondaryStepPC	Tap step or increment on the secondary winding. This defaults to 0.01 if left blank.
Float	MaxTapSecondaryPC	Maximum tap position on the secondary winding, normally positive or zero.
String	DbType	Gets the branch database type.
Integer	DbPar	Gets the branch database number in parallel.

1.34.2 IscUnbalancedTransformer Class

class** ipsa.**IscUnbalancedTransformer

Provides access to the three phase unbalanced transformer.

***SetName**(strName: **str**)* → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

GetIValue(*nFieldIndex: **int***) → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The integer value.

Return type

int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The double value.

Return type

float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The string value.

Return type

str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The boolean value.

Return type

bool

*SetIValue(nFieldIndex: **int**, nValue: **int**)* → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

*SetDValue(nFieldIndex: **int**, dValue: **float**)* → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns
True if successful.

Return type
bool

*SetSValue(nFieldIndex: **int**, strValue: **int**)* → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

*SetBValue(nFieldIndex: **int**, bValue: **bool**)* → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns
True if successful.

Return type
bool

GetRatingMVA(nRatingIndex: int) → **float**

Returns the MVA rating associated with the rating set. The same rating is used for all phases.

Parameters
nRatingIndex (*int*) – Specifies which rating set the data is applied to.

Returns
The MVA rating for the transformer.

Return type
float

SetRatingkA(nRatingIndex: int, dRatingkA: float) → **None**

Sets the kA rating to given value for the rating set given by the rating index. The same rating is used for all phases.

Parameters

- **nRatingIndex** (*int*) – The rating index.
- **dRatingkA** (*float*) – The kA rating value.

SetRatingMVA(nRatingIndex: int, dRatingMVA: float) → **None**

Sets the MVA rating to given value for the rating set given by the rating index. The same rating is used for all phases.

Parameters

- **nRatingIndex** (*int*) – The rating index.
- **dRatingMVA** (*float*) – The MVA rating value.

GetRatingSendkA(nRatingIndex: int) → **float**

Returns the sending end kA rating associated with the rating set given by the rating index. The same rating is used for all phases.

Parameters
nRatingIndex (*int*) – The rating index.

Returns
The sending end kA rating.

Return type
float

GetRatingReceivekA(nRatingIndex: int) → **float**

Returns the receiving end kA rating associated with the rating set given by the rating index. The same rating is used for all phases.

Parameters
nRatingIndex (*int*) – The rating index.

Returns
The receiving end kA rating.

Return type
float

GetRealPowerSendAMW() → float

Returns the branch sending end real power in MW in the A phase.

Returns
The branch sending end real power in MW in the A phase.

Return type
float

GetRealPowerSendBMW() → float

Returns the branch sending end real power in MW in the B phase.

Returns
The branch sending end real power in MW in the B phase.

Return type
float

GetRealPowerSendCMW() → float

Returns the branch sending end real power in MW in the C phase.

Returns
The branch sending end real power in MW in the C phase.

Return type
float

GetRealPowerSendNMW() → float

Returns the branch sending end real power in MW in the N phase.

Returns
The branch sending end real power in MW in the N phase.

Return type
float

GetReactivePowerSendAMVAr() → float

Returns the branch sending end reactive power in MVAr in the A phase.

Returns
The branch sending end reactive power in MVAr in the A phase.

Return type
float

GetReactivePowerSendBMVAr() → **float**

Returns the branch sending end reactive power in MVAr in the B phase.

Returns
The branch sending end reactive power in MVAr in the B phase.

Return type
float

GetReactivePowerSendCMVAr() → **float**

Returns the branch sending end reactive power in MVAr in the C phase.

Returns
The branch sending end reactive power in MVAr in the C phase.

Return type
float

GetReactivePowerSendNMVAr() → **float**

Returns the branch sending end reactive power in MVAr in the N phase.

Returns
The branch sending end reactive power in MVAr in the N phase.

Return type
float

GetSendPowerAMVA() → **float**

Returns the branch sending end power in MVA in the A phase.

Returns
The branch sending end power in MVA in the A phase.

Return type
float

GetSendPowerBMVA() → **float**

Returns the branch sending end power in MVA in the B phase.

Returns
The branch sending end power in MVA in the B phase.

Return type
float

GetSendPowerCMVA() → **float**

Returns the branch sending end power in MVA in the C phase.

Returns

The branch sending end power in MVA in the C phase.

