



POWERFACTORY

PowerFactory 2021

Python Function Reference

POWER SYSTEM SOLUTIONS
MADE IN GERMANY

F2021

Publisher:

DlgSILENT GmbH
Heinrich-Hertz-Straße 9
72810 Gomaringen / Germany
Tel.: +49 (0) 7072-9168-0
Fax: +49 (0) 7072-9168-88
info@digsilent.de

Please visit our homepage at:
<https://www.digsilent.de>

Copyright © 2021 DlgSILENT GmbH

All rights reserved. No part of this
publication may be reproduced or
distributed in any form without written
permission of DlgSILENT GmbH.

February 3, 2021
Revision 4

Contents

1	General Description	1
2	PowerFactory Module	2
3	Application Methods	4
3.1	File System	31
3.2	Date/Time	32
3.3	Dialogue Boxes	32
3.4	Environment	34
3.5	Mathematics	38
3.6	Output Window	44
4	Output Window	46
5	Object Methods	48
5.1	General Methods	48
5.2	Network Elements	74
5.2.1	ElmArea	74
5.2.2	ElmAsm	76
5.2.3	ElmAsmsc	79
5.2.4	ElmBbone	81
5.2.5	ElmBmu	83
5.2.6	ElmBoundary	84
5.2.7	ElmBranch	86
5.2.8	ElmCabsys	86
5.2.9	ElmComp	87
5.2.10	ElmCoup	87
5.2.11	ElmDsl	89
5.2.12	ElmFeeder	90
5.2.13	ElmFile	93
5.2.14	ElmFilter	93
5.2.15	ElmGenstat	94

5.2.16	ElmGndswt	97
5.2.17	ElmLne	99
5.2.18	ElmLnesec	104
5.2.19	ElmNec	105
5.2.20	ElmNet	105
5.2.21	ElmPvsys	107
5.2.22	ElmRelay	109
5.2.23	ElmRes	115
5.2.24	ElmShnt	128
5.2.25	ElmStactrl	128
5.2.26	ElmSubstat	130
5.2.27	ElmSvs	134
5.2.28	ElmSym	134
5.2.29	ElmTerm	138
5.2.30	ElmTr2	144
5.2.31	ElmTr3	147
5.2.32	ElmTr4	151
5.2.33	ElmTrfstat	156
5.2.34	ElmVac	158
5.2.35	ElmVoltreg	159
5.2.36	ElmXnet	160
5.2.37	ElmZone	162
5.3	Station Elements	164
5.3.1	StaCt	164
5.3.2	StaCubic	164
5.3.3	StaExtbrkmea	168
5.3.4	StaExtcmdmea	173
5.3.5	StaExtdatmea	178
5.3.6	StaExtfmea	183
5.3.7	StaExtfuelmea	188
5.3.8	StaExttimea	193
5.3.9	StaExtpfmea	198
5.3.10	StaExtpmea	203
5.3.11	StaExtqmea	208
5.3.12	StaExtsmea	213
5.3.13	StaExttapmea	217
5.3.14	StaExtv3mea	223
5.3.15	StaExtvmea	228
5.3.16	StaSwitch	233
5.4	Commands	234

5.4.1	ComAddlabel	234
5.4.2	ComAddon	235
5.4.3	ComAmpacity	246
5.4.4	ComAuditlog	246
5.4.5	ComBoundary	247
5.4.6	ComCapo	247
5.4.7	ComCimdbexp	249
5.4.8	ComCimdbimp	249
5.4.9	ComCimvalidate	250
5.4.10	ComConreq	253
5.4.11	ComContingency	254
5.4.12	ComCoordreport	260
5.4.13	ComDllmanager	270
5.4.14	ComDpl	270
5.4.15	ComFlickermeter	276
5.4.16	ComGenrelinc	276
5.4.17	ComGridtocim	277
5.4.18	ComHostcap	278
5.4.19	ComImport	279
5.4.20	ComInc	279
5.4.21	ComLdf	280
5.4.22	ComLink	282
5.4.23	ComMerge	284
5.4.24	ComMot	289
5.4.25	ComNmink	290
5.4.26	ComOmr	291
5.4.27	ComOpc	292
5.4.28	ComOutage	293
5.4.29	ComPfdimport	295
5.4.30	ComPrjconnector	295
5.4.31	ComProtgraphic	296
5.4.32	ComPvcurves	296
5.4.33	ComPython	296
5.4.34	ComRed	300
5.4.35	ComRel3	300
5.4.36	ComRelpost	302
5.4.37	ComRelreport	303
5.4.38	ComRes	303
5.4.39	ComShc	304
5.4.40	ComShctrace	306

5.4.41	ComSim	309
5.4.42	ComSimoutage	311
5.4.43	ComSvgexport	316
5.4.44	ComSvgimport	317
5.4.45	ComTasks	317
5.4.46	ComTececo	325
5.4.47	ComTransfer	325
5.4.48	ComUcte	326
5.4.49	ComUcteexp	326
5.4.50	ComWktimp	329
5.5	Settings	330
5.5.1	SetCluster	330
5.5.2	SetColscheme	331
5.5.3	SetDataext	337
5.5.4	SetDeskpage	339
5.5.5	SetDesktop	340
5.5.6	SetDistrstate	346
5.5.7	SetFilt	346
5.5.8	SetLevelvis	346
5.5.9	SetParalman	349
5.5.10	SetPath	350
5.5.11	SetSelect	352
5.5.12	SetTboxconfig	355
5.5.13	SetTime	356
5.5.14	SetUser	357
5.5.15	SetVipage	358
5.6	Others	362
5.6.1	BlkDef	362
5.6.2	BlkSig	364
5.6.3	ChaVecfile	365
5.6.4	CimModel	365
5.6.5	CimObject	370
5.6.6	GrpPage	374
5.6.7	IntAddonvars	379
5.6.8	IntCase	383
5.6.9	IntComtrade	385
5.6.10	IntComtradeset	391
5.6.11	IntDataset	397
5.6.12	IntDocument	398
5.6.13	IntDplmap	399

5.6.14 IntDplvec	402
5.6.15 IntEvt	404
5.6.16 IntExtaccess	405
5.6.17 IntGate	405
5.6.18 IntGrf	405
5.6.19 IntGrfgroup	406
5.6.20 IntGrflayer	407
5.6.21 IntGrfnet	408
5.6.22 Intlcon	409
5.6.23 IntLibrary	410
5.6.24 IntMat	411
5.6.25 IntMon	417
5.6.26 IntOutage	419
5.6.27 IntPlannedout	421
5.6.28 IntPlot	422
5.6.29 IntPrj	423
5.6.30 IntPrjfolder	435
5.6.31 IntQlim	436
5.6.32 IntRas	437
5.6.33 IntRunarrange	438
5.6.34 IntScenario	438
5.6.35 IntScensched	442
5.6.36 IntScheme	444
5.6.37 IntSscheduler	445
5.6.38 IntSstage	446
5.6.39 IntSubset	449
5.6.40 IntThrating	450
5.6.41 IntUrl	451
5.6.42 IntUser	451
5.6.43 IntUserman	452
5.6.44 IntVec	454
5.6.45 IntVecobj	456
5.6.46 IntVersion	458
5.6.47 IntViewbookmark	459
5.6.48 PltDataserie	459
5.6.49 PltLinebarplot	461
5.6.50 RelZpol	465
5.6.51 ScnFreq	466
5.6.52 ScnFrt	467
5.6.53 ScnSpeed	469

5.6.54 ScnSync	471
5.6.55 ScnVar	472
5.6.56 ScnVolt	474
5.6.57 StoMaint	476
5.6.58 TypAsmo	476
5.6.59 TypCtcore	476
5.6.60 TypLne	477
5.6.61 TypQdsl	478
5.6.62 TypTr2	479
5.6.63 VisBdia	480
5.6.64 VisDraw	482
5.6.65 VisHrm	484
5.6.66 VisMagndiffplt	486
5.6.67 VisOcplot	488
5.6.68 VisPath	490
5.6.69 VisPcompdiffplt	491
5.6.70 VisPlot	493
5.6.71 VisPlot2	498
5.6.72 VisPlottz	505
5.6.73 VisVec	506
5.6.74 VisVecres	506
5.6.75 VisXyplot	507
Index	509

1 General Description

This reference manual describes the syntax of all available functions and methods provided by the **PowerFactory** module. The used Python interface version is 2 (new in **PowerFactory** 2016). For syntax of the Python interface version 1 (**PowerFactory** 15.x) please use the [DPL reference](#) with the extended return values.

Please refer to the **PowerFactory User Manual** for general information about Python as scripting language and its usage.

2 **PowerFactory** Module

Overview

[__version__](#)
[GetApplication](#)
[GetApplicationExt](#)

__version__

Holds the version of **PowerFactory** e.g. "17.0.9" for **PowerFactory** 2017 SP7.

```
powerfactory.__version__
```

GetApplication

Creates a `powerfactory.Application` object and returns it. When the Python script is started from external **PowerFactory** will be started. Please hold the created application object as long as you use **PowerFactory** from Python and do not call this function twice.

```
Application powerfactory.GetApplication([str username = None,]  
                                       [str password = None,]  
                                       [str commandLineArguments = None])
```

ARGUMENTS

username (optional)

Name of the user to log in to **PowerFactory** (default None). None enforces the default behaviour as if PowerFactory was started via shortcut.

password (optional)

The password for the user which should be logged in (default None). None omits the password.

commandLineArguments (optional)

Additional command line options (default None). These need to be specified in the same way as if **PowerFactory** was started via a command shell. None omits the command line arguments.

RETURNS

Application object on success, otherwise None.

SEE ALSO

[GetApplicationExt\(\)](#)

GetApplicationExt

Same as `powerfactory.GetApplication()` but throwing an exception with an error code when **PowerFactory** was not able to start. Furthermore if **PowerFactory** exits due to an error, `powerfactory.ExitError` can be caught in an Python script for doing clean-up. A detailed description of all error codes can be found in the [Error Codes](#) documentation.

When **PowerFactory** was started by a Python script from external and the application object is deleted then **PowerFactory** is terminated but the current process keeps running. In this situation it is not possible to start **PowerFactory** again in the same process.

```
Application powerfactory.GetApplicationExt([str username = None,]  
                                           [str password = None,]  
                                           [str commandLineArguments = None])
```

RETURNS

Application object on success. In the error case an exception with a detailed error code is thrown.

EXAMPLE

The following example starts **PowerFactory** from external and handles initialisation errors:

```
import sys;  
sys.path.append(r"C:\Program Files\DIgSILENT\PowerFactory 2017\python\3.6")  
  
import powerfactory  
  
try:  
    app = powerfactory.GetApplicationExt()  
    # ... some calculations ...  
except powerfactory.ExitError as error:  
    print(error)  
    print('error.code = %d' % error.code)
```

SEE ALSO

[GetApplication\(\)](#)

3 Application Methods

Overview

ActivateProject
ClearRecycleBin
CommitTransaction
CreateFaultCase
CreateProject
DefineTransferAttributes
DeleteUntouchedObjects
ExecuteCmd
GetActiveCalculationStr
GetActiveNetworkVariations
GetActiveProject
GetActiveScenario
GetActiveScenarioScheduler
GetActiveStages
GetActiveStudyCase
GetAllUsers
GetAttributeDescription
GetAttributeUnit
GetBorderCubicles
GetBrowserSelection
GetCalcRelevantObjects
GetClassDescription
GetClassId
GetCurrentDiagram
GetCurrentScript
GetCurrentSelection
GetCurrentUser
GetCurrentZoomScaleLevel
GetDataFolder
GetDiagramSelection
GetFlowOrientation
GetFromStudyCase
GetGlobalLibrary
GetGraphicsBoard
GetInterfaceVersion
GetLanguage
GetLocalLibrary
GetMem
GetProjectFolder
GetRecordingStage
GetSettings
GetSummaryGrid

GetUserManager
Hide
ImportDz
ImportSnapshot
IsAttributeModelInternal
IsLdfValid
IsRmsValid
IsScenarioAttribute
IsShcValid
IsSimValid
IsWriteCacheEnabled
LoadProfile
MarkInGraphics
OutputFlexibleData
PostCommand
PrepForUntouchedDelete
Rebuild
ReloadProfile
ResetCalculation
ResGetData
ResGetDescription
ResGetFirstValidObject
ResGetFirstValidObjectVariable
ResGetFirstValidVariable
ResGetIndex
ResGetMax
ResGetMin
ResGetNextValidObject
ResGetNextValidObjectVariable
ResGetNextValidVariable
ResGetObject
ResGetUnit
ResGetValueCount
ResGetVariable
ResGetVariableCount
ResLoadData
ResReleaseData
ResSortToVariable
SaveAsScenario
SearchObjectByForeignKey
SelectToolbox
SetAttributeModelInternal
SetInterfaceVersion
SetShowAllUsers
SetWriteCacheEnabled
Show
SplitLine
StatFileGetXrange
StatFileResetXrange
StatFileSetXrange
WriteChangesToDb

ActivateProject

Activates a project with its name.

```
int Application.ActivateProject(str name)
```

ARGUMENTS

name Name ("Project"), full qualified name ("Project.IntPrj") or full qualified path ("\\User\\Project.IntPrj") of a project.

RETURNS

0 on success and 1 if project can not be found or activated.

ClearRecycleBin

Clears the recycle bin of currently logged in user.

```
None Application.ClearRecycleBin()
```

CommitTransaction

Writes pending changes to database.

While a script is running none of the changes are written to the database unless the script terminates. **PowerFactory** can be forced to write all pending changes to the database using this function.

```
None Application.CommitTransaction()
```

CreateFaultCase

Create fault cases from the given elements.

```
int Application.CreateFaultCase(set elms,  
                                int mode,  
                                [int createEvt],  
                                [DataObject folder]  
                                )
```

ARGUMENTS

elms Selected elements to create fault cases.

mode How the fault cases are created:

- 0** Single fault case containing all elements.
- 1** n-1 (multiple cases).
- 2** n-2 (multiple cases).
- 3** Collecting coupling elements and create fault cases for line couplings.

createEvt (optional)

Switch event:

- 0** Do NOT create switch events.
- 1** Create switch events.

folder (optional)

Folder in which the fault case is stored.

RETURNS

- 0** On success.
- 1** On error.

CreateProject

Creates a new Project inside the parent object. The default project stored in the Configuration/Default folder will be copied and if it contains any Study Cases the first will be used instead of creating a new one. A new grid will always be created. Returns the newly created project.

```
DataObject Application.CreateProject(str projectName,
                                   str gridName,
                                   [DataObject parent])
```

ARGUMENTS

projectName

Name of the new project. Leave empty to open up the IntPrj dialog and let the user enter a name.

gridName

Name of the grid that's created for the new project. Leave empty to open up the ElmNet dialog and let the user enter a name.

parent

The parent for the new project. Can be omitted to use the currently logged on user as default.

DefineTransferAttributes

Defines for a given classname the attributes for blockwise access via [DataObject.GetAttributes\(\)](#) and [DataObject.SetAttributes\(\)](#).

```
None Application.DefineTransferAttributes(str classname,
                                         str attributes
                                         )
```

ARGUMENTS

classname

Name of a class.

attributes Comma separated names of attributes.

SEE ALSO

[DataObject.GetAttributes\(\)](#), [DataObject.SetAttributes\(\)](#)

DeleteUntouchedObjects

Delete all objects stored in the grid (or stored in set) which are not modified. Requires call of [PrepForUntouchedDelete\(\)](#) first, see [Application.PrepareForUntouchedDelete\(\)](#). Hint: function should only be used when a variation is active. For details please contact support.

```
int Application.DeleteUntouchedObjects(DataObject grid)
int Application.DeleteUntouchedObjects(list grids)
```

ExecuteCmd

Executes given command string as it would be executed if typed directly into the Input Window. Current script will continue after the command has been executed.

This function is mainly intended for testing purpose and should be used by experienced users only.

```
None Application.ExecuteCmd(str command)
```

ARGUMENTS

command
The command string

GetActiveCalculationStr

Gets “calculation string” of currently valid calculation.

```
str Application.GetActiveCalculationStr()
```

RETURNS

None basic
Load Flow ldf
AC Load Flow Sensitivities acsens
AC Contingency Analysis acct
DC Load Flow dclfd
DC Load Flow Sensitivities dcsens
DC Contingency Analysis dccont
VDE/IEC Short-Circuit shc
Complete Short-Circuit shcfull
ANSI Short-Circuit shcansi
IEC 61363 shc61363
RMS-Simulation rms
Modal Analysis modal
EMT-Simulation emt
Harmonics/Power Quality harm
Frequency Sweep fsweep
Optimal Power Flow opf
DC Optimal Power Flow dcopf
DC OPF with Contingencies dccontopf
State Estimation est
Reliability rel
General Adequacy genrel
Tie Open Point Opt. topo
Motor Starting Calculation motstart
Arc Flash Calculation arcflash
Optimal Capacitor Placement optcapo
Voltage Plan Optimization mvplan
Backbone Calculation backbone
Optimal RCS Placement optrcs

GetActiveNetworkVariations

Returns all active variations for the 'Network Data' folder.

```
list Application.GetActiveNetworkVariations()
```

RETURNS

Returns currently active *IntScheme* objects. Set is empty in case of no scheme being currently active.

GetActiveProject

This function returns the currently active project.

```
DataObject Application.GetActiveProject()
```

RETURNS

Returns currently active *IntPrj* object or None in case of no project being currently active.

GetActiveScenario

Returns the currently active scenario. None is returned if there is no active scenario.

```
DataObject Application.GetActiveScenario()
```

RETURNS

Returns currently active *IntScenario* object or None in case of no scenario being currently active.

GetActiveScenarioScheduler

Returns currently active scenario scheduler.

```
DataObject Application.GetActiveScenarioScheduler()
```

RETURNS

Returns currently active *IntScensched* object or None in case of no scheduler being currently active.

GetActiveStages

Returns all active stages currently active for a given folder, e.g. 'Network Data' folder.

```
list Application.GetActiveStages([DataObject variedFolder])
```

ARGUMENTS

variedFolder (optional)

Folder for which all active stages will be returned; by default, the project folder 'Network Data' is taken.

RETURNS

Returns currently active *IntSstage* objects. Set is empty in case of no stages being currently active.

GetActiveStudyCase

Returns the active Study Case. None is returned if there is no active study case.

```
DataObject Application.GetActiveStudyCase()
```

RETURNS

The active study case (*IntCase* object) or None.

RETURNS

Returns currently active *IntCase* object or None in case of no study case being currently active.

GetAllUsers

Returns all known users, regardless of any Data Manager filters.

ARGUMENTS

forceReload

- | | |
|---|--|
| 0 | Default, returns the cached state if function was called before. |
| 1 | Forces the cache to be cleared, may impact performance. |

RETURNS

Returns a container with all known users.

GetAttributeDescription

Returns the description of an attribute.

```
str Application.GetAttributeDescription(str classname,  
                                       str name,  
                                       int short = 0)
```

ARGUMENTS

classname

Name of a class.

name

Name of an attribute.

short

- | | |
|---|--|
| 0 | Return long attribute description (default). |
| 1 | Return short attribute description. |

RETURNS

- | | |
|-------------|---|
| None | For an invalid class or attribute name. |
| str | Long or short attribute description. |

SEE ALSO

[Application.GetAttributeUnit\(\)](#)

GetAttributeUnit

Returns the unit of an attribute e.g. km, MW....

```
str Application.GetAttributeUnit(str classname,
                                str name
                                )
```

ARGUMENTS

classname Name of a class.
name Name of an attribute.

RETURNS

None For an invalid class or attribute name.
str Attribute unit.

SEE ALSO

[Application.GetAttributeDescription\(\)](#)

GetBorderCubicles

This function returns the border cubicles of the parent station of passed element topologically reachable from that element.

A cubicle (*StaCubic*) is considered to be a border cubicle if it resides inside the station

- and points to an element that sits outside the station
- or to a branch element that is connected to a terminal outside the station.

```
list Application.GetBorderCubicles(DataObject element)
```

ARGUMENTS

element Element from which the search for border cubicles starts

RETURNS

A set, containing border cubicles *StaCubic*. If the element does not reside in any substation or no border cubicles exist, the set is empty.

GetBrowserSelection

Returns all objects marked in the “on top” Data Manager (Browser, right side).

```
list Application.GetBrowserSelection()
```

RETURNS

Objects marked in the “on top” Data Manager (Browser, right side).

SEE ALSO

[Application.GetCurrentSelection\(\)](#), [Application.GetDiagramSelection\(\)](#)

GetCalcRelevantObjects

Returns all currently calculation relevant objects, i.e. the objects which are used by the calculations.

The set of objects depends on active study case, active grid(s) and variation(s).

In contrast to [DataObject.IsCalcRelevant\(\)](#) it does not return objects of the active study case e.g. simulation events (Evt*).

```
list Application.GetCalcRelevantObjects([str nameFilter,]
                                         [int includeOutOfService = 1,]
                                         [int topoElementsOnly = 0,]
                                         [int bAcSchemes = 0])
```

ARGUMENTS

nameFilter (optional)

(Class) name filter. Wildcards are supported. Multiple filters to be separated by comma ','. Must not contain a backslash '\'.
If omitted, all objects are returned (corresponds to '*.*').

Examples for valid filter strings:

- 'ElmTerm'
- 'A*.ElmTerm'
- '*.ElmLod,*.ElmSym'

includeOutOfService (optional)

Flag whether to include out of service objects. Default is 1 (=included).

topoElementsOnly (optional)

Flag to filter for topology relevant objects only. Default is 0 (=all objects).

bAcSchemes (optional)

Flag to include hidden objects in active schemes. Default is 0 (=not included).

RETURNS

The currently calculation relevant objects, according to the given arguments. The order of the set is undefined.

SEE ALSO

[DataObject.IsCalcRelevant\(\)](#)

GetClassDescription

Returns a description for a PowerFactory class.

```
str Application.GetClassDescription(str name)
```

ARGUMENTS

name Name of a **PowerFactory** class

RETURNS

Returns the description of a valid **PowerFactory** class, otherwise an empty string.

GetClassId

Returns a class identifier number.

Each class name corresponds to one unique number. The mapping of class name might be different for different build numbers of **PowerFactory**, but it is guaranteed that it will not be changed while an Api instance exists. (Do not keep these numbers static, get them dynamically in your code using this method).

```
int Application.GetClassId(str className)
```

ARGUMENTS

className
Class name e.g. "ElmTerm".

- 0** Class name invalid.
- >0** Class id of valid class name.

GetCurrentDiagram

This function offers access to the current diagram object (*IntGrfnet*).

```
DataObject Application.GetCurrentDiagram()
```

GetCurrentScript

Returns the current script (ComPython).

```
DataObject Application.GetCurrentScript()
```

RETURNS

The current script (ComPython) or None if started from external.

GetCurrentSelection

Returns all objects marked in the "on top" Data Manager (Browser, right side) or diagram.

```
list Application.GetCurrentSelection()
```

RETURNS

Objects marked in the "on top" Data Manager (Browser, right side) or diagram.

SEE ALSO

[Application.GetBrowserSelection\(\)](#), [Application.GetDiagramSelection\(\)](#)

GetCurrentUser

Returns the PowerFactory user of current session.

```
DataObject Application.GetCurrentUser()
```

RETURNS

Returns an *IntUser* object, never None.

GetCurrentZoomScaleLevel

Returns the zoom or scale level of the currently active diagram. If the active diagram is geographic, then the scale level is returned, otherwise the zoom level is returned.

```
int Application.GetCurrentZoomScale()
```

RETURNS

Zoom or scale level of the active diagram as integer.

- For geographic diagrams the scale level is returned. E.g. returns 50000 if 1:50000 is in the zoom/ratio combo box
- For all other diagrams the zoom level is returned. E.g. returns 150 if 150

A value of -1 is returned in case of no open diagram.

GetDataFolder

This function returns the folder in which the network data for the given class are stored.

```
DataObject Application.GetDataFolder(str classname,  
                                     [int iCreate])
```

ARGUMENTS

classname

Classname of the elements:

ElmBmu
ElmArea
ElmZone
ElmRoute
ElmOwner
ElmOperator
ElmFeeder
ElmCircuit
ElmBoundary
IntScales

iCreate(optional)

- | | |
|----------|---|
| 0 | The folder is searched and returned if found. If the folder does not exist, None is returned. |
| 1 | The folder is created if it does not exist. The found or created folder is returned. |

RETURNS

The network data folder, which is found or created.

SEE ALSO

[DataObject.IsNetworkDataFolder\(\)](#)

GetDiagramSelection

Returns all objects marked in the “on top” diagram.

```
list Application.GetDiagramSelection()
```

RETURNS

Objects marked in the “on top” diagram.

SEE ALSO

[Application.GetCurrentSelection\(\)](#), [Application.GetBrowserSelection\(\)](#)

GetFlowOrientation

This function returns the flow orientation setting of the active project.

```
int Application.GetFlowOrientation()
```

RETURNS

- 1** No project is active
- 0** Flow orientation of active project is “MIXED MODE”
- 1** Flow orientation of active project is “LOAD ORIENTED”
- 2** Flow orientation of active project is “GENERATOR ORIENTED”

GetFromStudyCase

Returns the first found object of class “className” from the currently active study case. The object is created when no object of the given name and/or class was found.

For commands the returned instance corresponds to the one that is used if opened via the main menu load-flow, short-circuit, transient simulation, etc.,

```
DataObject Application.GetFromStudyCase(str className)
```

ARGUMENTS

className

Class name of the object (“Class”), optionally preceded by an object name without wildcards and a dot (“Name.Class”).

RETURNS

The found or created object.

GetGlobalLibrary

Returns the global library for object-types of class “ClassName”. ClassName may be omitted, in which case the complete global library folder is returned.

```
DataObject Application.GetGlobalLibrary([str ClassName])
```

ARGUMENTS

ClassName (optional)

The classname of the objects for which the library folder is sought

RETURNS

The library folder

SEE ALSO

[Application.GetLocalLibrary\(\)](#)

GetGraphicsBoard

Returns the currently active Graphics Board.

```
DataObject Application.GetGraphicsBoard()
```

RETURNS

The graphics board object

GetInterfaceVersion

Returns the currently set interface version.

It holds the value set with SetInterfaceVersion() or the interface version from the current script (parameter 'interfaceVersion') if the python script is executed from within PowerFactory.

Have a look into the PowerFactor user manual to get more informations about the interface version of a script.

```
int Application.GetInterfaceVersion()
```

RETURNS

The currently set interface version or 0 if PowerFactory is started from external and SetInterfaceVersion() is not called.

GetLanguage

Returns a string for the current program language setting.

```
str Application.GetLanguage()
```

RETURNS

en	English
de	German
es	Spanish
fr	French

ru	Russian
cn	Simplified Chinese
tr	Turkish

GetLocalLibrary

Returns the local library for object-types of class “ClassName”. ClassName may be omitted, in which case the complete local library folder is returned.

```
DataObject Application.GetLocalLibrary([str ClassName])
```

ARGUMENTS

ClassName (optional)

The classname of the objects for which the library folder is sought

RETURNS

The library folder

SEE ALSO

[Application.GetGlobalLibrary\(\)](#)

GetMem

Allows to trace memory consumption (current working set size).

```
int GetMem([int calculateDelta=0],  
           [int inMegaByte=0]  
           )
```

ARGUMENTS

calculateDelta (optional)

Measure absolute memory consumption if 0, measure the delta since the last time it was called if 1 (default: 0).

inMegaByte (optional)

Returns consumption in byte if 0, in megabyte if 1 (default: 0).

RETURNS

The current working set size or the delta since the last call.

GetProjectFolder

Returns the project folder of a given type of active project. For each type (except ‘Generic’) there exist not more than one folder per type.

```
DataObject Application.GetProjectFolder(str type, [int create])
```

ARGUMENTS

type Type of the corresponding project folder. See [IntPrjfolder.GetProjectFolderType\(\)](#) for a list of possible values.

create Optional, default=0. Determines whether folder shall be created if it does not exist (1=create, 0=don’t create).

RETURNS

An *IntPrjFolder* object. If no project is currently active or project folder of this type does not exist and 'create' is not given as 0, None is returned.

GetRecordingStage

Returns the currently active recording scheme stage.

```
DataObject Application.GetRecordingStage()
```

RETURNS

An *IntSstage* object; None if there is no recording stage.

GetSettings

Offers read-only access to some selected **PowerFactory** settings.

```
str Application.GetSettings(str key)
```

ARGUMENTS

key

Key	Return type	Description
username	string	Name of logged-in user
installationdir	string	Fully qualified path of installation directory of PowerFactory
workingdir	string	Fully qualified path of working directory of PowerFactory
tempdir	string	Fully qualified path of temporary directory used by PowerFactory
sessionid	integer	ID of current session
db_driver	string	Name of used database driver
logfile	string	Path of current log file

RETURNS

Value of settings as string

GetSummaryGrid

Returns the summary grid in the currently active Study Case. The summary grid is the combination of all active grids in the study case.

```
DataObject Application.GetSummaryGrid()
```

RETURNS

A *ElmNet* object, or a 'None' object when no grids are active

GetUserManager

Offers access to the user manager object (IntUserman) stored in the configuration folder.

```
DataObject Application.GetUserManager()
```

RETURNS

The user manager object

Hide

Hides the **PowerFactory** application window.

```
None Application.Hide()
```

SEE ALSO

[Application.Show\(\)](#)

ImportDz

Imports a DZ file. Overwrites data if it already exists.

```
[int errorCode,  
list importedObjects] Application.ImportDz(DataObject target,  
                                           str dzFilePath)
```

ARGUMENTS

target Target object for imported data.

dzFilePath
Path to the DZ file that should be imported.

importedObjects (out)
Collection of top-level objects imported.

RETURNS

0 Success
-1 Wrong file extension
nonzero Import failed

ImportSnapshot

Imports a Snapshot DZS file.

```
[int errorCode,  
list importedObjects ] Application.ImportSnapshot(str dzsFilePath)
```

ARGUMENTS

dzsFilePath
Path to the DZ file that should be imported.

importedObjects (out)
Collection of top-level objects imported.

RETURNS

0 Success
-1 Wrong file extension
nonzero Import failed

IsAttributeModelInternal

Returns whether the attribute values are accessed as internally stored.

```
int Application.IsAttributeModeInternal()
```

RETURNS

- 0** Attribute values accessed as displayed in **PowerFactory** (unit conversion applied).
- 1** Attribute values accessed as internally stored.

SEE ALSO

[Application.SetAttributeModelInternal\(\)](#)

IsLdfValid

Checks to see if the last load-flow results are still valid and available.

```
int Application.IsLdfValid()
```

RETURNS

- 0 if no load-flow results are available

IsRmsValid

Checks to see if the last RMS simulation results are still valid and available.

```
int Application.IsRmsValid()
```

RETURNS

- 0 if no RMS simulation results are available

IsScenarioAttribute

Checks if a given attribute of a given class is recorded in scenario. It does not check whether a concrete instance is recorded at all. The check is just performed against the scenario configuration and is independent of a concrete scenario.

```
int Application.IsScenarioAttribute(str classname, str attributename)
```

ARGUMENTS

- classname*
Name of a **PowerFactory** class
- attributename*
Name of an attribute of given class

RETURNS

- 1** If attribute is scenario relevant according to current scenario configuration
- 0** If attribute is not scenario relevant

IsShcValid

Checks to see if the last short-circuit results are still valid and available.

```
int Application.IsShcValid()
```

RETURNS

0 if no short-circuit results are available

IsSimValid

Checks to see if the last simulation results are still valid and available.

```
int Application.IsSimValid()
```

RETURNS

0 if no simulation results are available

IsWriteCacheEnabled

Returns whether or not the cache method for optimizing performances is enabled.

```
int Application.IsWriteCacheEnabled()
```

RETURNS

- | | |
|----------|--------------------------|
| 0 | Write cache is disabled. |
| 1 | Write cache is enabled. |

SEE ALSO

[Application.SetWriteCacheEnabled\(\)](#), [Application.WriteChangesToDb\(\)](#)

LoadProfile

Activates a profile for current user. This corresponds to the select profile action via main menu "TOOLS-Profiles".

```
int Application.LoadProfile(str profileName)
```

ARGUMENTS

<i>profileName</i>	Name of profile to be loaded.
--------------------	-------------------------------

RETURNS

- | | |
|----------|---|
| 0 | On error, e.g. profile with given name not found. |
| 1 | On success. |

SEE ALSO

[Application.ReloadProfile\(\)](#)

MarkInGraphics

This function is not supported in GUI-less mode.

Marks all objects in the diagram in which the elements are found by hatch crossing them.

```
None Application.MarkInGraphics(list objects,
                                [int searchOpenedDiagramsOnly = 0])
```

ARGUMENTS

objects Objects to be marked.

searchOpenedDiagramsOnly (optional)

Search can be restricted to currently shown diagrams on the desktop, instead of all diagrams.

- 0** Searching all diagrams, not only the ones which are currently shown on the desktop. If there is more than one occurrence the user will be prompted which diagrams shall be opened (default).
- 1** Only search in currently opened diagrams and open the first diagram in which the elements were found.

OutputFlexibleData

Outputs the Flexible Data of the given objects to the output window.

Has identical functionality to that implemented in the Object Filter dialogue, whereby the user can right-click on a single row or multiple rows in a Flexible Data page and select Output ... Flexible Data. The OutputFlexibleData() function assumes that the user has already defined a Flexible Data page for the objects in the set. Upon execution of this function, all Flexible Data defined for the objects in the set is output to the **PowerFactory** output window in a tabular format.

```
None Application.OutputFlexibleData(list objects,
                                    [str flexibleDataPage = ''])
```

ARGUMENTS

objects Objects to output the Flexible Data for.

flexibleDataPage (optional)

Name of the Flexible Data page to be outputted. If multiple Flexible Data pages are defined and no or an empty string is given then a dialog to select a Flexible Data page is shown.

PostCommand

Adds a command to the command pipe of the “input window”. The posted commands will be executed after the currently running script has finished.

```
None Application.PostCommand(str command)
```

ARGUMENTS

command The command string.

PrepForUntouchedDelete

Mark (as not modified) all objects stored in the grid (or stored in set). Required for function [Application.DeleteUntouchedObjects\(\)](#).

```
void Application.PrepForUntouchedDelete(DataObject grid)
void Application.PrepForUntouchedDelete(list grids)
```

Rebuild

Rebuilds the currently visible single line diagram or plot.

```
None Application.Rebuild([int iMode])
```

ARGUMENTS

iMode (optional)

- | | |
|----------|---|
| 0 | Draws graphic objects only |
| 1 | (default) Reads graphic objects (IntGrf) from database and draws |
| 2 | For single line diagrams: Reads graphic objects (IntGrf) from database, re-calculates intersections and draws.
For plot pages: Adjust view to page format and re-create plots. |

ReloadProfile

Reloads currently selected user profile. (See main menu "TOOLS-Profiles")

```
None Application.ReloadProfile()
```

SEE ALSO

[Application.LoadProfile\(\)](#)

ResetCalculation

Resets all calculations and deletes all calculation results.

Results that have been written to result objects (for display in graphs) will not be destroyed. All results that are visible in the single line diagrams, however, will be destroyed.

```
None Application.ResetCalculation()
```

SEE ALSO

[Application.IsAutomaticCalculationResetEnabled\(\)](#), [Application.SetAutomaticCalculationResetEnabled\(\)](#)

ResGetData

This function is deprecated. Please use [ElmRes.GetValue\(\)](#) or [IntComtrade.GetValue\(\)](#) instead.

```
[int error,
float d    ] Application.ResGetData(DataObject resultObject,
                                   int iX,
                                   [int col])
```

ResGetDescription

This function is deprecated. Please use [ElmRes.GetDescription\(\)](#) or [IntComtrade.GetDescription\(\)](#) instead.

```
str Application.ResGetDescription(DataObject resultObject,
                                int col,
                                [int ishort])
```

ResGetFirstValidObject

This function is deprecated. Please use [ElmRes.GetFirstValidObject\(\)](#) instead.

```
int Application.ResGetFirstValidObject(DataObject resultFile,
                                       int row,
                                       [str classNames]
                                       [str variableName,]
                                       [float limit,]
                                       [int limitOperator,]
                                       [float limit2,]
                                       [int limitOperator2])
int Application.ResGetFirstValidObject([DataObject resultFile,]
                                       [int row,]
                                       [list objects])
```

ResGetFirstValidObjectVariable

This function is deprecated. Please use [ElmRes.GetFirstValidObjectVariable\(\)](#) instead.

```
int Application.ResGetFirstValidObjectVariable(DataObject resultFile,
                                              [str variableNames])
```

ResGetFirstValidVariable

This function is deprecated. Please use [ElmRes.GetFirstValidVariable\(\)](#) instead.

```
int Application.ResGetFirstValidVariable(DataObject resultFile,
                                       int row,
                                       [str variableNames])
```

ResGetIndex

This function is deprecated. Please use [ElmRes.FindColumn\(\)](#) or [IntComtrade.FindColumn\(\)](#) instead.

```
int Application.ResGetIndex(DataObject resultFile,
                           DataObject obj,
                           [str varName])
int Application.ResGetIndex(DataObject resultFile,
                           DataObject obj,
                           [int colIndex])
int Application.ResGetIndex(DataObject resultFile,
                           [str varName,]
                           [int colIndex])
```


ResGetMax

This function is deprecated. Please use [ElmRes.FindMaxInColumn\(\)](#) or [IntComtrade.FindMaxInColumn\(\)](#) instead.

```
[int row,
float value] Application.ResGetMax(DataObject resultFile,
                                int col)
```

ResGetMin

This function is deprecated. Please use [ElmRes.FindMinInColumn\(\)](#) or [IntComtrade.FindMinInColumn\(\)](#) instead.

```
[int row,
float value] Application.ResGetMin(DataObject resultFile,
                                int col)
```