Return type

float

GetSendPowerNMVA() → float

Returns the branch sending end power in MVA in the N phase.

Returns

The branch sending end power in MVA in the N phase.

Return type

float

GetRealPowerSendAkW() → float

Returns the branch sending end real power in kW in the A phase.

Returns

The branch sending end real power in kW in the A phase.

Return type

float

GetRealPowerSendBkW() → float

Returns the branch sending end real power in kW in the B phase.

Returns

The branch sending end real power in kW in the B phase.

Return type

float

GetRealPowerSendCkW() → float

Returns the branch sending end real power in kW in the C phase.

Returns

The branch sending end real power in kW in the C phase.

Return type

float

GetRealPowerSendNkW() → float

Returns the branch sending end real power in kW in the N phase.

Returns

The branch sending end real power in kW in the N phase.

Return type

float

GetReactivePowerSendAkVAr() → **float**

Returns the branch sending end reactive power in kVAr in the A phase.

Returns
The branch sending end reactive power in kVAr in the A phase.

Return type
float

GetReactivePowerSendBkVAr() → **float**

Returns the branch sending end reactive power in kVAr in the B phase.

Returns
The branch sending end reactive power in kVAr in the B phase.

Return type
float

GetReactivePowerSendCkVAr() → **float**

Returns the branch sending end reactive power in kVAr in the C phase.

Returns
The branch sending end reactive power in kVAr in the C phase.

Return type
float

GetReactivePowerSendNkVAr() → **float**

Returns the branch sending end reactive power in kVAr in the N phase.

Returns
The branch sending end reactive power in kVAr in the N phase.

Return type
float

GetSendPowerAkVA() → **float**

Returns the branch sending end power in kVA in the A phase.

Returns
The branch sending end power in kVA in the A phase.

Return type
float

GetSendPowerBkVA() → **float**

Returns the branch sending end power in kVA in the B phase.

Returns
The branch sending end power in kVA in the B phase.

Return type
float

GetSendPowerCkVA() → **float**

Returns the branch sending end power in kVA in the C phase.

Returns
The branch sending end power in kVA in the C phase.

Return type
float

GetSendPowerNkVA() → **float**

Returns the branch sending end power in kVA in the N phase.

Returns
The branch sending end power in kVA in the N phase.

Return type
float

GetRealPowerRecvAMW() → **float**

Returns the branch receive end real power in MW in the A phase.

Returns
The branch receive end real power in MW in the A phase.

Return type
float

GetRealPowerRecvBMW() → **float**

Returns the branch receive end real power in MW in the B phase.

Returns
The branch receive end real power in MW in the B phase.

Return type
float

GetRealPowerRecvCMW() → **float**

Returns the branch receive end real power in MW in the C phase.

Returns
The branch receive end real power in MW in the C phase.

Return type
float

GetRealPowerRecvNMW() → **float**

Returns the branch receive end real power in MW in the N phase.

Returns
The branch receive end real power in MW in the N phase.

Return type
float

GetReactivePowerRecvAMVAr() → float

Returns the branch receive end reactive power in MVAr in the A phase.

Returns
The branch receive end reactive power in MVAr in the A phase.

Return type
float

GetReactivePowerRecvBMVAr() → float

Returns the branch receive end reactive power in MVAr in the B phase.

Returns
The branch receive end reactive power in MVAr in the B phase.

Return type
float

GetReactivePowerRecvCMVAr() → float

Returns the branch receive end reactive power in MVAr in the C phase.

Returns
The branch receive end reactive power in MVAr in the C phase.

Return type
float

GetReactivePowerRecvNMVAr() → float

Returns the branch receive end reactive power in MVAr in the N phase.

Returns
The branch receive end reactive power in MVAr in the N phase.

Return type
float

GetRecvPowerAMVA() → float

Returns the branch receive end power in MVA in the A phase.

Returns
The branch receive end power in MVA in the A phase.

Return type
float

GetRecvPowerBMVA() → **float**

Returns the branch receive end power in MVA in the B phase.

Returns
The branch receive end power in MVA in the B phase.

Return type
float

GetRecvPowerCMVA() → **float**

Returns the branch receive end power in MVA in the C phase.

Returns
The branch receive end power in MVA in the C phase.

Return type
float

GetRecvPowerNMVA() → **float**

Returns the branch receive end power in MVA in the N phase.

Returns
The branch receive end power in MVA in the N phase.