ResGetNextValidObject

This function is deprecated. Please use [ElmRes.GetNextValidObject\(\)](#) instead.

```
int Application.ResGetNextValidObject(DataObject resultFile,
                                     [str classNames]
                                     [str variableName,]
                                     [float limit,]
                                     [int limitOperator,]
                                     [float limit2,]
                                     [int limitOperator2])
int Application.ResGetNextValidObject(DataObject resultFile,
                                     list objects)
```

ResGetNextValidObjectVariable

This function is deprecated. Please use [ElmRes.GetNextValidObjectVariable\(\)](#) instead.

```
int Application.ResGetNextValidObjectVariable(DataObject resultFile,
                                             [str variableNames])
```

ResGetNextValidVariable

This function is deprecated. Please use [ElmRes.GetNextValidVariable\(\)](#) instead.

```
int Application.ResGetNextValidVariable(DataObject resultFile,
                                       [str variableNames])
```

ResGetObject

This function is deprecated. Please use [ElmRes.GetObject\(\)](#) instead.

```
DataObject Application.ResGetObj(DataObject resultObject,
                                int col)
```

ResGetUnit

This function is deprecated. Please use [ElmRes.GetUnit\(\)](#) or [IntComtrade.GetUnit\(\)](#) instead.

```
str Application.ResGetUnit(DataObject resultObject,  
                           int col)
```

ResGetValueCount

This function is deprecated. Please use [ElmRes.GetNumberOfRows\(\)](#) or [IntComtrade.GetNumberOfRows\(\)](#) instead.

```
int Application.ResGetValueCount(DataObject resultObject,  
                                [int col])
```

ResGetVariable

This function is deprecated. Please use [ElmRes.GetVariable\(\)](#) or [IntComtrade.GetVariable\(\)](#) instead.

```
str Application.ResGetVariable(DataObject resultObject,  
                              int col)
```

ResGetVariableCount

This function is deprecated. Please use [ElmRes.GetNumberOfColumns\(\)](#) or [IntComtrade.GetNumberOfColumns\(\)](#) instead.

```
int Application.ResGetVariableCount(DataObject resultObject)
```

ResLoadData

This function is deprecated. Please use [ElmRes.Load\(\)](#) or [IntComtrade.Load\(\)](#) instead.

```
None Application.ResLoadData(DataObject resultObject)
```

ResReleaseData

This function is deprecated. Please use [ElmRes.Release\(\)](#) or [IntComtrade.Release\(\)](#) instead.

```
None Application.ResReleaseData(DataObject resultObject)
```

ResSortToVariable

This function is deprecated. Please use [ElmRes.SortAccordingToColumn\(\)](#) or [IntComtrade.SortAccordingToColumn\(\)](#) instead.

```
int Application.ResSortToVariable(DataObject resultObject,  
                                  int col)
```

SaveAsScenario

Saves the operational data or relevant network elements as a new scenario.

```
DataObject Application.SaveAsScenario(str pName,
                                     int iSetActive)
```

ARGUMENTS

pName Name of the new scenario.

iSetActive

1 Activate the new scenario afterwards.

0 Do not activate the new scenario.

RETURNS

Returns newly created *IntScenario* object. None is returned in case of creation of a new scenario was not allowed (e.g. no active project).

SearchObjectByForeignKey

Searches for an object by foreign key within an active project.

```
DataObject Application.SearchObjectByForeignKey(str foreignKey)
```

ARGUMENTS

foreignKey
Foreign key

RETURNS

Object if found, otherwise None.

SelectToolbox

Sets tool box to be displayed at a switchable tool box group.

```
int Application.SelectToolbox(int toolbar,
                             str groupName,
                             str toolboxName)
```

ARGUMENTS

toolbar **1** Main tool bar
 2 Drawing tool bar (SGL)

groupName
Name of tool box group.

toolboxName
Name of tool box to be selected.

RETURNS

0 On error, e.g. no matching tool box found.

1 On success.

SetAttributeModelInternal

Changes the way how attribute values are accessed.

```
None Application.SetAttributeModeInternal(int internalMode)
```

ARGUMENTS

<i>internalMode</i>	0	Access attribute values as displayed in PowerFactory (unit conversion applied).
	1	Access attribute values as internally stored.

SEE ALSO

[Application.IsAttributeModelInternal\(\)](#)

SetInterfaceVersion

Sets the current interface version. Only values which can be set to the python script parameter 'interfaceVersion' are allowed. Setting the interface version does not affect the parameter 'interfaceVersion' of the current script.

Have a look into the PowerFactor user manual to get more informations about the interface version of a script.

```
int Application.SetInterfaceVersion(int version)
```

ARGUMENTS

<i>version</i>	interface version to be set
----------------	-----------------------------

RETURNS

0 if the version is successfully set, otherwise 1.

SetShowAllUsers

Enables or disables the filtering of all available users in data manager. All users are only visualised in data manager when enabled.

```
int Application.SetShowAllUsers(int enabled)
```

ARGUMENTS

<i>enabled</i>	0	Disabled, only Demo, Public Area Users and current user are shown
	1	Enabled, all available users are listed

RETURNS

Returns previous setting.

1	If enabled before.
0	If disabled before.

SetWriteCacheEnabled

This function intends to optimize performances or disable the value consistency check of attributes. In order to modify objects in **PowerFactory**, those must be set in a special edit mode before any value can be changed. Switching back and forth between edit mode and stored mode is time consuming; enabling the write cache flag will set objects in edit mode and they will not be switched back until WriteChangeToDb() is called.

Another purpose of this function is to disable the value consistency check executed whenever an attribute is set. Note: Enabling the write cache delays the value consistency check until WriteChangeToDb() is called or the write cache is disabled. This might be required when dependent attributes are set where the object is temporarily left in an invalid state until all attributes are set. In DPL please use SetConsistencyCheck() for the same purpose.

None Application.SetWriteCacheEnabled(int enabled)

ARGUMENTS

<i>enabled</i>	0	Disables the write cache.
	1	Enables the write cache.

SEE ALSO

[Application.IsWriteCacheEnabled\(\)](#), [Application.WriteChangesToDb\(\)](#)

Show

Shows the **PowerFactory** application window (only possible with a full license, not supported for engine licenses).

None Application.Show()

SEE ALSO

[Application.Hide\(\)](#)

SplitLine

Splits the passed line or ElmBranch-object at a given position

```
DataObject Application.SplitLine(DataObject Line,
                                [float percent = 50,]
                                [int createSwitchSide0 = 0,]
                                [int createSwitchSide1 = 0])
```

ARGUMENTS

double rPercent (optional)

Position in percent from the first connection side where line shall be split.

int iCreateSwitchSide0 (optional)

0 (default) Do not create switch on first connection side.

1 Create switch on first connection side.

int iCreateSwitchSide1 (optional)

0 (default) Do not create switch on second connection side.

1 Create switch on second connection side.

RETURNS

Inserted terminal
0 = error

StatFileGetXrange

Gets the x-range for the statistic result file.

```
[int error,
float min,
float max] Application.StatFileGetXrange()
```

ARGUMENTS

min (out) First point in time considered in statistics.
max (out) Last point in time considered in statistics.

RETURNS

0 If time range of statistic result file was found.
1 On errors (There is no statistic result file).

StatFileResetXrange

Reset the user defined x-range of the statistic result file. The complete x-range will be considered in the statistic results after calling this function.

```
None Application.StatFileResetXrange()
```

StatFileSetXrange

Sets the user defined x-range of the statistic result file. The statistic results consider only the given time range.

```
None Application.StatFileSetXrange(float min,
float max)
```

ARGUMENTS

min First point in time to be considered in statistics.
max Last point in time to be considered in statistics.

WriteChangesToDb

This function combined with [Application.SetWriteCacheEnabled\(\)](#) is meant to optimize performances. If the write cache flag is enabled all objects remain in edit mode until WriteChangesToDb is called and all the modifications made to the objects are saved into the database.

```
None Application.WriteChangesToDb()
```

SEE ALSO

[Application.SetWriteCacheEnabled\(\)](#), [Application.IsWriteCacheEnabled\(\)](#)

3.1 File System

Overview

[GetInstallationDirectory](#)
[GetTemporaryDirectory](#)
[GetWorkspaceDirectory](#)

GetInstallationDirectory

Returns the installation directory of **PowerFactory** .

```
str Application.GetInstallationDirectory()
```

RETURNS

Full path to installation directory of current **PowerFactory** .

DEPRECATED NAMES

GetInstallDir

SEE ALSO

[Application.GetTemporaryDirectory\(\)](#), [Application.GetWorkspaceDirectory\(\)](#)

GetTemporaryDirectory

Returns the temporary directory of used by **PowerFactory** .

```
str Application.GetTemporaryDirectory()
```

RETURNS

Full path to a directory where temporary data can be stored. This directory is also used by **PowerFactory** to store temporary data.

DEPRECATED NAMES

GetTempDir

SEE ALSO

[Application.GetWorkspaceDirectory\(\)](#), [Application.GetInstallationDirectory\(\)](#)

GetWorkspaceDirectory

Returns the workspace directory of **PowerFactory** .

```
str Application.GetWorkspaceDirectory()
```

RETURNS

Full path to the directory where currently used workspace is stored.

DEPRECATED NAMES

`GetWorkingDir`

SEE ALSO

[Application.GetTemporaryDirectory\(\)](#), [Application.GetInstallationDirectory\(\)](#)

3.2 Date/Time

Overview

[GetStudyTimeObject](#)

GetStudyTimeObject

Returns the date and time object (`SetTime`) from the study case. This is the object being used by the characteristics, scenarios, ...

RETURNS

`SetTime` or `None`.

3.3 Dialogue Boxes

Overview

[CloseTableReports](#)
[GetTableReports](#)
[ShowModalBrowser](#)
[ShowModalSelectBrowser](#)
[ShowModelessBrowser](#)
[UpdateTableReports](#)

CloseTableReports

This function is not supported in GUI-less mode.

Closes all open table reports.

Please note: Table reports currently running one of their scripts are not closed.

```
None Application.CloseTableReports()
```

GetTableReports

This function is not supported in GUI-less mode.

Returns all open table reports.

```
list Application.GetTableReports()
```


ShowModalBrowser

This function is not supported in GUI-less mode.

Opens a modal browser window and lists all given objects.

```
None Application.ShowModalBrowser(list objects,
                                   [int detailMode = 0,]
                                   [str title = '',]
                                   [str page = ''])
```

ARGUMENTS

objects Objects to be listed. The listing is in detailed mode, if only one kind of objects (e.g. only ElmTerm) is contained.

detailMode (optional)

- 0** Show browser in normal mode (default).
- 1** Show browser in detail mode.

title (optional)

String for user defined window title. The default window title is shown when no or an empty string is given.

page (optional)

Name of page to be shown in browser e.g. 'Flexible Data' (only in detailed mode). The default page is shown when no or an empty string is given.

ShowModalSelectBrowser

This function is not supported in GUI-less mode.

Opens a modal browser window and lists all given objects. The user can make a selection from the list.

```
list Application.ShowModalSelectBrowser(list objects,
                                         [str title,]
                                         [str classFilter,]
                                         [str page = ''])
```

ARGUMENTS

objects Objects to be listed. The listing is in detailed mode, if only one kind of objects (e.g. only ElmTerm) is contained.

title (optional)

String for user defined window title. The default window title is shown when no or an empty string is given.

classFilter (optional)

Class name filter. If set, only objects matching that filter will be listed in the dialog e.g. 'Elm*', 'ElmTr?' or 'ElmTr2,ElmTr3'.

page (optional)

Name of page to be shown in browser e.g. 'Flexible Data' (only in detailed mode). The default page is shown when no or an empty string is given.

RETURNS

Set of selected objects. The set is empty if “cancel” is pressed.

ShowModelessBrowser

This function is not supported in GUI-less mode.

Opens a modeless browser window and lists all given objects.

```
None Application.ShowModelessBrowser(list objects,
                                     [int detailMode = 0,]
                                     [str title = '',]
                                     [str page = ''])
```

ARGUMENTS

objects Objects to be listed. The listing is in detailed mode, if only one kind of objects (e.g. only ElmTerm) is contained.

detailMode (optional)

- 0** Show browser in normal mode (default).
- 1** Show browser in detail mode.

title (optional)

String for user defined window title. The default window title is shown when no or an empty string is given.

page (optional)

Name of page to be shown in browser e.g. 'Flexible Data' (only in detailed mode). The default page is shown when no or an empty string is given.

UpdateTableReports

This function is not supported in GUI-less mode.

Updates all open table reports.

```
None Application.UpdateTableReports()
```

3.4 Environment

Overview

[EchoOff](#)
[EchoOn](#)
[IsAutomaticCalculationResetEnabled](#)
[IsFinalEchoOnEnabled](#)
[SetAutomaticCalculationResetEnabled](#)
[SetFinalEchoOnEnabled](#)
[SetGraphicUpdate](#)
[SetGuiUpdateEnabled](#)
[SetProgressBarUpdatesEnabled](#)
[SetUserBreakEnabled](#)

EchoOff

Freezes (de-activates) the user-interface. For each `EchoOff()`, an `EchoOn()` should be called. An `EchoOn()` is automatically executed at the end of the execution of a `ComDpl` or `ComPython`. This could be changed with [Application.SetFinalEchoOnEnabled\(\)](#).

```
None Application.EchoOff()
```

SEE ALSO

[Application.EchoOn\(\)](#), [Application.IsFinalEchoOnEnabled\(\)](#), [Application.SetFinalEchoOnEnabled\(\)](#)

EchoOn

Re-activates the user interface. For more informations see [Application.EchoOff\(\)](#).

```
None Application.EchoOn()
```

SEE ALSO

[Application.EchoOff\(\)](#), [Application.IsFinalEchoOnEnabled\(\)](#), [Application.SetFinalEchoOnEnabled\(\)](#)

IsAutomaticCalculationResetEnabled

Returns whether the automatic calculation reset while setting attributes is enabled. See [Application.SetAutomaticCalculationResetEnabled\(\)](#) for more informations.

```
int Application.IsAutomaticCalculationResetEnabled()
```

SEE ALSO

[Application.SetAutomaticCalculationResetEnabled\(\)](#), [Application.ResetCalculation\(\)](#)

IsFinalEchoOnEnabled

Returns whether the automatic [Application.EchoOn\(\)](#) at the end of each *ComDpl* or *ComPython* is enabled.

```
int Application.IsFinalEchoOnEnabled();
```

RETURNS

- 1** Final [Application.EchoOn\(\)](#) is enabled.
- 0** Final [Application.EchoOn\(\)](#) is disabled.

SEE ALSO

[Application.SetFinalEchoOnEnabled\(\)](#), [Application.EchoOn\(\)](#), [Application.EchoOff\(\)](#)

SetAutomaticCalculationResetEnabled

Enables or disables the automatic calculation reset while setting attributes.

In Python/API the automatic calculation reset is by default enabled. Thus changing an object attribute could lead to a calculation reset, e.g. changing the scaling factor of a load, but do not have to, e.g. renaming an object.

Even if the automatic calculation reset is disabled, changing the "outserv" attribute of an arbitrary network element or the "on_off" attribute of a switch device resets automatically the current calculation.

When the calculation is reset the load-flow will be calculated with a flat start. Thus switching the automatic calculation reset off can be helpful e.g. when calculating a load-flow without a flat start. On the other side it could lead to wrong results e.g. doing short-circuit calculations after changing the short-circuit-location of a branch without calling [Application.ResetCalculation\(\)](#).

```
None Application.SetAutomaticCalculationResetEnabled(int enabled)
```

SEE ALSO

[Application.IsAutomaticCalculationResetEnabled\(\)](#), [Application.ResetCalculation\(\)](#)

SetFinalEchoOnEnabled

Enables or disables the automatic [Application.EchoOn\(\)](#) at the end of each *ComDpl* or *ComPython*.

```
None Application.SetFinalEchoOnEnabled(int enabled);
```

ARGUMENTS

enabled

- | | |
|----------|---|
| 1 | Enables the final Application.EchoOn() . |
| 0 | Disables the final Application.EchoOn() . |

SEE ALSO

[Application.IsFinalEchoOnEnabled\(\)](#), [Application.EchoOn\(\)](#), [Application.EchoOff\(\)](#)

SetGraphicUpdate

Enables or disables the updates of the single line graphics.

```
None Application.SetGraphicUpdate(int enabled)
```

ARGUMENTS

enabled

- | | |
|----------|--|
| 0 | disabled (graphic will not be updated automatically) |
| 1 | enabled |

SetGuiUpdateEnabled

Enables or disables updates of the graphical user interface (e.g. application window) while the script is running.

This can be useful to get maximum execution performance. However, the user interface might look frozen and becomes not responsive.

Please note that the progress bar, which is located in the status bar of the application window, is not affected by this. Updates of the progress bar can be enabled or disabled separately by invoking [Application.SetProgressBarUpdatesEnabled\(\)](#).

The updates will automatically be re-enabled after termination of the script. In case of sub-scripts, the restore is done at termination of main script.

```
int Application.SetGuiUpdateEnabled(int enabled)
```

ARGUMENTS

enabled

- | | |
|----------|-----------------------|
| 0 | Disables GUI updates. |
| 1 | Enables GUI updates. |

RETURNS

Previous state before the function was called

- | | |
|----------|-----------------------------------|
| 0 | GUI updates were disabled before. |
| 1 | GUI updates were enabled before. |

DEPRECATED NAMES

SetRescheduleFlag

SEE ALSO

[Application.SetProgressBarUpdatesEnabled\(\)](#), [Application.SetGraphicUpdate\(\)](#)

SetProgressBarUpdatesEnabled

Enables or disables updates of the progress bar (located in the status bar of the application window) while the script is running. Other components of the status bar are not affected.

If a script executes a high number of small, fast commands that report progress (noticeable by the progress bar repeatedly and quickly filling up), disabling progress bar updates can provide an immense performance boost.

The updates will automatically be re-enabled after termination of the script. In case of sub-scripts, the restore is done at termination of main script.

```
int Application.SetProgressBarUpdatesEnabled(int enabled)
```

ARGUMENTS

enabled

- | | |
|----------|--------------------------------|
| 0 | Disables progress bar updates. |
| 1 | Enables progress bar updates. |

RETURNS

Previous state before the function was called

- | | |
|----------|--|
| 0 | Progress bar updates were disabled before. |
| 1 | Progress bar updates were enabled before. |

SEE ALSO

[Application.SetGuiUpdateEnabled\(\)](#), [Application.SetGraphicUpdate\(\)](#)

SetUserBreakEnabled

Enables or disables the “Break” button in main tool bar. After script execution it is disabled automatically.

```
None Application.SetUserBreakEnabled(int enabled)
```

ARGUMENTS

enabled

- | | |
|----------|--------------------------|
| 0 | Disables “Break” button. |
| 1 | Enable “Break” button. |

DEPRECATED NAMES

SetEnableUserBreak

3.5 Mathematics

Overview

[GetRandomNumber](#)
[GetRandomNumberEx](#)
[InvertMatrix](#)
[RndExp](#)
[RndGetMethod](#)
[RndGetSeed](#)
[RndNormal](#)
[RndSetup](#)
[RndUnifInt](#)
[RndUnifReal](#)
[RndWeibull](#)
[SetRandomSeed](#)

GetRandomNumber

This method is marked as deprecated since PowerFactory 2017. Please use [Application.RndUnifReal\(\)](#) instead.