Return type
float

GetRealPowerRecvAkW() → **float**

Returns the branch receive end real power in kW in the A phase.

Returns
The branch receive end real power in kW in the A phase.

Return type
float

GetRealPowerRecvBkW() → **float**

Returns the branch receive end real power in kW in the B phase.

Returns
The branch receive end real power in kW in the B phase.

Return type
float

GetRealPowerRecvCkW() → **float**

Returns the branch receive end real power in kW in the C phase.

Returns
The branch receive end real power in kW in the C phase.

Return type
float

GetRealPowerRecvNkW() → **float**

Returns the branch receive end real power in kW in the N phase.

Returns
The branch receive end real power in kW in the N phase.

Return type
float

GetReactivePowerRecvAkVAr() → **float**

Returns the branch receive end reactive power in kVAr in the A phase.

Returns
The branch receive end reactive power in kVAr in the A phase.

Return type
float

GetReactivePowerRecvBkVAr() → **float**

Returns the branch receive end reactive power in kVAr in the B phase.

Returns
The branch receive end reactive power in kVAr in the B phase.

Return type
float

GetReactivePowerRecvCkVAr() → **float**

Returns the branch receive end reactive power in kVAr in the C phase.

Returns
The branch receive end reactive power in kVAr in the C phase.

Return type
float

GetReactivePowerRecvNkVAr() → **float**

Returns the branch receive end reactive power in kVAr in the N phase.

Returns
The branch receive end reactive power in kVAr in the N phase.

Return type
float

GetRecvPowerAkVA() → **float**

Returns the branch receive end power in kVA in the A phase.

Returns
The branch receive end power in kVA in the A phase.

Return type
float

GetRecvPowerBkVA() → **float**

Returns the branch receive end power in kVA in the B phase.

Returns
The branch receive end power in kVA in the B phase.

Return type
float

GetRecvPowerCkVA() → **float**

Returns the branch receive end power in kVA in the C phase.

Returns
The branch receive end power in kVA in the C phase.

Return type
float

GetRecvPowerNkVA() → **float**

Returns the branch receive end power in kVA in the N phase.

Returns
The branch receive end power in kVA in the N phase.

Return type
float

GetRealPowerSendMeanMW() → **float**

Returns the real power mean in MW of the three branch phase send end powers.

Returns
The real power mean in MW of the three branch phase send end powers.

Return type
float

GetReactivePowerSendMeanMVAr() → **float**

Returns the reactive power mean in MVAr of the three branch phase send end powers.

Returns
The real power mean in MVAr of the three branch phase send end powers.

Return type
float

GetSendPowerMeanMVA() → **float**

Returns the power mean in MVA of the three branch phase send end powers.

Returns
The power mean in MVA of the three branch phase send end powers.

Return type
float

GetRealPowerSendMeankW() → **float**

Returns the real power mean in kW of the three branch phase send end powers.

Returns
The real power mean in kW of the three branch phase send end powers.

Return type
float

GetReactivePowerSendMeankVAr() → **float**

Returns the reactive power mean in kVAr of the three branch phase send end powers.

Returns
The reactive power mean in kVAr of the three branch phase send end powers.

Return type
float

GetSendPowerMeankVA() → **float**

Returns the power mean in kVA of the three branch phase send end powers.

Returns
The power mean in kVA of the three branch phase send end powers.

Return type
float

GetRealPowerSendMaxMW() → **float**

Returns the highest real power of the three branch phase send end powers in MW.

Returns
The highest real power of the three branch phase send end powers in MW.

Return type
float

GetReactivePowerSendMaxMVAr() → **float**

Returns the highest reactive power of the three branch phase send end powers in MVAr.

Returns
The highest reactive power of the three branch phase send end powers in MVAr.

Return type
float

GetSendPowerMaxMVA() → **float**

Returns the highest power of the three branch phase send end powers in MVA.

Returns
The highest power of the three branch phase send end powers in MVA.

Return type
float

GetRealPowerSendMaxkW() → **float**

Returns the highest real power of the three branch phase send end powers in kW.

Returns
The highest real power of the three branch phase send end powers in kW.

Return type
float

GetReactivePowerSendMaxkVAr() → **float**

Returns the highest reactive power of the three branch phase send end powers in kVAr.

Returns
The highest reactive power of the three branch phase send end powers in kVAr.

Return type
float

GetSendPowerMaxkVA() → **float**

Returns the highest power of the three branch phase send end powers in kVA.