Draws a uniformly distributed random number. Uses the 'global random number generator'. If x_1 and x_2 are omitted, the distribution will be uniform in the interval $[0, 1]$. If only x_1 is given, the distribution is uniform in $[0, x_1]$ and with both x_1 and x_2 , the distribution is uniform in $[x_1, x_2]$.

```
float Application.GetRandomNumber([float x1,]
                                  [float x2])
```

ARGUMENTS

x1 (optional)
 x_2 not given: maximum; x_1 and x_2 given: minimum

x2 (optional)
 maximum

RETURNS

A uniformly distributed random number

GetRandomNumberEx

This method is marked as deprecated since PowerFactory 2017. Please use [Application.RndUnifReal\(\)](#), [Application.RndNormal\(\)](#) or [Application.RndWeibull\(\)](#) instead.

Draws a random number according to a specific probability distribution. Uses the 'global random number generator'.

```
float Application.GetRandomNumberEx(int distribution,
                                   [float p1,]
                                   [float p2])
```

ARGUMENTS

distribution

- 0** uniform distribution
- 1** normal distribution
- 2** weibull distribution
- else** returns 0.0

p1 (optional)

- distribution = 0 (uniform), argument p2 is also given: min
- distribution = 0 (uniform), argument p2 is not given: max (min is assumed to be 0).
- distribution = 1 (normal) : mean
- distribution = 2 (weibull) : scale

p2 (optional)

- distribution = 0 (uniform) : max
- distribution = 1 (normal) : stddev
- distribution = 2 (weibull) : weibull

RETURNS

- double** Newly drawn random number from the specified distribution.
- 0.0** On failure e.g. non-supported mode.

InvertMatrix

This routine calculates the inverse matrix by the Gauss-Jordan method. It uses scaled partial pivoting preceded by column equilibration of the input matrix. The routine can be called in two different versions:

- **Real Inversion:** Only one matrix, *realPart*, is provided as an input to the function. Then, *realPart* is inverted and the result, $realPart^{-1}$, is stored into the input matrix *realPart* on success.
- **Complex Inversion:** Two matrices, *realPart* and *imaginaryPart*, are provided as inputs to this function. Then, a complex matrix *C* is formed, with entries

$$C(i,j) = A(i,j) + j \cdot imaginaryPart(i,j).$$

The complex matrix *C* is inverted and, on success, the resulting real part of C^{-1} is written to *realPart* whereas the resulting imaginary part of C^{-1} is written to *imaginaryPart*. Please note that *realPart* and *imaginaryPart* must have the same dimensions.

```
int Application.InvertMatrix(DataObject realPart,
                             [DataObject imaginaryPart])
```

ARGUMENTS

realPart If imaginaryPart is not set, realpart is the matrix to invert on input. In case of success, it will be overwritten by the inverted input matrix. If imaginaryPart is set, it holds the real part of the complex matrix to invert on input and is overwritten by the real part of the inverted complex matrix on output.

imaginaryPart

If this is set, it should hold the imaginary part of the matrix to invert on input and is overwritten by the imaginary part of the inverted matrix on output.

RETURNS

- 1** Matrix inversion failed. The provided input matrix is singular.
- 0** Matrix inversion was successful. Resulting inverted matrix returned in input matrix/matrices.

RndExp

Returns a random number distributed according to exponential distribution with given rate. See the example given in the DPL description of [Application.RndSetup\(\)](#).

```
float RndExp(float rate, [int rngNum])
```

ARGUMENTS

rate Rate of exponential distribution.

rngNum (optional)

Number of the random number generator.

0 (default) 'Global random number generator'.

1, 2, ... Other random number generators accessible via this number.

RETURNS

double Random number

RndGetMethod

Returns the used method of a random number generator. See the example given in the DPL description of [Application.RndSetup\(\)](#).

```
str RndGetMethod([int rngNum])
```

ARGUMENTS

rngNum (optional)

Number of the random number generator of which the method type is returned.

0 (default) 'Global random number generator'.

1, 2, ... Other random number generators accessible via this number.

RETURNS

string Name of the used method

RndGetSeed

Returns the used seed of a random number generator. See the example given in the DPL description of [Application.RndSetup\(\)](#).

```
int RndGetSeed([int rngNum])
```

ARGUMENTS

rngNum (optional)

Number of the random number generator.

0 (default) 'Global random number generator'.

1, 2, ... Other random number generators accessible via this number.

RETURNS

int Used seed

RndNormal

Returns a random number distributed according to normal distribution with given mean and standard deviation. See the example given in the DPL description of [Application.RndSetup\(\)](#).

```
float RndNormal(float mean, float stddev, [int rngNum])
```

ARGUMENTS

mean Mean of normal distribution.

stddev Standard deviation of normal distribution.

rngNum (optional)

Number of the random number generator.

0 (default) 'Global random number generator'.

1, 2, ... Other random number generators accessible via this number.

RETURNS

double Random number

RndSetup

Initializes a random number generator. Allows to choose:

1. Whether to seed automatically or not.
2. The seed, if not automatically seeded.
3. The type or random number generator.
4. The random number generator to use.

Supported types of random number generators:

1. Mersenne Twister,
2. Linear Congruential,

3. Additive Lagged Fibonacci.

Internally a vector of random number generators is used. These can be accessed via the number passed as last argument. Number 0 corresponds to the 'global random number generator', updated also in ComInc and ComGenrelinc. Numbers 1,2,... will access different random number generators, which can be setup individually.

```
None RndSetup(int seedAutomatic, [int seed], [int rngType], [int rngNum])
```

ARGUMENTS

seedAutomatic

Seed the random number generator automatically

- 0** Do not seed automatically.
- 1** Seed automatically.

seed (optional)

Seed for the random number generator. (default: 0) Note, that for the Additive Lagged Fibonacci generator, only the seeds 0,...,9 are supported.

rngType (optional)

Type of random number generator

- 0** Mersenne Twister (recommended) (default).
- 1** Linear Congruential.
- 2** Additive Lagged Fibonacci.

rngNum (optional)

Number of random number generator to be used

- 0 (default)** 'Global random number generator'.
- 1, 2, ...** Other random number generators accessible via this number.

RndUnifInt

Returns a random number distributed according to uniform distribution on the set of numbers $\{min, \dots, max\}$. See the example given in the DPL description of [Application.RndSetup\(\)](#).

```
int RndUnifInt(int min, int max, [int rngNum])
```

ARGUMENTS

min Smallest possible number

max Largest possible number

rngNum (optional)

Number of the random number generator.

- 0 (default)** 'Global random number generator'.
- 1, 2, ...** Other random number generators accessible via this number.

RETURNS

int Random number

RndUnifReal

Returns a random number distributed according to uniform distribution on the interval $[min, max]$. See the example given in the DPL description of [Application.RndSetup\(\)](#).

```
float RndUnifReal(float min, float max, [int rngNum])
```

ARGUMENTS

- min* Lower endpoint of interval $[min, max]$
- max* Upper endpoint of interval $[min, max]$
- rngNum* (*optional*)
Number of the random number generator.
0 (default) 'Global random number generator'.
1, 2, ... Other random number generators accessible via this number.

RETURNS

double Random number

RndWeibull

Returns a random number distributed according to Weibull distribution with given shape and scale parameters. See the example given in the DPL description of [Application.RndSetup\(\)](#).

```
float RndWeibull(float shape, float scale, [int rngNum])
```

ARGUMENTS

- shape* Shape parameter of Weibull distribution.
- scale* Scale parameter of Weibull distribution.
- rngNum* (*optional*)
Number of the random number generator.
0 (default) 'Global random number generator'.
1, 2, ... Other random number generators accessible via this number.

RETURNS

double Random number

SetRandomSeed

This method is marked as deprecated since PowerFactory 2017. Please use [Application.RndSetup\(\)](#) instead.

Initializes the 'global random number generator' as Additive Lagged Fibonacci random number generator. Sets the seed for the random number generator. One out of 10 predefined initialization seeds can be selected.

```
None Application.SetRandomSeed(int seed)
```

ARGUMENTS

seed seed 0..9

3.6 Output Window

Overview

[ClearOutputWindow](#)
[GetOutputWindow](#)
[PrintError](#)
[PrintInfo](#)
[PrintPlain](#)
[PrintWarn](#)
[SetOutputWindowState](#)

ClearOutputWindow

Clears the output window.

```
None Application.ClearOutputWindow()
```

GetOutputWindow

Returns the **PowerFactory** Output Window.

```
OutputWindow Application.GetOutputWindow()
```

RETURNS

An instance of the class `OutputWindow`.

EXAMPLE

```
import powerfactory
app = powerfactory.GetApplication()
outputWindow = app.GetOutputWindow()
outputWindow.Print(powerfactory.OutputWindow.MessageType.Plain, "Hello World!")
```

PrintError

Prints a message as error into the **PowerFactory** Output Window. See [OutputWindow.Print\(\)](#) for more informations.

```
None Application.PrintError(str message)
```

PrintInfo

Prints a message as information into the **PowerFactory** Output Window. See [OutputWindow.Print\(\)](#) for more informations.

```
None Application.PrintInfo(str message)
```

PrintPlain

Prints a message as normal text into the **PowerFactory** Output Window. See [OutputWindow.Print\(\)](#) for more informations.

```
None Application.PrintPlain(str message)
```

PrintWarn

Prints a message as warning into the **PowerFactory** Output Window. See [OutputWindow.Print\(\)](#) for more informations.

```
None Application.PrintWarn(str message)
```

SetOutputWindowState

Changes the display state of the output window.

```
None Application.SetOutputWindowState(int newState)
```

ARGUMENTS

newState

- | | |
|-----------|--------------------------|
| 0 | Minimized output window. |
| 1 | Maximized output window. |
| -1 | Restore previous state. |

4 Output Window

Overview

[Clear](#)
[GetContent](#)
[Print](#)
[Save](#)
[SetState](#)

Clear

Clears the output window.

```
None OutputWindow.Clear()
```

GetContent

Returns the content (unfiltered or filtered) of the output window.

```
[str] OutputWindow.GetContent()  
[str] OutputWindow.GetContent(OutputWindow.MessageType filter)
```

ARGUMENTS

filter (optional)

If given, then only text from messages of this type will be returned.

RETURNS

List of texts of output window messages (unfiltered or filtered).

SEE ALSO

[OutputWindow.Print\(\)](#)

Print

Prints a message as normal text, as information, as warning or as error into the **PowerFactory** Output Window.

```
None OutputWindow.Print(str message)  
None OutputWindow.Print(OutputWindow.MessageType type, str message)
```

ARGUMENTS

type Type of message to print:

OutputWindow.MessageType.Plain Normal text.

OutputWindow.MessageType.Info Information.

OutputWindow.MessageType.Warn Warning.

OutputWindow.MessageType.Error Error.

The message is printed as normal text if no message type is given.

message Message to print.

EXAMPLE

```
import powerfactory
app = powerfactory.GetApplication()
outputWindow = app.GetOutputWindow()
outputWindow.Print(powerfactory.OutputWindow.MessageType.Plain, "Hello World!")
```

Save

Saves the content of current output window to a file.

```
None OutputWindow.Save(str filePath)
```

ARGUMENTS

filePath Full file name with path e.g. d:\data\output.txt. Possible file formats are html, txt and out.

SetState

Changes the display state of the output window.

```
None OutputWindow.SetState(int newState)
```

ARGUMENTS

newState

- | | |
|-----------|--------------------------|
| 0 | Minimized output window. |
| 1 | Maximized output window. |
| -1 | Restore previous state. |

5 Object Methods

5.1 General Methods

Overview

AddCopy
ContainsNonAsciiCharacters
CopyData
CreateObject
Delete
Energize
GetAttribute
GetAttributeDescription
GetAttributeLength
GetAttributes
GetAttributeShape
GetAttributeType
GetAttributeUnit
GetChildren
GetClassName
GetCombinedProjectSource
GetConnectedElements
GetConnectionCount
GetContents
GetControlledNode
GetCubicle
GetFullName
GetImpedance
GetInom
GetNode
GetOperator
GetOwner
GetParent
GetReferences
GetRegion
GetSupplyingSubstations
GetSupplyingTransformers
GetSupplyingTrfstations
GetSystemGrounding
GetUnom
GetZeroImpedance
HasAttribute
HasResults
IsCalcRelevant

[IsDeleted](#)
[IsEarthed](#)
[IsEnergized](#)
[IsHidden](#)
[IsInFeeder](#)
[IsNetworkDataFolder](#)
[IsNode](#)
[IsObjectActive](#)
[IsObjectModifiedByVariation](#)
[Isolate](#)
[IsOutOfService](#)
[IsReducible](#)
[IsShortCircuited](#)
[MarkInGraphics](#)
[Move](#)
[PasteCopy](#)
[PurgeUnusedObjects](#)
[ReplaceNonAsciiCharacters](#)
[ReportNonAsciiCharacters](#)
[ReportUnusedObjects](#)
[SearchObject](#)
[SetAttribute](#)
[SetAttributeLength](#)
[SetAttributes](#)
[SetAttributeShape](#)
[ShowEditDialog](#)
[ShowModalSelectTree](#)
[SwitchOff](#)
[SwitchOn](#)
[WriteChangesToDb](#)

AddCopy

Adds a copy of a single object or a set of objects to this object (= target object).

When copying a single object it is possible to give the new name. The new name will be concatenated by the given name parts. This is not possible for a project (IntPrj).

The target object must be able to receive a copy of the objects.

Copying a set of objects will respect all internal references between those objects. Copying a set of lines and their types, for example, will result in a set of copied lines and line types, where the copied lines will use the copied line types.

When source object(s) and target object are inside different projects the method [DataObject.PasteCopy\(\)](#) has to be used instead, since it adapts all references automatically.

```

DataObject DataObject.AddCopy(DataObject objectToCopy,
                               [int|str partOfName0,]
                               [...])
DataObject DataObject.AddCopy(list objectsToCopy)

```

ARGUMENTS

objectToCopy

Object to copy.

objectNameParts (optional)

Parts of the name of the new copy which will be concatenated to the object name.

objectsToCopy

Set of objects to copy.

RETURNS

Returns the copy that has been created on success or None, when copying a single object.

SEE ALSO

[DataObject.PasteCopy\(\)](#), [DataObject.Move\(\)](#), [Application.Delete\(\)](#)

ContainsNonAsciiCharacters

Checks whether an object contains texts attributes with non-ASCII characters.

```
int DataObject.ContainsNonAsciiCharacters()
```

RETURNS

Returns 1 if the object contains at least one non-ASCII characters. Otherwise 0.

CopyData

Copies all parameters except for loc_name and containers from one object to another.

```
None DataObject.CopyData(DataObject source)
```

ARGUMENTS

source Object from which parameters are to be copied

RETURNS

0	ok
1	error

CreateObject

Creates a new object of given class and name in the target object. The object name will be concatenated by the given object name parts. The target object must be able to store an object of the given class in its content otherwise the currently running script will stop with an error.

```
DataObject DataObject.CreateObject(str className,
                                     [int|str objectNamePart0,]
                                     [...]
                                     )
```

ARGUMENTS

className
The class name of the object to create.

objectNameParts (optional)
Parts of the name of the object to create (without classname) which will be concatenated to the object name.

RETURNS

- object** Newly created object.
- None** When no object was created.

Delete

Deletes the object from the database. The object is not destroyed but moved to the recycle bin.

```
int DataObject.Delete()
```

RETURNS

- 0** Object successfully deleted.
- ≠ 0** Deletion failed e.g. not allowed.

SEE ALSO

[DataObject.CreateObject\(\)](#)

Energize

Performs an “energize” action on the network element. This corresponds to removing earthings from current region (if any) followed by a “switch on” action on the element. The action is identical to that in the context menu.

```
[int error,  
list changedSwitches] DataObject.Energize([int resetRA])
```

ARGUMENTS

changedSwitches (optional, out)

All switches whose switching state was changed by the action are filled into this set.

resetRA (optional)

Determines whether an active running arrangement that would prevent switching action should be deactivated or not.

- 1** All running arrangements that cause blocking of relevant switches are applied and reset automatically before the switching is performed.
- 0** (default) Active running arrangements are not reset. Blocked switches will cause the switching action to fail.

RETURNS

Information about the success of the action:

- 0** Action was successful.
- 1** Action failed.

SEE ALSO

[DataObject.SwitchOn\(\)](#), [DataObject.SwitchOff\(\)](#), [DataObject.Isolate\(\)](#)

GetAttribute

Returns the value of an attribute.

```
int|float|str|DataObject|list DataObject.GetAttribute(str name)
```

ARGUMENTS

name Name of an attribute.

RETURNS

Value of an attribute name in its current unit (like in the edit dialog seen). An exception is thrown for invalid attribute names.

SEE ALSO

[DataObject.SetAttribute\(\)](#), [DataObject.GetAttributeType\(\)](#)

GetAttributeDescription

Returns the description of an attribute.

```
str DataObject.GetAttributeDescription(str name,  
                                     int short = 0)
```

ARGUMENTS

name Name of an attribute.

short **0** Return long attribute description (default).
 1 Return short attribute description.

RETURNS

None For an invalid attribute name.
str Long or short attribute description.

SEE ALSO

[DataObject.GetAttributeType\(\)](#), [DataObject.GetAttributeUnit\(\)](#)

GetAttributeLength

Returns the length of a vector or matrix attribute. The length of a matrix attribute is the number of rows.

```
int DataObject.GetAttributeLength(str name)
```

ARGUMENTS

name Name of an attribute.

RETURNS

> 0 Length of a valid vector or matrix attribute.
0 All other valid attributes.
-1 For invalid attribute names.

SEE ALSO

[DataObject.GetAttributeShape\(\)](#), [DataObject.SetAttributeLength\(\)](#), [DataObject.GetAttributeType\(\)](#)

GetAttributes

Gets the values of the transfer attributes defined via [Application.DefineTransferAttributes\(\)](#).

```
list DataObject.GetAttributes()
```

RETURNS

List of values for the defined transfer attributes. An exception is thrown for invalid attribute names.

SEE ALSO

[DataObject.SetAttributes\(\)](#), [Application.DefineTransferAttributes\(\)](#)

GetAttributeShape

Returns the shape of an attribute. The shape is a list of the form [number of rows, number of columns].

```
[int rows,  
int columns ] DataObject.GetAttributeShape(str name)
```

ARGUMENTS

name Name of an attribute.

RETURNS

[≥ 0 , ≥ 0] Shape of a valid vector or matrix attribute.
 [0, 0] All other valid attributes.
 [-1, 0] For invalid attribute names.

SEE ALSO

[DataObject.GetAttributeLength\(\)](#), [DataObject.SetAttributeShape\(\)](#), [DataObject.GetAttributeType\(\)](#)

GetAttributeType

Returns the type of an attribute.

The following attribute types exist:

<i>AttributeType.INVALID</i>	Attribute does not exist.
<i>AttributeType.INTEGER</i>	Integer attribute.
<i>AttributeType.INTEGER_VEC</i>	Integer vector attribute.
<i>AttributeType.DOUBLE</i>	Double attribute.
<i>AttributeType.DOUBLE_VEC</i>	Double vector attribute.
<i>AttributeType.DOUBLE_MAT</i>	Double matrix attribute.
<i>AttributeType.OBJECT</i>	Data object attribute.
<i>AttributeType.OBJECT_VEC</i>	Data object vector attribute.
<i>AttributeType.STRING</i>	String attribute.
<i>AttributeType.STRING_VEC</i>	String vector attribute.
<i>AttributeType.INTEGER64</i>	64-bit integer attribute.
<i>AttributeType.INTEGER64_VEC</i>	64-bit integer vector attribute.

```
AttributeType DataObject.GetAttributeType(str name)
```

ARGUMENTS

name Name of an attribute.

RETURNS

The type of an attribute or *AttributeType.INVALID* for an invalid attribute name.

SEE ALSO

[DataObject.GetAttribute\(\)](#), [DataObject.SetAttribute\(\)](#)

GetAttributeUnit

Returns the unit of an attribute e.g. km, MW...

```
str DataObject.GetAttributeUnit(str name)
```

ARGUMENTS

name Name of an attribute.

RETURNS

None For an invalid attribute name.

str Attribute unit.

SEE ALSO

[DataObject.GetAttributeType\(\)](#), [DataObject.GetAttributeDescription\(\)](#)

GetChildren

This function returns the objects that are stored within the object the function was called on. In contrast to [DataObject.GetContents\(\)](#) this function gives access to objects that are currently hidden due to scheme management.

```
list DataObject.GetChildren(int hiddenMode,
                           [str filter,]
                           [int subfolders])
```

ARGUMENTS

hiddenMode

Determines how hidden objects are handled.

0 Hidden objects are ignored. Only nonhidden objects are returned.

1 Hidden objects and nonhidden objects are returned.

2 Only hidden objects are returned. Nonhidden objects are ignored.

filter (optional)

Name filter, possibly containing '*' and '?' characters.

subfolder (optional)

Determines if children of subfolders are returned.

0 Only direct children are returned, children of subfolders are ignored (Default).

1 Also children of subfolders are returned.

RETURNS

Objects that are stored in the called object.

SEE ALSO

[DataObject.GetContents\(\)](#)

GetClassName

Returns the class name of the object.

```
str DataObject.GetClassName()
```

RETURNS

The class name of the object.

GetCombinedProjectSource

For an object in a combined project return the intermediate folder where the object is contained, indicating the original source project.

```
DataObject DataObject.GetCombinedProjectSource()
```

RETURNS

The intermediate folder for that object or nothing when not applicable.

GetConnectedElements

Returns the set of connected elements. Only electrically connected elements are returned when the conditions of all switches are regarded. Possible connections will also be returned when rBrk and/or rDis is zero, in the case of open breakers and/or disconnectors. The default values are (0,0,0).

```
list DataObject.GetConnectedElements([int rBrk],  
                                      [int rDis],  
                                      [int rOut])
```

ARGUMENTS

rBrk (optional)

if 1, regards position of breakers

rDis (optional)

if 1, regards position of disconnectors

rOut (optional)

if 1, regards in-service or out-of-service status

RETURNS

The set of connected elements.

GetConnectionCount

Returns the number of electrical connections.

```
int DataObject.GetConnectionCount()
```

RETURNS

The number of electrical connections.

GetContents

Returns the objects that are stored in the object and whose name matches the argument name. No object is returned if the object's container is empty, or if the object is not capable of storing objects. The argument name may contain the complete name and classname, or parts of the name with wildcard and class name.

```
list DataObject.GetContents([str Name,]
                             [int recursive])
```

ARGUMENTS

Name (optional)

loc name.class name, name possibly contains wildcards: '*' and '?' characters

recursive (optional)

- 1** All contained objects will be added recursively.
- 0** (default) Only direct children of current object will be collected.

RETURNS

Objects that are stored in the object.

GetControlledNode

Returns the target terminal and the resulting target voltage for generators and other voltage regulating units.

```
[DataObject node,
float targetVoltage] DataObject.GetControlledNode(int bus,
                                                  [int check])
```

ARGUMENTS

bus)

- 1** currently controlled bus
- 0** HV bus
- 1** MV/ LV bus
- 2** LV bus

targetVoltage (out)

The target voltage of the voltage regulating unit in pu.

check (optional)

- 0** (default) Do not check if the control mode is set to voltage control.
- 1** Only return the controlled node if the control mode is set to voltage control.

RETURNS

Controlled node, None if no controlled terminal exists (or not voltage controlled if check=1)

GetCubicle

Returns the cubicle of an object at the connection with index n, or None if there is no cubicle inside the object.

```
DataObject DataObject.GetCubicle(int side)
```

ARGUMENTS

side The connection number.

RETURNS

The cubicle object or None.

GetFullName

Returns the full name of the object as a string.

```
str DataObject.GetFullName([int type])
```

ARGUMENTS

type(optional)

Is used to determine the format of the returned full name:

not given

No special formatting.

= 0

The full name (complete database path including the name and class name) of the object. It becomes a clickable link if printed to the output window.

> 0 (but less or equal to 190)

Formatted exactly to this length and also clickable if printed to the output window.

RETURNS

The fullname (complete database path including the name and class name) of the object.

GetImpedance

Returns the positive sequence impedance of an element referred to a given voltage.

```
[int error,  
float real,  
float imag] DataObject.GetImpedance(float refVoltage,  
[int i3Trf])
```

ARGUMENTS

real (out) Real part of the impedance in Ohm.

imag (out)

Imaginary part of the impedance in Ohm.

refVoltage

Reference voltage for the impedance in kV.

i3Trf (optional)

When used with an *ElmTr3*

- 0** Return the HV-MV impedance.
- 1** Return the HV-LV impedance.
- 2** Return the MV-LV impedance.

RETURNS

- 1** An error occurred.
- 0** Otherwise.

SEE ALSO

`object.GetZeroImpedance()`

GetInom

Returns the nominal current of the object at given bus index.

```
float DataObject.GetInom([int busIndex = 0],
                        [int inclChar = 0])
```

ARGUMENTS

busIndex (optional)

Bus index, default value is 0.

inclChar (optional)

option to consider thermal rating objects and values modified by characteristics on the (de-)rating factor.

- 0** Not considering thermal rating objects or values modified by characteristics on the (de-)rating factor (default).
- 1** Considering thermal rating objects and values modified by characteristics on the (de-)rating factor.
- 2** Considering thermal rating objects but not values modified by characteristics on the (de-)rating factor.

RETURNS

The nominal current at bus index.

SEE ALSO

[DataObject.GetUnom\(\)](#)

GetNode

Returns the node connected to the object at specified bus index.

```
DataObject DataObject.GetNode(int busIndex,
                              [int considerSwitches = 0])
```

ARGUMENTS

busIndex Bus index.

considerSwitches (optional)

- | | |
|----------|--|
| 0 | Ignore the status of the switches (default). |
| 1 | Consider the status of the switches. |

RETURNS

- object** Connected node object at specified bus index.
None If no node at bus index is found.

GetOperator

Returns the element's operator (*ElmOperator*).

```
DataObject DataObject.GetOperator()
```

RETURNS

Object of class *ElmOperator* determined according to following rules

- If operator is set in current object instance (attribute "pOperator") this operator object is returned.
- Else the operator inherited from its parent is used (recursively applied).
- None if none if its parents have an operator set.

GetOwner

Returns the elements's owner (*ElmOwner*).

```
DataObject DataObject.GetOwner()
```

RETURNS

Object of class *ElmOwner* determined according to following rules

- If owner is set in current object instance (attribute "pOwner") this owner object is returned.
- Else the owner inherited from its parent is used (recursively applied).
- None if none if its parents have an operator set.

GetParent

Returns the parent folder object (same as parameter 'fold_id').

```
DataObject DataObject.GetParent()
```

RETURNS

- DataObject** The parent folder object.
None On the root database folder e.g. parent of a user.

SEE ALSO

[DataObject.GetContents\(\)](#)

GetReferences

Returns a set containing all objects with references to the object the method was called on. By default, references from IntSubset objects or hidden objects are ignored.

```
list DataObject.GetReferences([str filter = '*',]
                             [int includeSubsets = 0,]
                             [int includeHiddenObjects = 0])
```

ARGUMENTS

filter (optional)

Object filter to get only objects whose name matches this filter string, e.g. '*.ElmLne'. (default: '*')

includeSubsets (optional)

Forces references from IntSubset objects to be evaluated. These are normally not included for performance reasons. (default: 0)

includeHiddenObjects (optional)

Include also hidden objects. By default they are not included. In contrast hidden objects are always included in the 'Reference List' output of the data browser. (default: 0)

RETURNS

Set of objects with references to the object the method was called on.

GetRegion

All network components are internally associated with an artificial region. A region consists of topologically connected elements. This means, two elements *elm1* and *elm2* are topologically connected \Leftrightarrow *elm1*.GetRegion() == *elm2*.GetRegion().

A region is simply identified by a number that can be access via this function.

```
int DataObject.GetRegion()
```

RETURNS

Region index >0. A value of '-1' means status is unknown for that element (normally for not topology relevant elements).

GetSupplyingSubstations

Returns the closest supplying substation(s) for a network component.

"Closest" means that there is no other supplying element of same type in topological path between network component and the supplying component(s) returned by this function.

```
list DataObject.GetSupplyingSubstations()
```

RETURNS

List of substations (objects of class ElmSubstat). Can be empty.

SEE ALSO

[ElmTr2.GetSuppliedElements\(\)](#), [ElmTr3.GetSuppliedElements\(\)](#), [ElmSubstat.GetSuppliedElements\(\)](#),
[ElmTrfstat.GetSuppliedElements\(\)](#), [DataObject.GetSupplyingTransformers\(\)](#), [DataObject.GetSupplyingTrfstations\(\)](#)

GetSupplyingTransformers

Returns the closest supplying transformer(s) for a network component. “Closest” means that there is no other supplying element of same type in topological path between network component and the supplying component(s) returned by this function.

```
list DataObject.GetSupplyingTransformers()
```

RETURNS

List of transformers (objects of class *ElmTr2* or *ElmTr3*). Can be empty.

SEE ALSO

[ElmTr2.GetSuppliedElements\(\)](#), [ElmTr3.GetSuppliedElements\(\)](#), [ElmSubstat.GetSuppliedElements\(\)](#),
[ElmTrfstat.GetSuppliedElements\(\)](#), [DataObject.GetSupplyingSubstations\(\)](#), [DataObject.GetSupplyingTrfstations\(\)](#)

GetSupplyingTrfstations

Returns the closest supplying transformer station(s) for a network component. “Closest” means that there is no other supplying element of same type in topological path between network component and the supplying component(s) returned by this function.

```
list DataObject.GetSupplyingTrfstations()
```

RETURNS

List of transformer stations (objects of class *ElmTrfstat*). Can be empty.

SEE ALSO

[ElmTr2.GetSuppliedElements\(\)](#), [ElmTr3.GetSuppliedElements\(\)](#), [ElmSubstat.GetSuppliedElements\(\)](#),
[ElmTrfstat.GetSuppliedElements\(\)](#), [DataObject.GetSupplyingTransformers\(\)](#), [DataObject.GetSupplyingSubstation](#)

GetSystemGrounding

Returns the grounding type employed in the grounding area of the grid the object belongs to. The grounding area is defined by network components separating the zero sequence system (e.g. star-delta transformers).

```
int DataObject.GetSystemGrounding()
```

RETURNS

- 1 grounding type can not be determined
- 0 system is solidly grounded
- 1 system is compensated
- 2 system is isolated

GetUnom

Returns the nominal voltage of the object.

```
float DataObject.GetUnom([int busIndex = 0])
```

ARGUMENTS

busIndex (optional)
Bus index, default value is 0.

RETURNS

The nominal voltage at bus index.

SEE ALSO

[DataObject.GetInom\(\)](#)

GetZeroImpedance

Returns the zero sequence impedance of an element referred to a given voltage.

```
[int error,  
float real,  
float imag] DataObject.GetZeroImpedance(float refVoltage,  
                                         [int i3Trf])
```

ARGUMENTS

real (out) Real part of the impedance in Ohm.

imag (out)
Imaginary part of the impedance in Ohm.

refVoltage
Reference voltage for the impedance in kV.

i3Trf (optional)
When used with an *ElmTr3*

- 0** Return the HV-MV impedance.
- 1** Return the HV-LV impedance.
- 2** Return the MV-LV impedance.

RETURNS

- 1** An error occurred.
- 0** Otherwise.

SEE ALSO

[object.GetImpedance\(\)](#)

HasAttribute

Returns whether the given name is a currently valid attribute name.

```
int DataObject.HasAttribute(str name)
```

ARGUMENTS

name Name of an attribute.

RETURNS

- 0** Given name is not a currently valid attribute name.
- 1** Given name is a currently valid attribute name.

SEE ALSO

[DataObject.GetAttribute\(\)](#), [DataObject.SetAttribute\(\)](#)

HasResults

Checks if the object has calculated result parameters.

```
int DataObject.HasResults([int ibus])
```

ARGUMENTS

ibus (optional)

Bus index

- 1(default)** Checks if "c:" quantities exist
- >= 0** Checks if 'm:xxxx:bus ' quantities exist for bus index=ibus
- 2** Hidden objects are returned

RETURNS

- 0** no results available
- 1** results exist

IsCalcRelevant

Returns whether the object is relevant for calculation.

In contrast to [Application.GetCalcRelevantObjects\(\)](#) it also returns 1 for objects in the active study case e.g. simulation events (Evt*).

```
int DataObject.IsCalcRelevant()
```

RETURNS

- 0** When the object is not used for calculations.
- 1** When the object is currently used for calculations.

SEE ALSO

[Application.GetCalcRelevantObjects\(\)](#)

IsDeleted

Returns 1 if the object is deleted.

```
int DataObject.IsDeleted()
```

RETURNS

- | | |
|----------|----------------------------|
| 1 | Object is already deleted. |
| 0 | Object is not deleted. |

IsEarthed

Checks if a network component is topologically connected to any earthed component. Earthing components are terminals / busbars (*ElmTerm*) with attribute 'iEarth' = 1 and all closed grounding switches (*ElmGndswt*). An energized component is never considered to be earthed.

```
int DataObject.IsEarthed()
```

RETURNS

- | | |
|----------|---|
| 1 | Component is earthed (connected to an earthing component) |
| 0 | Component is not earthed |

IsEnergized

Checks if a network component is energized. A component is considered to be energized, if it is topologically connected to a generator. All other elements are considered to be deenergized.

```
int DataObject.IsEnergized()
```

RETURNS

- | | |
|-----------|---|
| 1 | Component is energized |
| 0 | Component is deenergized |
| -1 | Component has no energizing status (status unknown) |

IsHidden

Checks whether an object is hidden with respect to currently activated variation. An object is hidden if it is

- *deleted* in currently active variation or
- *added* in a variation that is currently not active

```
int DataObject.IsHidden()
```

RETURNS

- | | |
|----------|--------------------------------|
| 0 | not hidden, currently 'active' |
| 1 | hidden, currently 'inactive' |

IsInFeeder

Checks if the object is part of the given feeder. A network element is considered being part of a feeder if a topological path from the feeder definition to the element exists.

This function is based on load flow calculation results. Therefore, it can only be used after such a calculation has been successfully executed and as long as the results are available.

```
int DataObject.IsInFeeder(DataObject Feeder,
                          [int OptNested=0])
```


ARGUMENTS

Feeder The Feeder definition object *ElmFeeder*

OptNested (optional)

- 0** Nested feeders are not considered.
- 1** Nested feeders are considered.

RETURNS

- 1** If “Feeder” is a feeder definition and the object is part of that feeder.
- 0** Otherwise

SEE ALSO

[ElmFeeder.GetAll\(\)](#)

IsNetworkDataFolder

Checks whether given object is a special folder within a project that stores specific data elements. Each project can not have more than one instance per folder type. The following folder types are distinguished (**PowerFactory** class names):

IntArea stores *ElmArea* objects

IntBbone stores *ElmBbone* and *SetBbone* objects

IntBmu stores *ElmBmu* objects

IntBoundary stores *ElmBoundary* objects

IntCircuit stores *ElmCircuit* objects

IntFeeder stores *ElmFeeder* objects

IntMeteostat stores *ElmMeteostat* objects

IntOperator stores *ElmOperator* objects

IntOwner stores *ElmOwner* objects

IntPath stores *SetPath* objects

IntRoute stores *ElmRoute* objects

IntScales stores *Tri** objects

```
int DataObject.IsNetworkDataFolder()
```

RETURNS

- 0** false, object is not a network data folder
- 1** true, object is a network data folder

SEE ALSO

[Application.GetDataFolder\(\)](#)

IsNode

Indicates whether an object is a node (terminal or busbar).

```
int DataObject.IsNode()
```

RETURNS

- 1** Object is a node.
- 0** Otherwise.

IsObjectActive

Check if an object is active for specific time.

```
int DataObject.IsObjectActive(int time)
```

ARGUMENTS

time Time in seconds since 01.01.1970 00:00:00.

RETURNS

- 0** Object is not active (hidden or deleted).
- 1** Object is active.

IsObjectModifiedByVariation

Check if an object is active for specific time.

```
int DataObject.IsObjectModifiedByVariation(int considerADD, int considerDEL, int considerDELTA)
```

ARGUMENTS

considerADD

checks if an ADD-object exists

- 0** ignore ADD-objects
- 1** consider ADD-objects

considerDEL

check if a DELETE-Object exists or exist for the parent objects

- 0** ignore DELETE-objects
- 1** consider DELETE-objects

considerDELTA

check if a DELTA-Object exists

- 0** ignore DELTA-objects
- 1** consider DELTA-objects

RETURNS

- 0** Object is not modified by an active variation
- 1** Object is modified by an active variation

Isolate

Performs an “isolate” action on the network element. This corresponds to performing a “switch off” action followed by an additional earthing of switched off region. The action is identical to that in the context menu.

```
[int error,
list changedSwitches] DataObject.Isolate([int resetRA,]
                                         [int isolateCBs])
```

ARGUMENTS

changedSwitches (optional, out)

All switches whose switching state was changed by the action are filled into this set

resetRA (optional)

Determines whether an active running arrangement that would prevent switching action should be deactivated or not.

- 1** All running arrangements that cause blocking of relevant switches are applied and reset automatically before the switching is performed.
- 0** (default) Active running arrangements are not reset. Blocked switches will cause the switching action to fail

isolateCBs (optional)

Determines if, in addition, circuit breakers should be isolated by opening its adjacent disconnectors (if not given, default will be taken from project settings)

- 0** No additional opening of disconnectors
- 1** Also open disconnectors adjacent to switched circuit breakers)

RETURNS

Information about the success of the action:

- 0** Action was successful
- 1** Action failed

SEE ALSO

[DataObject.SwitchOn\(\)](#), [DataObject.SwitchOff\(\)](#), [DataObject.Energize\(\)](#)

IsOutOfService

Indicates whether or not the object is currently out of service.

```
int DataObject.IsOutOfService()
```

RETURNS

- 0** When the object is in service.
- 1** When the object is out of service.