Returns
The highest power of the three branch phase send end powers in kVA.

Return type
float

GetRealPowerRecvMeanMW() → **float**

Returns the mean of the three branch phase receive end real powers in MW.

Returns
The mean of the three branch phase receive end real powers in MW.

Return type
float

GetReactivePowerRecvMeanMVAr() → **float**

Returns the mean of the three branch phase receive end reactive powers in MVAr.

Returns
The mean of the three branch phase receive end reactive powers in MVAr.

Return type
float

GetRecvPowerMeanMVA() → **float**

Returns the mean of the three branch phase receive end powers in MVA.

Returns
The mean of the three branch phase receive end powers in MVA.

Return type
float

GetRealPowerRecvMeankW() → **float**

Returns the mean of the three branch phase receive end real powers in kW.

Returns
The mean of the three branch phase receive end real powers in kW.

Return type
float

GetReactivePowerRecvMeankVAr() → **float**

Returns the mean of the three branch phase receive end reactive powers in kVAr.

Returns
The mean of the three branch phase receive end reactive powers in kVAr.

Return type
float

GetRecvPowerMeankVA() → **float**

Returns the mean of the three branch phase receive end powers in kVA.

Returns
The mean of the three branch phase receive end powers in kVA.

Return type
float

GetRealPowerRecvMaxMW() → **float**

Returns the highest of the three branch phase receive end real powers in MW.

Returns
The highest of the three branch phase receive end real powers in MW.

Return type
float

GetReactivePowerRecvMaxMVAr() → **float**

Returns the highest of the three branch phase receive end reactive powers in MVAr.

Returns
The highest of the three branch phase receive end reactive powers in MVAr.

Return type
float

GetRecvPowerMaxMVA() → **float**

Returns the highest of the three branch phase receive end powers in MVA.

Returns
The highest of the three branch phase receive end powers in MVA.

Return type
float

GetRealPowerRecvMaxkW() → **float**

Returns the highest of the three branch phase receive end real powers in kW.

Returns
The highest of the three branch phase receive end real powers in kW.

Return type
float

GetReactivePowerRecvMaxkVAr() → **float**

Returns the highest of the three branch phase receive end reactive powers in kVAr.

Returns
The highest of the three branch phase receive end reactive powers in kVAr.

Return type
float

GetRecvPowerMaxkVA() → **float**

Returns the highest of the three branch phase receive end powers in kVA.

Returns
The highest of the three branch phase receive end powers in kVA.

Return type
float

GetRealPowerSendPosMW() → **float**

Returns the positive branch phase sequence send end real power in MW.

Returns
The positive branch phase sequence send end real power in MW.

Return type
float

GetRealPowerSendNegMW() → **float**

Returns the negative branch phase sequence send end real power in MW.

Returns
The negative branch phase sequence send end real power in MW.

Return type
float

GetRealPowerSendZeroMW() → **float**

Returns the zero branch phase sequence send end real power in MW.

Returns
The zero branch phase sequence send end real power in MW.

Return type
float

GetReactivePowerSendPosMVAr() → **float**

Returns the positive branch phase sequence send end reactive power in MVAr.

Returns
The positive branch phase sequence send end reactive power in MVAr.

Return type
float

GetReactivePowerSendNegMVAr() → **float**

Returns the negative branch phase sequence send end reactive power in MVAr.

Returns
The negative branch phase sequence send end reactive power in MVAr.

Return type
float

GetReactivePowerSendZeroMVAr() → **float**

Returns the zero branch phase sequence send end reactive power in MVAr.

Returns
The zero branch phase sequence send end reactive power in MVAr.

Return type
float

GetSendPowerPosMVA() → **float**

Returns the positive branch phase sequence send end power in MVA.

Returns
The positive branch phase sequence send end power in MVA.

Return type
float

GetSendPowerNegMVA() → **float**

Returns the negative branch phase sequence send end power in MVA.

Returns
The negative branch phase sequence send end power in MVA.

Return type
float

GetSendPowerZeroMVA() → **float**

Returns the zero branch phase sequence send end power in MVA.

Returns
The zero branch phase sequence send end power in MVA.

Return type
float

GetSendPowerPoskVA() → **float**

Returns the positive branch phase sequence send end power in kVA.

Returns
The positive branch phase sequence send end power in kVA.

Return type
float

GetSendPowerNegkVA() → **float**

Returns the negative branch phase sequence send end power in kVA.

Returns
The negative branch phase sequence send end power in kVA.

Return type
float

GetSendPowerZerokVA() → **float**

Returns the zero branch phase sequence send end power in kVA.

Returns
The zero branch phase sequence send end power in kVA.

Return type
float

GetRealPowerSendPoskW() → **float**

Returns the positive branch phase sequence send end real power in kW.

Returns
The positive branch phase sequence send end real power in kW.

Return type
float

GetRealPowerSendNegkW() → **float**

Returns the negative branch phase sequence send end real power in kW.

Returns
The negative branch phase sequence send end real power in kW.

Return type
float

GetRealPowerSendZerokW() → **float**

Returns the zero branch phase sequence send end real power in kW.

Returns
The zero branch phase sequence send end real power in kW.

Return type
float

GetReactivePowerSendPoskVAr() → **float**

Returns the positive branch phase sequence send end reactive power in kVAr.

Returns
The positive branch phase sequence send end reactive power in kVAr.

Return type
float

GetReactivePowerSendNegkVAr() → **float**

Returns the negative branch phase sequence send end reactive power in kVAr.

Returns
The negative branch phase sequence send end reactive power in kVAr.

Return type
float

GetReactivePowerSendZerokVAr() → **float**

Returns the zero branch phase sequence send end reactive power in kVAr.

Returns
The zero branch phase sequence send end reactive power in kVAr.

Return type
float

GetRealPowerRecvPosMW() → **float**

Returns the positive branch phase sequence receive end real power in MW.

Returns
The positive branch phase sequence receive end real power in MW.

Return type
float

GetRealPowerRecvNegMW() → **float**

Returns the negative branch phase sequence receive end real power in MW.

Returns
The negative branch phase sequence receive end real power in MW.

Return type
float

GetRealPowerRecvZeroMW() → **float**

Returns the zero branch phase sequence receive end real power in MW.

Returns
The zero branch phase sequence receive end real power in MW.

Return type
float

GetReactivePowerRecvPosMVAr() → **float**

Returns the positive branch phase sequence receive end reactive power in MVAr.

Returns
The positive branch phase sequence receive end reactive power in MVAr.

Return type
float

GetReactivePowerRecvNegMVAr() → **float**

Returns the negative branch phase sequence receive end reactive power in MVAr.

Returns
The negative branch phase sequence receive end reactive power in MVAr.

Return type
float

GetReactivePowerRecvZeroMVAr() → **float**

Returns the zero branch phase sequence receive end reactive power in MVAr.

Returns
The zero branch phase sequence receive end reactive power in MVAr.

Return type
float

GetRecvPowerPosMVA() → **float**

Returns the positive branch phase sequence receive end power in MVA.

Returns
The positive branch phase sequence receive end power in MVA.

Return type
float

GetRecvPowerNegMVA() → **float**

Returns the negative branch phase sequence receive end power in MVA.

Returns
The negative branch phase sequence receive end power in MVA.

Return type
float

GetRecvPowerZeroMVA() → **float**

Returns the zero branch phase sequence receive end power in MVA.

Returns
The zero branch phase sequence receive end power in MVA.

Return type
float

GetRealPowerRecvPoskW() → **float**

Returns the positive branch phase sequence receive end real power in kW.

Returns
The positive branch phase sequence receive end real power in kW.

Return type
float

GetRealPowerRecvNegkW() → **float**

Returns the negative branch phase sequence receive end real power in kW.

Returns
The negative branch phase sequence receive end real power in kW.

Return type
float

GetRealPowerRecvZerokW() → **float**

Returns the zero branch phase sequence receive end real power in kW.

Returns
The zero branch phase sequence receive end real power in kW.

Return type
float

GetReactivePowerRecvPoskVAr() → **float**

Returns the positive branch phase sequence receive end reactive power in kVAr.

Returns
The positive branch phase sequence receive end reactive power in kVAr.

Return type
float

GetReactivePowerRecvNegkVAr() → **float**

Returns the negative branch phase sequence receive end reactive power in kVAr.

Returns
The negative branch phase sequence receive end reactive power in kVAr.

Return type
float

GetReactivePowerRecvZerokVAr() → **float**

Returns the zero branch phase sequence receive end reactive power in kVAr.

Returns
The zero branch phase sequence receive end reactive power in kVAr.

Return type
float

GetRecvPowerPoskVA() → **float**

Returns the positive branch phase sequence receive end power in kVA.