IsReducible

Checks if object can be reduced during network reduction.

```
int DataObject.IsReducible()
```

RETURNS

- 0** object can never be reduced.
- 1** object can be reduced (e.g. switch, zero-length lines)
- 2** in principle the object can be reduced, but not now (e.g. switch that is set to be detailed)

IsShortCircuited

Returns whether an element is short-circuited or not.

```
int DataObject.IsShortCircuited()
```

RETURNS

- 0** No short-circuit found.
- 1** Element is short-circuited.

MarkInGraphics

This function is not supported in GUI-less mode.

Marks the object in the diagram in which the element is found by hatch crossing it. By default all the currently opened diagrams are searched for the element to mark beginning with the diagram shown. The first diagram in which the element is found will be opened and the element is marked.

Alternatively the search can be extended to all existing diagrams by passing 1 as parameter. If the element exists in more than one diagram the user can select from a list of diagrams which diagram shall be opened.

```
None DataObject.MarkInGraphics([int searchAllDiagramsAndSelect = 0])
```

ARGUMENTS

searchAllDiagramsAndSelect (optional)

Search can be extended to all diagrams, not only the ones which are currently shown on the desktop.

- 0** Only search in currently opened diagrams and open the first diagram in which the element was found. (default)
- 1** Searching all diagrams, not only the ones which are currently shown on the desktop. If there is more than one occurrence the user will be prompted which diagrams shall be opened.

RETURNS

A diagram in which the element is drawn is opened and the element is marked.

Move

Moves an object or a set of objects to this folder.

```
int DataObject.Move(DataObject objectToMove)
int DataObject.Move(list objectsToMove)
```

ARGUMENTS

objectToMove

Object to move.

objectToMove

Set of objects to move.

RETURNS

0 On success.**1** On error.

SEE ALSO

[DataObject.AddCopy\(\)](#), [DataObject.PasteCopy\(\)](#), [Application.Delete\(\)](#)**PasteCopy**

Pastes a copy of a single object or a set of objects to this object (= target object) using the merge tool when source object(s) and target object are inside different projects (equivalent to a manual copy&paste operation).

```
[int error,
DataObject newObject] DataObject.PasteCopy(DataObject objectToCopy,
                                             [int resetMissingReferences = 0])
int DataObject.PasteCopy(list objectsToCopy)
```

ARGUMENTS

objectToCopy

Object to be copied.

objectsToCopy

Set of objects to copy.

newObject (out)

The new object copy of objectToCopy is assigned on success.

resetMissingReferences (optional)

Determines how to handle missing references.

0 No action is taken, the operation is cancelled with an error (default).**1** Missing references are automatically reset.

RETURNS

0 Object(s) successfully copied.**1** Error.

SEE ALSO

[DataObject.AddCopy\(\)](#), [DataObject.Move\(\)](#), [Application.Delete\(\)](#)**PurgeUnusedObjects**

The function deletes the following child objects:

1. All 'hidden' objects without corresponding "Add" object. These objects are only deleted, if the condition is fulfilled for all child objects (hidden without corresponding 'Add' object).

2. All internal expansion stage objects with invalid target object (target object reference is missing).

It's crucial that there is no study case active when executing the function.

```
None DataObject.PurgeUnusedObjects()
```

SEE ALSO

[DataObject.ReportUnusedObjects\(\)](#)

ReplaceNonAsciiCharacters

Replaces all non-ASCII characters in all text attributes by similar ASCII characters. Emits a warning if a character can not be replaced, because no replacement character was defined.

```
int DataObject.ReplaceNonAsciiCharacters(DataObject map,
                                         str defaultReplacementCharacter)
```

ARGUMENTS

map IntMat object with two columns: the first column contains the codes of the non-ASCII character, the second column contains the code of the ASCII character.

defaultReplacementCharacter
String containing one ASCII character. If map does not contain a replacement for a non-ASCII character, it is replaced by defaultReplacementCharacter.

RETURNS

Returns 1 when the function was executed successfully.

ReportNonAsciiCharacters

Reports all text attributes of this objects containing non-ASCII characters in the output window.

```
None DataObject.ReportNonAsciiCharacters()
```

ReportUnusedObjects

Prints a report in the PowerFactory output window, which object will be deleted when function [DataObject.PurgeUnusedObjects\(\)](#) is called. It's crucial that there is no study case active when executing the function.

```
None DataObject.ReportUnusedObjects()
```

SEE ALSO

[DataObject.PurgeUnusedObjects\(\)](#)

SearchObject

Searches for an object with a full name, such as 'rootfolder.class\subfolder.class\...\locname.class'.

```
DataObject DataObject.SearchObject(str name)
```

ARGUMENTS

name string to search

RETURNS

Returns the searched object.

SEE ALSO

[DataObject.GetFullName\(\)](#)

SetAttribute

Sets the value of an attribute.

```
None DataObject.SetAttribute(str name,  
                             int|float|str|DataObject|list value)
```

ARGUMENTS

name Name of an attribute.

value Value to be set in its current unit (like in the edit dialog seen). An exception is thrown for invalid attribute names.

SEE ALSO

[DataObject.GetAttribute\(\)](#), [DataObject.GetAttributeType\(\)](#)

SetAttributeLength

Sets the length of a vector or matrix attribute. The length of a matrix attribute is the number of rows.

```
int DataObject.SetAttributeLength(str name, int length)
```

ARGUMENTS

name Name of an attribute.

length New length of the attribute.

RETURNS

0 On success.

1 On error.

SEE ALSO

[DataObject.SetAttributeShape\(\)](#), [DataObject.GetAttributeLength\(\)](#), [DataObject.GetAttributeType\(\)](#)

SetAttributes

Sets the values of the transfer attributes defined via [Application.DefineTransferAttributes\(\)](#).

```
None DataObject.SetAttributes(list values)
```

ARGUMENTS

values Values to be set in its current unit (like in the edit dialog seen). The internal unit can be used with [Application.SetAttributeModelInternal\(\)](#). An exception is thrown for invalid attribute names or values.

SEE ALSO

[Application.DefineTransferAttributes\(\)](#), [DataObject.GetAttributes\(\)](#)

SetAttributeShape

Sets the shape of a matrix or vector attribute. The shape is a list of the form [number of rows, number of columns]. Number of columns has to be 0 for vectors.

```
int DataObject.SetAttributeShape(str name, list(int, int) shape)
```

ARGUMENTS

name Name of an attribute.

shape New shape of the attribute.

RETURNS

0 On success.

1 On error.

SEE ALSO

[DataObject.SetAttributeLength\(\)](#), [DataObject.GetAttributeShape\(\)](#), [DataObject.GetAttributeType\(\)](#)

ShowEditDialog

This function is not supported in GUI-less mode.

Opens the edit dialogue of the object. Command objects (such as *ComLdf*) will have their “Execute” button disabled. The execution of the running script will be halted until the edit dialogue is closed again.

Editing of command objects (*ComDPL*, *ComPython*) is not supported.

```
int DataObject.ShowEditDialog()
```

RETURNS

1 Edit dialogue was cancelled by the user.

0 Otherwise.

ShowModalSelectTree

This function is not supported in GUI-less mode.

Shows a modal window with the database object tree of the object on which the function is called on.

```
DataObject DataObject.ShowModalSelectTree([str title,]  
                                           [str filter])
```


ARGUMENTS

title (optional)

Title of the dialog. If omitted, a default title will be used.

filter (optional)

Classname filter e.g. 'ElmLne' or 'Com*'. If set, a selection is only accepted if the classname of the selected object matches that filter.

RETURNS

DataObject Selected object.**None** No object selcted e.g. 'Cancel' clicked.**SwitchOff**

Performs a “switch off” action on the network element. This action is identical to that in the context menu.

```
[int error,
list changedSwitches] DataObject.SwitchOff([int resetRA,]
                                           [int simulateOnly])
```

ARGUMENTS

changedSwitches (optional, out)

All switches whose switching state was changed by the action are filled into this set

resetRA (optional)

Determines whether an active running arrangement that would prevent switching action should be deactivated or not.

1 All running arrangements that cause blocking of relevant switches are applied and reset automatically before the switching is performed.

0 (default) Active running arrangements are not reset. Blocked switches will cause the switching action to fail

simulateOnly (optional)

Can be used to get the switches that would be changed. No switching is performed if set to '1'. (default is '0')

RETURNS

Information about the success of the action:

0 Action was successful**1** Action failed

SEE ALSO

[DataObject.SwitchOn\(\)](#), [DataObject.Isolate\(\)](#), [DataObject.Energize\(\)](#)**SwitchOn**

Performs a “switch on” action on the network element. This action is identical to that in the context menu.

```
[int error,
list changedSwitches] DataObject.SwitchOn([int resetRA,]
                                           [int simulateOnly])
```

ARGUMENTS*changedSwitches (optional, out)*

All switches whose switching state was changed by the action are filled into this set

resetRA (optional)

Determines whether an active running arrangement that would prevent switching action should be deactivated or not.

- 1** All running arrangements that cause blocking of relevant switches are applied and reset automatically before the switching is performed.
- 0** (default) Active running arrangements are not reset. Blocked switches will cause the switching action to fail

simulateOnly (optional)

Can be used to get the switches that would be changed. No switching is performed if set to '1'. (default is '0')

RETURNS

Information about the success of the action:

- 0** Action was successful
- 1** Action failed

SEE ALSO

[DataObject.SwitchOff\(\)](#), [DataObject.Isolate\(\)](#), [DataObject.Energize\(\)](#)

WriteChangesToDb

See [Application.WriteChangesToDb\(\)](#) for a detailed description.

```
None DataObject.WriteChangesToDb()
```

5.2 Network Elements

5.2.1 ElmArea

Overview

[CalculateInterchangeTo](#)
[DefineBoundary](#)
[GetAll](#)
[GetBranches](#)
[GetBuses](#)
[GetObjs](#)

CalculateInterchangeTo

Calculates interchange power to the given area (calculated quantities are: Pinter, Qinter, Pexport, Qexport, Pimport, Qimport). Prior the calculation the valid load flow calculation is required.

```
int ElmArea.CalculateInterchangeTo(DataObject area)
```

ARGUMENTS

area Area to which the interchange is calculated

RETURNS

< 0 Calculation error (i.e. no valid load flow, empty area...)
 0 No interchange power to the given area
 1 Interchange power calculated

DefineBoundary

Defines boundary with this area as interior part. Resulting cubicles of boundary are busbar-oriented towards the area.

```
DataObject ElmArea.DefineBoundary(int shift)
```

ARGUMENTS

shift Elements outside the area that are within a distance of shift many elements to a boundary cubicle of the area are added to the interior part of the resulting boundary

RETURNS

The defined boundary is returned in case of success. Otherwise NULL, if an error appeared in the definition of the boundary.

GetAll

Returns all objects which belong to this area.

```
list ElmArea.GetAll()
```

RETURNS

The set of contained objects.

GetBranches

Returns all branches which belong to this area.

```
list ElmArea.GetBranches()
```

RETURNS

The set of branch objects.

GetBuses

Returns all buses which belong to this area.

```
list ElmArea.GetBuses()
```

RETURNS

The set of objects.

GetObjs

Returns all objects of the given class which belong to this area.

```
list ElmArea.GetObjs(str classname)
```

ARGUMENTS

classname

Name of the class (i.e. "ElmTr2").

RETURNS

The set of objects.

5.2.2 ElmAsm

Overview

[CalcEfficiency](#)
[GetAvailableGenPower](#)
[GetElecTorque](#)
[GetGroundingImpedance](#)
[GetMechTorque](#)
[GetMotorStartingFlag](#)
[GetStepupTransformer](#)
[IsPQ](#)

CalcEfficiency

Calculate efficiency for connected storage model at given active power value.

```
float ElmAsm.CalcEfficiency(float activePowerMW)
```

ARGUMENTS

activePowerMW

Active power value to calculate efficiency for.

RETURNS

Returns the resulting efficiency in p.u.

GetAvailableGenPower

Returns the available power that can be dispatched from the generator, for the particular study time.

For the case of conventional generators (no wind generation selected), the available power is equal to the nominal power specified.

For wind generators, the available power will depend on the wind model specified:

- **No Wind Model:** No available power.
- **Stochastic Wind Model:** Given the specified mean wind speed, the available power is calculated from the Power Curve. If the units of the Power Curve are in MW, the returned value is directly the available power. In the other hand, if the units are in PU, the returned value is multiplied by the nominal power of the generator to return the available power.
- **Time Series Characteristics of Active Power Contribution:** The available power is the average of the power values (in MW) obtained from all the specified time characteristics for the current study time.
- **Time Series Characteristics of Wind Speed:** The available power is calculated with the average of the power values (in MW) calculated for all the specified time characteristics. A power value for any time characteristic is calculated by obtaining the wind speed for the current study time, and then calculating the power from the specified Power Curve. If the units of the Power Curve are in MW, the returned value is directly the power value. In the other hand, if the units are in PU, the returned value is multiplied by the nominal power of the generator to return the power value.

For motors, the available power is zero.

```
float ElmAsm.GetAvailableGenPower()
```

RETURNS

Available generation power

GetElecTorque

Calculates the electrical torque for a given speed and voltage.

```
float ElmAsm.GetElecTorque(float speed,
                           float uReal,
                           [float addZReal,]
                           [float addZImag]
                           )
```

ARGUMENTS

- speed* speed value in p.u.
- uReal* voltage value (real part) in p.u.
- addZReal (optional)*
 additional impedance (real part) in p.u.
- addZImag (optional)*
 additional impedance (imaginary part) in p.u.

RETURNS

Returns the calculated electrical torque.

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,  
float resistance,  
float reactance ] ElmAsm.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.

resistance (out)
Real part of the grounding impedance in Ohm.

reactance (out)
Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

GetMechTorque

Calculates the electrical torque for a given speed and voltage.

```
float ElmAsm.GetMechTorque(float speed,  
                           float uReal  
                           )
```

ARGUMENTS

speed speed value in p.u.

uReal voltage value (real part) in p.u.

RETURNS

Returns the calculated mechanical torque.

GetMotorStartingFlag

Returns the starting motor condition.

```
int ElmAsm.GetMotorStartingFlag()
```

RETURNS

Returns the motor starting condition. Possible values are:

- 1** in the process of being calculated
- 0** not calculated
- 1** successful start
- 2** unsuccessful start

GetStepupTransformer

Performs a topological search to find the step-up transformer of the asynchronous machine.

```
DataObject ElmAsm.GetStepupTransformer(float Voltage,
                                       int swStatus
                                       )
```

ARGUMENTS

hvVoltage

voltage level at which the search will stop

ignSwStatus

consideration of switch status. Possible values are:

- | | |
|----------|----------------------------|
| 0 | consider all switch status |
| 1 | ignore breaker status |
| 2 | ignore all switch status |

RETURNS

Returns the first collected step-up transformer object. It is empty if not found (e.g. start terminal already at hvVoltage).

IsPQ

Informs whether or not it is a "PQ" machine (constant Q control mode).

```
int ElmAsm.IsPQ()
```

RETURNS

Returns 1 if it is a "PQ" machine.

5.2.3 ElmAsmsc

Overview

[GetAvailableGenPower](#)
[GetGroundingImpedance](#)
[GetStepupTransformer](#)

GetAvailableGenPower

Returns the available power that can be dispatched from the generator, for the particular study time.

For the case of conventional generators (no wind generation selected), the available power is equal to the nominal power specified.

For wind generators, the available power will depend on the wind model specified:

- No Wind Model: No available power.
- Stochastic Wind Model: Given the specified mean wind speed, the available power is calculated from the Power Curve. If the units of the Power Curve are in MW, the returned value is directly the available power. In the other hand, if the units are in PU, the returned value is multiplied by the nominal power of the generator to return the available power.

- Time Series Characteristics of Active Power Contribution: The available power is the average of the power values (in MW) obtained from all the specified time characteristics for the current study time.
- Time Series Characteristics of Wind Speed: The available power is calculated with the average of the power values (in MW) calculated for all the specified time characteristics. A power value for any time characteristic is calculated by obtaining the wind speed for the current study time, and then calculating the power from the specified Power Curve. If the units of the Power Curve are in MW, the returned value is directly the power value. In the other hand, if the units are in PU, the returned value is multiplied by the nominal power of the generator to return the power value.

For motors, the available power is zero.

```
float ElmAsmSc.GetAvailableGenPower()
```

RETURNS

Available generation power

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,
float resistance,
float reactance ] ElmAsmSc.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.

resistance (out)
Real part of the grounding impedance in Ohm.

reactance (out)
Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

GetStepupTransformer

Performs a topological search to find the step-up transformer of the asynchronous machine.

```
DataObject ElmAsmSc.GetStepupTransformer(float Voltage,
                                           int swStatus
                                           )
```

ARGUMENTS

hvVoltage
voltage level at which the search will stop

ignSwStatus
consideration of switch status. Possible values are:

- | | |
|----------|----------------------------|
| 0 | consider all switch status |
| 1 | ignore breaker status |
| 2 | ignore all switch status |

RETURNS

Returns the first collected step-up transformer object. It is empty if not found (e.g. start terminal already at hvVoltage).

5.2.4 ElmBbone

Overview

[CheckBbPath](#)
[GetBbOrder](#)
[GetCompleteBbPath](#)
[GetFOR](#)
[GetMeanCs](#)
[GetMinCs](#)
[GetTieOpenPoint](#)
[GetTotLength](#)
[HasGnrlMod](#)

CheckBbPath

Check whether the backbone object is still valid. This means:

- a** Terminals determining backbone path are still directly connected.
- b** One switch is open on the path of an inter-feeder backbone.
- c** Contents of backbone match specified starting-feeder (and end feeder).
- d** Start and end of feeder are calculation-relevant.
- e** Path is unique via the defined terminals (no parallel elements (only warning!)).

```
int ElmBbone.CheckBbPath(int outputMsg)
```

ARGUMENTS

outputMsg

- | | |
|----------|--|
| 1 | Output resulting messages of check function. |
| 0 | Only check, no output of messages. |

RETURNS

- | | |
|----------|---|
| 0 | Backbone is valid. |
| 1 | Backbone is invalid because of one or more of the above listed reasons. |

GetBbOrder

Get order of backbone object, determined by backbone calculation according to the selected criterion.

```
int ElmBbone.GetBbOrder()
```

RETURNS

The order of the backbone object. The smaller the returned value, the better the backbone according to chosen criterion. The order 1 is returned for the best backbone.

GetCompleteBbPath

Get the complete (ordered) path containing all terminals and connecting elements of the backbone.

```
list ElmBbone.GetCompleteBbPath(int iReverse,
                                [int iStopAtTieOpen = 0])
```

ARGUMENTS

AllElmsOnBb (out)

Ordered path containing all terminals and connecting elements of the backbone.

iReverse

- | | |
|----------|---|
| 0 | Return ordered path from start feeder to end feeder |
| 1 | Return ordered path from end feeder to start feeder |

iStopAtTieOpen

- | | |
|----------|--|
| 0 | return complete path |
| 1 | only return part of path in start feeder (iReverse=0) / in end feeder (iReverse=1) |

GetFOR

Get aggregated forced outage rate (FOR) of all elements on the path of the backbone.

```
float ElmBbone.GetFOR()
```

RETURNS

The aggregated forced outage rate (FOR) of all elements on the path of the backbone [in 1/a].

GetMeanCs

Get mean cross section value of all elements on the path of the backbone. Every cross section value is weighted with the relative length corresponding to the total length of the backbone.

```
float ElmBbone.GetMeanCs()
```

RETURNS

The mean cross section of the elements on the backbone path [in mm²].

GetMinCs

Get minimum cross section value of all elements on the path of the backbone. Optional: a set with all elements on the backbone path featuring this cross section may be returned.

```
[float minCs,
set ElmsMinCs] ElmBbone.ElmBoundary.IsSplitting()
```

ARGUMENTS

ElmsMinCs

Elements on the backbone path featuring minimum cross section value.

RETURNS

The minimum cross section of all elements on the backbone path [in mm²].

GetTieOpenPoint

Search and obtain the first open switching device (ElmCoup, StaSwitch) on the backbone path (starting from the infeeding point of the starting feeder).

```
DataObject ElmBbone.GetTieOpenPoint()
```

RETURNS

The switching device (ElmCoup or StaSwitch) or None if backbone is invalid.

GetTotLength

Get total length of all elements on the path of the backbone.

```
float ElmBbone.GetTotLength()
```

RETURNS

The total length of the backbone path [in km].

HasGnrlMod

Check whether backbone object ElmBbone has a valid CalBbone where corresponding results are stored. This is only the case after a backbone calculation by scoring method (until the calculation is reset).

```
int ElmBbone.HasGnrlMod()
```

RETURNS

- 1** ElmBbone has a calculation model,
- 0** no calculation model available.

5.2.5 ElmBmu**Overview**

[Apply](#)
[Update](#)

Apply

Applies the power dispatch. Depending on the selected 'Distribution Mode ' this is done by a built-in algorithm based on 'Merit Order' or by a user-defined DPL script that is stored in the contents of the virtual power plant object.

```
int ElmBmu.Apply()
```

RETURNS

- 0** on success, no error occurred
- 1** error during dispatch by virtual power plant. Please note, a value of 1 is also returned in case the power plant is current set out-of-service.

Update

Updates the list of machines in the tables: 'Dispatchable Machines' and 'Non-dispatchable (fixed) Machines'.

```
None ElmBmu.Update()
```

5.2.6 ElmBoundary

Overview

[AddCubicle](#)
[CalcShiftedReversedBoundary](#)
[Clear](#)
[GetInterior](#)
[IsSplitting](#)
[Resize](#)
[Update](#)

AddCubicle

Adds a given cubicle with given orientation to an existing boundary. The cubicle is added only if it is not already contained within the boundary.

```
int ElmBoundary.AddCubicle(DataObject cubicle,
                           int orientation
                           )
```

RETURNS

- 0** cubicle was successfully added
- 1** cubicle was not added because it is already contained (including given orientation)

CalcShiftedReversedBoundary

Defines boundary where exterior and interior part of this boundary are exchanged. Resulting boundary cubicles are branch-oriented.

```
[int error,
DataObject boundary] ElmBoundary.CalcShiftedReversedBoundary(float shift)
```

ARGUMENTS

- shift* Elements that are within a distance of shift many elements to a boundary cubicle of this boundary are added to the exterior part of the resulting boundary.

boundary (out)

Defined boundary.

RETURNS

- 0** Successful call, boundary defined.
- 1** Error during determination of boundary cubicles.

Clear

Removes all boundary cubicles from an existing boundary.

```
None ElmBoundary.Clear()
```

GetInterior

Returns a set of all elements that are contained in the interior region of the boundary.

```
list ElmBoundary.GetInterior()
```

RETURNS

Returns the set of interior elements.

IsSplitting

Checks if the boundary splits the network into two regions. A boundary is called splitting, if and only if, for each boundary cubicle, the adjacent terminal and the adjacent branch component belong to different sides of the boundary.

```
[int isSplitting,  
list notSplittingCubicles] ElmBoundary.IsSplitting()
```

ARGUMENTS

notSplittingCubicles (optional, out)

All cubicles that prevent the boundary from being splitting are filled into this set.

RETURNS

- 0** not splitting boundary
- 1** splitting boundary

Resize

Resizes the boundary cubicle vector or the cubicle orientation vector. It is strongly advised that the size of both vectors must be the same.

```
None ElmBoundary.Resize(float size,  
                        str name  
                        )
```

ARGUMENTS

- size* size of the referenced vector (number of cubicles)
- name* reference to the vector ('iorient' or 'cubicles')

RETURNS

If the resize is unsuccessful the error message shall be issued.

Update

Updates cached information (such as topological interior). Required when boundary definition was changed via DPL or Python.

```
None ElmBoundary.Update()
```

5.2.7 ElmBranch**Overview**

[Update](#)

Update

Updates connection points and contained elements of the branch. If the branch element externally modified by the user, then the update shall refresh all connections in the correct manner. Behaves same as the update button within the ElmBranch.

```
None ElmBranch.Update()
```

5.2.8 ElmCabsys**Overview**

[FitParams](#)
[GetLineCable](#)
[Update](#)

FitParams

Calculates distributed parameters for cable system elements. Whether this function calculates constant parameters or frequency dependent parameters depends on the user setting of the parameter 'i_model' in the ElmCabsys dialog. The settings are as follows: i_model=0: constant parameters; i_model=1: frequency dependent parameters.

```
int ElmCabsys.FitParams()
```

RETURNS

0	on success
1	on error

GetLineCable

Gets cable type for the corresponding line, within the cable system.

```
DataObject ElmCabsys.GetLineCable(int line)
```

ARGUMENTS

line Index of line.

RETURNS

cable type On success.

None On error.

Update

Updates cable system element depending on configuration of the associated cable system type.

```
int ElmCabsys.Update()
```

RETURNS

1 On success.

0 On error.

5.2.9 ElmComp**Overview**

[SlotUpdate](#)

SlotUpdate

Performs a slot update for the composite model, to try to reassign each model found in the composite model contents to the corresponding slot.

```
None ElmComp.SlotUpdate()
```

DEPRECATED NAMES

Slotupd

5.2.10 ElmCoup**Overview**

[Close](#)
[GetRemoteBreakers](#)
[IsBreaker](#)
[IsClosed](#)
[IsOpen](#)
[Open](#)

Close

Closes the switch by changing its status to 'close'. This action will fail if the status is currently determined by an active running arrangement.

```
int ElmCoup.Close()
```

RETURNS

- 0** On success
- ≠ 0** On error

SEE ALSO

[ElmCoup.Open\(\)](#)

GetRemoteBreakers

Returns the remote circuit breakers or connected bus bars.

This information is determined by a topological search that starts at given breaker in all directions, generally stopping at

- switches of type circuit breaker
- switches that are open
- busbars (ElmTerm::iUsage == 0)

If the search reaches a busbar while only reducible components have been passed (see [DataObject.IsReducible\(\)](#)) it stops and returns the connected busbar (no breaker found). In case of non-reducible components have been passed, the search continues until next circuit breaker is found. Only breakers with given breaker state are returned.

```
[list remoteBreakers,
list foundBreakers,
list foundBusbars ] ElmCoup.GetRemoteBreakers(int desiredBreakerState)
```

ARGUMENTS

desiredBreakerState

Only breakers with given status are collected.

- 1** Return all remote circuit breakers
- 1** Return all closed remotod circuit breakers
- 0** Return all opened remotod circuit breakers

foundBreakers (out)

The list of the remote circuit breakers

foundBusbars (optional, out)

The list of the local bus bars

IsBreaker

Checks if type of current switch is 'circuit-breaker'.

```
int ElmCoup.IsBreaker()
```

RETURNS

- 1** Switch is a circuit-breaker.
- 0** Switch is not a circuit-breaker.

IsClosed

Returns information about current switch state.

```
int ElmCoup.IsClosed()
```

RETURNS

- | | |
|----------|------------------|
| 1 | switch is closed |
| 0 | switch is open |

SEE ALSO

[ElmCoup.IsOpen\(\)](#)

IsOpen

Returns information about current switch state.

```
int ElmCoup.IsOpen()
```

RETURNS

- | | |
|----------|------------------|
| 1 | switch is open |
| 0 | switch is closed |

SEE ALSO

[ElmCoup.IsClosed\(\)](#)

Open

Opens the switch by changing its status to 'open'. This action will fail if the status is currently determined by an active running arrangement.

```
int ElmCoup.Open()
```

RETURNS

- | | |
|------------|------------|
| 0 | On success |
| ≠ 0 | On error |

SEE ALSO

[ElmCoup.Close\(\)](#)

5.2.11 ElmDsl

Overview

[ExportToClipboard](#)
[ExportToFile](#)

ExportToClipboard

Export the parameter list to clipboard.

```
None ElmDsl.ExportToClipboard([str colSeparator],
```

```
[int useLocalHeader]
)
```

ARGUMENTS

colSeparator (optional)

Separator between the columns (default: tab character).

useLocalHeader (optional)

Use the localised version of the header. Possible values are:

- 1** Yes (default).
- 0** No (use English language header).

ExportToFile

Export the parameter list to CSV file(s).

```
None ElmDsl.ExportToFile(str filePath,
                          [str colSeparator],
                          [int useLocalHeader]
                          )
```

ARGUMENTS

filePath Path of the CSV target file. In case of array and matrix parameters (names: “array_NAME” and “matrix_NAME”), additional CSV files are created in the same location with names obtained by appending “_array_NAME” and “_matrix_NAME” to the target file name.

colSeparator (optional)

Separator between the columns (default: “;”).

useLocalHeader (optional)

Use the localised version of the header. Possible values are:

- 1** Yes (default).
- 0** No (use English language header).

5.2.12 ElmFeeder

Overview

[CalcAggrVarsInRadFeed](#)

[GetAll](#)

[GetBranches](#)

[GetBuses](#)

[GetNodesBranches](#)

[GetObjs](#)

CalcAggrVarsInRadFeed

Computes all the aggregated variables in radial feeders.

```
int ElmFeeder.CalcAggrVarsInRadFeed([int lookForRoot,]
                                     [int considerNested])
```

ARGUMENTS

lookForRoot (optional)

Calculates the variables from the deepest root. Possible values are:

- 0** Start from this feeder
- 1** (default) Find the deepest root.

considerNested (optional)

Calculates the variables also for any nested subfeeders. Possible values are:

- 0** Ignore any nested feeders
- 1** (default) Consider nested feeders.

RETURNS

Returns whether or not the aggregated variables were calculated. Possible values are:

- 0** error during calculation
- 1** calculated correctly

GetAll

Returns a set with all objects belonging to this feeder.

```
list ElmFeeder.GetAll([int iNested])
```

ARGUMENTS

iNested (optional)

Affects the collection of objects in case of nested feeders:

- 0** Only the objects of this feeder will be returned.
- 1** (default) All elements including those of nested feeders will be returned.

RETURNS

The set of network elements belonging to this feeder. Can be empty.

SEE ALSO

[DataObject.IsInFeeder\(\)](#)**GetBranches**

Returns a set with all branch elements belonging to this feeder.

```
list ElmFeeder.GetBranches([int iNested])
```

ARGUMENTS

iNested (optional)

Affects the collection of objects in case of nested feeders:

- 0** Only the objects of this feeder will be returned.
- 1** (default) All elements including those of nested feeders will be returned.

RETURNS

The set of bus and branch elements in feeder.

GetBuses

Returns a set with all buses belonging to this feeder.

```
list ElmFeeder.GetBuses([int iNested])
```

ARGUMENTS

iNested (optional)

Affects the collection of objects in case of nested feeders:

- | | |
|----------|--|
| 0 | Only the objects of this feeder will be returned. |
| 1 | (default) All elements including those of nested feeders will be returned. |

RETURNS

The set of bus elements in feeder.

GetNodesBranches

Returns a set with all buses and branches belonging to this feeder.

```
list ElmFeeder.GetNodesBranches([int iNested])
```

ARGUMENTS

iNested (optional)

Affects the collection of objects in case of nested feeders:

- | | |
|----------|--|
| 0 | Only the objects of this feeder will be returned. |
| 1 | (default) All elements including those of nested feeders will be returned. |

RETURNS

The set of bus and branch elements in feeder.

GetObjs

Returns a set with all objects of class 'ClassName' which belong to this feeder.

```
list ElmFeeder.GetObjs(str ClassName,  
                        [int iNested])
```

ARGUMENTS

iNested (optional)

Affects the collection of objects in case of nested feeders:

- | | |
|----------|--|
| 0 | Only the objects of this feeder will be returned. |
| 1 | (default) All elements including those of nested feeders will be returned. |

RETURNS

The set of feeder objects.

5.2.13 ElmFile

Overview

[LoadFile](#)

[SaveFile](#)

LoadFile

(Re)Loads the file into a buffer.

```
int ElmFile.LoadFile([int loadComplete = 1])
```

ARGUMENTS

loadComplete (optional)

- | | |
|----------|--|
| 0 | Removes all points in the future simulation time and adds all points from the file (including the current interpolated value). |
| 1 | Clears the buffer and reloads the complete file (default). |

RETURNS

- | | |
|------------|-------------|
| 0 | On success. |
| ≠ 0 | On error. |

SaveFile

Saves the buffer and overwrites the file.

```
int ElmFile.SaveFile()
```

RETURNS

- | | |
|------------|-------------|
| 0 | On success. |
| ≠ 0 | On error. |

5.2.14 ElmFilter

Overview

[GetGroundingImpedance](#)

GetGroundingImpedance

Returns the impedance of the internal grounding. Single phase filters connected to neutral are considered as grounding devices themselves; i.e. instead of the dedicated grounding parameters, the filters parameters are used.

```
[int valid,  
float resistance,
```

```
float reactance ] ElmFilter.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.

resistance (out)
Real part of the grounding impedance in Ohm.

reactance (out)
Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

5.2.15 ElmGenstat

Overview

[CalcEfficiency](#)
[Derate](#)
[Disconnect](#)
[GetAvailableGenPower](#)
[GetGroundingImpedance](#)
[GetStepupTransformer](#)
[IsConnected](#)
[Reconnect](#)
[ResetDerating](#)

CalcEfficiency

Calculate efficiency for connected storage model at given active power value.

```
float ElmGenstat.CalcEfficiency(float activePowerMW)
```

ARGUMENTS

activePowerMW
Active power value to calculate efficiency for.

RETURNS

Returns the resulting efficiency in p.u.

Derate

Derates the value of the Max. Active Power Rating according to the specified value given in MW.

The following formula is used: $P_{max_uc} = P_{max_uc} - "Deratingvalue"$.

```
None ElmGenstat.Derate(float deratingP)
```

ARGUMENTS

deratingP Derating value

Disconnect

Disconnects a static generator by opening the first circuit breaker. The topological search performed to find such a breaker, stops at any busbar.

```
int ElmGenstat.Disconnect()
```

RETURNS

- 0** breaker already open or successfully opened
- 1** an error occurred (no breaker found, open action not possible (earthing / RA))

GetAvailableGenPower

Returns the available power that can be dispatched from the generator, for the particular study time.

For the case of conventional generators (no wind generation selected), the available power is equal to the nominal power specified.

For wind generators, the available power will depend on the wind model specified:

- No Wind Model: No available power.
- Stochastic Wind Model: Given the specified mean wind speed, the available power is calculated from the Power Curve. If the units of the Power Curve are in MW, the returned value is directly the available power. In the other hand, if the units are in PU, the returned value is multiplied by the nominal power of the generator to return the available power.
- Time Series Characteristics of Active Power Contribution: The available power is the average of the power values (in MW) obtained from all the specified time characteristics for the current study time.
- Time Series Characteristics of Wind Speed: The available power is calculated with the average of the power values (in MW) calculated for all the specified time characteristics. A power value for any time characteristic is calculated by obtaining the wind speed for the current study time, and then calculating the power from the specified Power Curve. If the units of the Power Curve are in MW, the returned value is directly the power value. In the other hand, if the units are in PU, the returned value is multiplied by the nominal power of the generator to return the power value.

For motors, the available power is zero.

```
float ElmGenstat.GetAvailableGenPower()
```

RETURNS

Available generation power

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,
float resistance,
float reactance ] ElmGenstat.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.

resistance (out)
Real part of the grounding impedance in Ohm.

reactance (out)
Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

GetStepupTransformer

Performs a topological search to find the step-up transformer of the static generator.

```
DataObject ElmGenstat.GetStepupTransformer(float voltage,
                                             int swStatus
                                             )
```

ARGUMENTS

voltage voltage level at which the search will stop

swStatus consideration of switch status. Possible values are:

- 0** consider all switch status
- 1** ignore breaker status
- 2** ignore all switch status

RETURNS

Returns the first collected step-up transformer object. It is empty if not found (e.g. start terminal already at hvVoltage).

IsConnected

Checks if generator is topologically connected to any busbar.

```
int ElmGenstat.IsConnected()
```

RETURNS

- 0** false, not connected to a busbar
- 1** true, generator is connected to a busbar

Reconnect

Connects a static generator by closing all switches (breakers and isolators) up to the first breaker on the HV side of a transformer. The topological search to find all the switches, stops at any busbar.

```
int ElmGenstat.Reconnect()
```


RETURNS

- 0** the machine was successfully closed
- 1** a error occurred and the machine could not be connected to any busbar

ResetDerating

Resets the derating value, setting the Max. Active Power Rating according to the rating factor. The following formula is used: $P_{max_uc} = p_{maxratf} * P_n * n_{gnum}$.

```
None ElmGenstat.ResetDerating()
```

5.2.16 ElmGndswt**Overview**

[Close](#)
[GetGroundingImpedance](#)
[IsClosed](#)
[IsOpen](#)
[Open](#)

Close

Closes the switch by changing its status to 'close'. If closed, the connected node will be considered as being earthed.

```
int ElmGndswt.Close()
```

RETURNS

- 1, always

SEE ALSO

[ElmGndswt.Open\(\)](#)

GetGroundingImpedance

Returns the impedance of the internal grounding. ElmGndswt is only considered to have an internal grounding if it is single phase and connected to neutral.