Returns
The positive branch phase sequence receive end power in kVA.

Return type
float

GetRecvPowerNegkVA() → **float**

Returns the negative branch phase sequence receive end power in kVA.

Returns
The negative branch phase sequence receive end power in kVA.

Return type
float

GetRecvPowerZerokVA() → **float**

Returns the zero branch phase sequence receive end power in kVA.

Returns
The zero branch phase sequence receive end power in kVA.

Return type
float

1.35 IscAnnotation

The *IscAnnotation* class provides access to a diagram annotation allowing annotation text to be set and cleared.

1.35.1 Field Values

Table 32: **IscAnnotation Field Values**

Type	Field Name	Description
String	HTMLText	Gets or sets the text displayed in the annotation. A limited set of simple HTML formats is supported.

1.35.2 IscAnnotation Class

class *ipsa.**IscAnnotation***

Provides access to a diagram annotation allowing annotation text to be set and cleared.

SetName(*strName:* ***str***) → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

GetIValue(*nFieldIndex:* ***int***) → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The integer value.

Return type

int

GetDValue(*nFieldIndex:* ***int***) → **float**

Returns a double value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The double value.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns
True if successful.

Return type
bool

SetSValue(*nFieldIndex: **int**, strValue: **int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns
True if successful.

Return type
bool

SetBValue(*nFieldIndex: **int**, bValue: **bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns
True if successful.

Return type
bool

1.36 IscProtectionDevice

The *IscProtectionDevice* class provides access to a single protection device, such as a relay, allowing data to be set and cleared.

1.36.1 Field Values

Table 33: **IscProtectionDevice Field Values**

Type	Field Name	Description
Integer	FromUID	Gets the unique ID of the nearest busbar to the protection device.
String	BusName	Gets of the nearest busbar to the protection device.
String	Name	Gets and sets the name of the protection device.

continues on next page

Table 33 – continued from previous page

Type	Field Name	Description
Integer	Status	Status: • 0 = Switched in • -1 = Switched out
String	DeviceManufac-turer	Gets the name of the manufacturer for the relay assigned to the protection device.
String	DeviceFamily	Gets the name of the relay family for the relay assigned to the protection device.
String	DeviceDBName	Gets the data base name of the relay assigned to the protection device.
String	DeviceVersion	Gets the version text of the relay assigned to the protection device.
String	DeviceComments	Gets the comments for the relay assigned to the protection device.
Float	OCNominalCur-rentA	Gets the UID nominal operating current of the relay in Amps.

1.36.2 IscProtectionDevice Class

class** ipsa.**IscProtectionDevice

Provides access to a single protection device, such as a relay.

***SetName**(strName: **str**)* → **bool**

Sets the name as a string.

Parameters

strName (***str***) – The selected string name.

Returns

True if successful.

Return type

bool

***GetIValue**(nFieldIndex: **int**)* → **int**

Returns an integer value for the enumerated field.

Parameters

nFieldIndex (***int***) – The field index.

Returns

The integer value.

Return type
int

GetDValue(*nFieldIndex: **int***) → **float**

Returns a double value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The double value.

Return type
float

GetSValue(*nFieldIndex: **int***) → **str**

Returns a string value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The string value.

Return type
str

GetBValue(*nFieldIndex: **int***) → **bool**

Returns a boolean value for the enumerated field.

Parameters
nFieldIndex (***int***) – The field index.

Returns
The boolean value.

Return type
bool

SetIValue(*nFieldIndex: **int**, nValue: **int***) → **bool**

Sets the value for the enumerated field from an integer.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **nValue** (***int***) – The given integer value.

Returns
True if successful.

Return type
bool

SetDValue(*nFieldIndex: **int**, dValue: **float***) → **bool**

Sets the value for the enumerated field from a double.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **dValue** (***float***) – The given double value.

Returns

True if successful.

Return type

bool

SetSValue(*nFieldIndex: **int**, strValue: **int***) → **bool**

Sets the value for the enumerated field from a string.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **strValue** (***str***) – The given string value.

Returns

True if successful.

Return type

bool

SetBValue(*nFieldIndex: **int**, bValue: **bool***) → **bool**

Sets the value for the enumerated field from boolean.

Parameters

- **nFieldIndex** (***int***) – The field index.
- **bValue** (***bool***) – The given boolean value.

Returns

True if successful.

Return type

bool