```
[int valid,  
float resistance,  
float reactance ] ElmGndswt.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

- busIdx* Bus index where the grounding should be determined.
- resistance (out)*
Real part of the grounding impedance in Ohm.
- reactance (out)*
Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

IsClosed

Returns information about current switch state.

```
int ElmGndswt.IsClosed()
```

RETURNS

- 1** switch is closed
- 0** switch is open

SEE ALSO

[ElmGndswt.IsOpen\(\)](#)

IsOpen

Returns information about current switch state.

```
int ElmGndswt.IsOpen()
```

RETURNS

- 1** switch is open
- 0** switch is closed

SEE ALSO

[ElmGndswt.IsClosed\(\)](#)

Open

Opens the switch by changing its status to 'open'.

```
int ElmGndswt.Open()
```

RETURNS

- 0**, always

SEE ALSO

[ElmGndswt.Close\(\)](#)

5.2.17 ElmLne

Overview

[AreDistParamsPossible](#)
[CreateFeederWithRoutes](#)
[FitParams](#)
[GetIthr](#)
[GetType](#)
[GetY0m](#)
[GetY1m](#)
[GetZ0m](#)
[GetZ1m](#)
[GetZmatDist](#)
[HasRoutes](#)
[HasRoutesOrSec](#)
[IsCable](#)
[IsNetCoupling](#)
[MeasureLength](#)
[SetDetailed](#)

AreDistParamsPossible

Check if the line fulfils conditions for the calculation of distributed parameters:

ElmLne No routes, no sections

TypTow only 1 circuit x 3 phases

TypGeo only 1 circuit x 3 phases

TypLne AC system, 3 phases and 0 neutral

TypCabsys only 1 circuit x 3 phases

```
int ElmLne.AreDistParamsPossible()
```

RETURNS

The returned value are:

- 0** All conditions fulfilled
- 1** Line contains routes
- 2** Line contains sections
- 3** Line has no type
- 4** TypTow/TypCabsys does not fulfil conditions for distributed paramters
- 5** TypLne does not fulfil conditions for distributed parameters
- 6** Short-circuit flag is set (EMT or RMS simulations)
- 7** TypLne/TypTow: B0 and B1 = 0
- 8** Error, no condition state could be determined

CreateFeederWithRoutes

Creates a new feeder in the line by splitting the line into 2 routes and inserting a terminal.

```
int ElmLne.CreateFeederWithRoutes(float dis,
                                  float rem,
                                  DataObject O)
int ElmLne.CreateFeederWithRoutes(float dis,
                                  float rem,
                                  DataObject O,
                                  [int sw0,]
                                  [int sw1])
```

ARGUMENTS

<i>dis</i>	Inserting operation occurs after this distance
<i>rem</i>	Remaining distance, percentage of distance 'dis'
<i>O</i>	Branch object that is to be connected at the inserted terminal
<i>sw0</i>	If set to (1), switch is inserted on the first side
<i>sw1</i>	If set to (1), switch is inserted on the second side

RETURNS

0	Success, feeders created
1	Error

FitParams

Calculates distributed parameters of the line element. Whether this function calculates constant parameters or frequency dependent parameters depends on the user setting of the parameter 'i_model' in the ElmLne dialogue. The settings are as follows: i_model=0: constant parameters; i_model=1: frequency dependent parameters.

```
int ElmLne.FitParams([int isRMSModel = 0,]
                    [int modify = 1])
```

ARGUMENTS

isRMSModel

modify

RETURNS

0	Success
1	Error

GetIthr

Returns the rated short-time current of the line element.

```
float ElmLne.GetIthr()
```

RETURNS

Returns rated short-time current value

GetType

Returns the line type object.

```
DataObject ElmLne.GetType()
```

RETURNS

The TypLne object if exists or None

GetY0m

The function returns the zero-sequence mutual coupling admittance (G0m, B0m) in Ohm of the line and input argument line (object Lne2). When Lne2 = line, the function returns the zero-sequence self admittance.

```
[int error,
float G0m,
float B0m ] ElmLne.GetY0m(DataObject Lne2)
```

ARGUMENTS

Lne2 Line element

G0m (out)
Resulting G0m value

B0m (out)
Resulting B0m value

RETURNS

0 Success, data obtained
1 Error, e.g. no coupling objects defined

GetY1m

The function returns the positive-sequence mutual coupling admittance (G1m, B1m) in Ohm of the line and input argument line (object Lne2). When Lne2 = line, the function returns the positive-sequence self admittance.

```
[int error,
float G1m,
float B1m ] ElmLne.GetY1m (DataObject Lne2)
```

ARGUMENTS

Lne2 Line element

G1m (out)
Resulting G1m value

B1m (out)
Resulting B1m value

RETURNS

- 0** Success, data obtained
- 1** Error, e.g. no coupling objects defined

GetZ0m

Gets the zero-sequence mutual coupling impedance (R_{0m} , X_{0m}) in Ohm of the line and input argument line (object *otherLine*). When *otherLine* = line, the function returns the zero-sequence self impedance.

```
[int error,
float R0m,
float X0m ] ElmLne.GetZ0m(DataObject otherLine)
```

ARGUMENTS

otherLine Line element

R0m (out)

To be obtained R_{0m} value

X0m (out)

To be obtained X_{0m} value

RETURNS

- 0** Success, data obtained
- 1** Error, e.g. no coupling objects defined

GetZ1m

The function returns the positive-sequence mutual coupling impedance (R_{1m} , X_{1m}) in Ohm of the line and input argument line (object *Lne2*). When *Lne2* = line, the function returns the positive-sequence self impedance.

```
[int error,
float R1m,
float X1m ] ElmLne.GetZ1m(DataObject Lne2)
```

ARGUMENTS

Lne2 Line element

R1m (out)

Resulting R_{1m} value

X1m (out)

Resulting X_{1m} value

RETURNS

- 0** Success, data obtained
- 1** Error, e.g. no coupling objects defined

GetZmatDist

The function gets impedance matrix in phase domain (only amplitudes), for a line with distributed parameters, short-circuit ended.

```
int ElmLine.GetZmatDist(float frequency,
                        int exact,
                        DataObject matrix)
```

ARGUMENTS

- frequency* Frequency for which the calculation is carried out
- exact* 0: Approximated solution, 1: Exact solution for 'frequency'
- matrix* Impedance matrix to be filled with the impedance amplitudes

RETURNS

The returned value reports if the impedance matrix acquired:

- 1** Error, no matrix acquired
- 0** Success, matrix acquired

HasRoutes

Checks if the line is subdivided into routes.

```
int ElmLine.HasRoutes()
```

RETURNS

- 0** When the line is a single line
- 1** When the line is subdivided into routes

HasRoutesOrSec

Checks if the line is subdivided into routes or sections.

```
int ElmLine.HasRoutesOrSec()
```

RETURNS

- 0** When the line is a single line
- 1** When the line is subdivided into routes
- 2** When the line is subdivided into sections

IsCable

Checks if this line is a cable.

```
int ElmLine.IsCable()
```

RETURNS

- 1** Line is a cable
- 0** Line is not a cable

IsNetCoupling

Checks if the line connects two grids.

```
int ElmLne.IsNetCoupling()
```

RETURNS

The returned value reports if the line is a coupler:

- 1** The line is a coupler (connects two grids)
- 0** The line is not a coupler

MeasureLength

Measures the length of this line using the active diagram. For graphical measurement the active diagram needs to have a scaling factor. Geographic diagrams by default have a scaling factor. If `iUseGraphic = 1`, the line length is determined directly from the positions given in (latitude/longitude) considering the earth as a perfect sphere. In this case no graphic needs to be open.

```
float ElmLne.MeasureLength([int iUseGraphic])
```

ARGUMENTS

iUseGraphic (optional)

Use SGL diagram for calculation or not.

- 1** Use displayed diagram for calculation (default)
- 0** Calculate distance without diagram

RETURNS

- ≥ 0 Returns the graphical length of this line in its current unit
- < 0 Error: E.g. when line is not represented in the active diagram and `iUseGraphic=1`

SetDetailed

The function can be used to prevent the automatically reduction of a line e.g. if the line is a line dropper (length = 0). The function should be called when no calculation method is valid (before first load flow). The internal flag is automatically reset after the first calculation is executed.

```
int ElmLne.SetDetailed()
```

5.2.18 ElmLnesec**Overview**

[IsCable](#)

IsCable

Checks if this line section is a cable.

```
int ElmLnsec.IsCable()
```

RETURNS

- 1** Line section is a cable
- 0** Line section is not a cable
- 1** Error

5.2.19 ElmNec**Overview**

[GetGroundingImpedance](#)

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,
float resistance,
float reactance ] ElmNec.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

- busIdx* Bus index where the grounding should be determined.
- resistance (out)*
Real part of the grounding impedance in Ohm.
- reactance (out)*
Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

5.2.20 ElmNet**Overview**

[Activate](#)
[CalculateInterchangeTo](#)
[Deactivate](#)
[DefineBoundary](#)

Activate

Adds a grid to the active study case. Can only be applied if there are is no currently active calculation (i.e. running contingency analysis).

```
int ElmNet.Activate()
```

RETURNS

- 0** on success
- 1** on error

CalculateInterchangeTo

This function calculates the power flow from current grid to a connected grid. The values are stored in current grid in the following attributes (values from the previous load flow calculation are overwritten):

- Pinter: Active Power Flow
- Qinter: Reactive Power Flow
- ExportP: Export Active Power Flow
- ExportQ: Export Reactive Power Flow
- ImportP: Import Active Power Flow
- ImportQ: Import Reactive Power Flow

```
int ElmNet.CalculateInterchangeTo(DataObject net)
```

ARGUMENTS

- net* Connected grid

RETURNS

- < 0** error
- = 0** grids are not connected, no interchange exists
- > 0** ok

Deactivate

Removes a grid from the active study case. Can only be applied if there are is no currently active calculation.

```
int ElmNet.Deactivate()
```

RETURNS

- 0** on success
- 1** on error

DefineBoundary

Defines boundary with this grid as interior part. Resulting cubicles of boundary are busbar-oriented towards the grid.

```
DataObject ElmNet.DefineBoundary(int shift)
```

ARGUMENTS

shift Elements outside the grid that are within a distance of *shift* many elements to a boundary cubicle of the grid are added to the interior part of the resulting boundary

RETURNS

The defined boundary is returned in case of success. Otherwise NULL, if an error appeared in the definition of the boundary.

5.2.21 ElmPvsys

Overview

[CalcEfficiency](#)
[Derate](#)
[Disconnect](#)
[GetAvailableGenPower](#)
[GetGroundingImpedance](#)
[IsConnected](#)
[Reconnect](#)
[ResetDerating](#)

CalcEfficiency

Calculate efficiency for connected storage model at given active power value.

```
float ElmPvsys.CalcEfficiency(float activePowerMW)
```

ARGUMENTS

activePowerMW
Active power value to calculate efficiency for.

RETURNS

Returns the resulting efficiency in p.u.

Derate

Derates the value of the Max. Active Power Rating according to the specified value given in MW.

The following formula is used: $P_{max_uc} = P_{max_uc} - "Deratingvalue"$.

```
None ElmPvsys.Derate(float deratingP)
```

ARGUMENTS

deratingP Derating value

Disconnect

Disconnects a PV system by opening the first circuit breaker. The topological search performed to find such a breaker, stops at any busbar.

```
int ElmPvsys.Disconnect()
```

RETURNS

- 0** breaker already open or successfully opened
- 1** an error occurred (no breaker found, open action not possible (earthing / RA))

GetAvailableGenPower

Returns the available power that can be dispatched from the generator, for the particular study time.

For the case of conventional generators (no wind generation selected), the available power is equal to the nominal power specified.

For wind generators, the available power will depend on the wind model specified:

- No Wind Model: No available power.
- Stochastic Wind Model: Given the specified mean wind speed, the available power is calculated from the Power Curve. If the units of the Power Curve are in MW, the returned value is directly the available power. In the other hand, if the units are in PU, the returned value is multiplied by the nominal power of the generator to return the available power.
- Time Series Characteristics of Active Power Contribution: The available power is the average of the power values (in MW) obtained from all the specified time characteristics for the current study time.
- Time Series Characteristics of Wind Speed: The available power is calculated with the average of the power values (in MW) calculated for all the specified time characteristics. A power value for any time characteristic is calculated by obtaining the wind speed for the current study time, and then calculating the power from the specified Power Curve. If the units of the Power Curve are in MW, the returned value is directly the power value. In the other hand, if the units are in PU, the returned value is multiplied by the nominal power of the generator to return the power value.

For motors, the available power is zero.

```
float ElmPvsys.GetAvailableGenPower()
```

RETURNS

Available generation power

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,
float resistance,
float reactance ] ElmPvsys.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.

resistance (out)
Real part of the grounding impedance in Ohm.

reactance (out)
Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

IsConnected

Checks if a PV system is already connected to any busbar.

```
int ElmPvsys.IsConnected()
```

RETURNS

- 0** false, not connected to a busbar
- 1** true, generator is connected to a busbar

Reconnect

Connects a PV system by closing all switches (breakers and isolators) up to the first breaker on the HV side of a transformer. The topological search to find all the switches, stops at any busbar.

```
int ElmPvsys.Reconnect()
```

RETURNS

- 0** the machine was successfully closed
- 1** a error occurred and the machine could not be connected to any busbar

ResetDerating

Resets the derating value, setting the Max. Active Power Rating according to the rating factor. The following formula is used: $P_{max_uc} = p_{maxratf} * P_n * n_{gnum}$.

```
None ElmPvsys.ResetDerating()
```

5.2.22 ElmRelay**Overview**

[CheckRanges](#)
[GetCalcRX](#)
[GetMaxFdetectCalcI](#)
[GetSlot](#)
[GetUnom](#)
[IsStarted](#)
[SetImpedance](#)
[SetMaxI](#)
[SetMaxIearth](#)
[SetMinI](#)
[SetMinIearth](#)
[SetOutOfService](#)
[SetTime](#)
[SlotUpdate](#)

CheckRanges

Checks the settings of all elements in the relay for range violations.

```
int ElmRelay.CheckRanges()
```

RETURNS

- 0** All settings are valid.
- 1** At least one setting was forced into range.
- 1** An error occurred.

GetCalcRX

Gets the calculated impedance from the polarising unit.

```
[int error,
float real,
float imag ] ElmRelay.GetCalcRX(int inSec,
                                int unit)
```

ARGUMENTS

inSec

- 0** Get the value in pri. Ohm.
- 1** Get the value in sec. Ohm.

unit

- 0** Get the value from Phase-Phase or Multifunctional polarizing.
- 1** Get the value from Phase-Earth or Multifunctional polarizing.
- 2** Get the value from Multifunctional polarizing

real (out) Real part of the impedance in Ohm.

imag (out) Imaginary part of the impedance in Ohm.

RETURNS

- 0** No error occurred, the output is valid.
- 1** An error occurred, the output is invalid.

GetMaxFdetectCalcI

Get the current measured by the starting unit.

```
[int error,
float Iabs ] ElmRelay.GetMaxFdetectCalcI(int earth,
                                          int unit
                                          )
```

ARGUMENTS

Iabs (out) The measured current in A

earth

- 0** Get the phase current.

	1	Get the earth current.
<i>unit</i>	0	Get the current in pri. A.
	1	Get the current in sec. A.

RETURNS

0	No error, output is valid.
1	An error occurred, the output is invalid.

GetSlot

Returns the element in the slot with the given name.

```
DataObject ElmRelay.GetSlot(str name,
                             [int iShowErr]
                             )
```

ARGUMENTS

<i>name</i>	Exact name of the slot to search for (no wildcards).
<i>iShowErr (optional)</i>	
0	Do not show error messages.
1	Show error messages if a slot is not found or empty.

RETURNS

The object in the slot or None.

GetUnom

Returns the nominal voltage of the local bus of the relay.

```
float ElmRelay.GetUnom()
```

RETURNS

The nominal voltage of the local bus of the relay in kV.

IsStarted

Checks if the starting unit detected a fault.

```
int ElmRelay.IsStarted()
```

RETURNS

0	No fault was detected.
1	Fault was detected.
-1	An error occurred.

SetImpedance

Sets the the given impedance to the distance blocks matching the criteria.

```
int ElmRelay.SetImpedance(float real,
                          float imag,
                          int inSec,
                          int zone,
                          int unit
                          )
int ElmRelay.SetImpedance(float real,
                          float imag,
                          float lineAngle,
                          float Rarc,
                          int inSec,
                          int zone,
                          int unit
                          )
```

ARGUMENTS

<i>real</i>	Real part of the impedance in Ohm.
<i>imag</i>	Imaginary part of the impedance in Ohm.
<i>inSec</i>	
0	The values are in pri. Ohm.
1	The values are in sec. Ohm.
<i>zone</i>	Set the impedance for elments with this zone number.
<i>unit</i>	
0	Set the impedance for Phase - Phase or Multifunctional elements.
1	Set the impedance for Phase - Earth or Multifunctional elements.
2	Set the impedance for Multifunctional elements.

ARGUMENTS

<i>real</i>	Real part of the impedance in Ohm.
<i>imag</i>	Imaginary part of the impedance in Ohm.
<i>lineAngle</i>	The line angle in deg.
<i>Rarc</i>	The arc resistance in Ohm.
<i>inSec</i>	
0	The values are in pri. Ohm.
1	The values are in sec. Ohm.
<i>zone</i>	Set the impedance for elments with this zone number.
<i>unit</i>	
0	Set the impedance for Phase - Phase or Multifunctional elements.
1	Set the impedance for Phase - Earth or Multifunctional elements.
2	Set the impedance for Multifunctional elements.

RETURNS

0	No error occurred.
1	An error occurred or no element was found.

SetMaxI

Sets the “Max. Phase Fault Current” of the relay to the currently measured value.

```
None ElmRelay.SetMaxI()
```

SetMaxIearth

Sets the “Max. Earth Fault Current” of the relay to the currently measured value.

```
None ElmRelay.SetMaxIearth()
```

SetMinI

Sets the “Min. Phase Fault Current” of the relay to the currently measured value.

```
None ElmRelay.SetMinI()
```

SetMinIearth

Sets the “Min. Earth Fault Current” of the relay to the currently measured value.

```
None ElmRelay.SetMinIearth()
```

SetOutOfService

Sets the “Out of Service” flag of elements contained in the relay.

```
int ElmRelay.SetOutOfService(int outServ,
                             int type,
                             int zone,
                             int unit
                             )
```

ARGUMENTS

outServ

- | | |
|----------|------------------------------|
| 0 | Set elements in service. |
| 1 | Set Elements out of service. |

type

- | | |
|----------|--|
| 1 | Set the flag for overcurrent elements. |
| 2 | Set the flag for distance elements. |

zone

Set the flag for elements with this zone number (only when settings distance elements).

unit

- | | |
|----------|---|
| 0 | Set the flag for Phase-Phase or Multifunctional elements. |
| 1 | Set the flag for Phase-Earth or Multifunctional elements. |
| 2 | Set the flag for Multifunctional elements. |

RETURNS

- 0** No error occurred.
- 1** An error occurred or no element was found.

SetTime

Sets the tripping time for elements contained in the relay.

```
int ElmRelay.SetTime(int time,
                    int type,
                    int zone,
                    int unit
                    )
```

ARGUMENTS

- time* Time in s.
- type*
 - 1** Set the time for overcurrent elements.
 - 2** Set the time for distance elements.
- zone* Set the time for elements with this zone number (only when settings distance elements).
- unit*
 - 0** Set the time for Phase-Phase or Multifunctional elements.
 - 1** Set the time for Phase-Earth or Multifunctional elements.
 - 2** Set the time for Multifunctional elements.

RETURNS

- 0** No error occurred.
- 1** An error occurred or no element was found.

SlotUpdate

Triggers a slot update of the relay.

```
None ElmRelay.SlotUpdate()
```

DEPRECATED NAMES

slotupd

5.2.23 ElmRes

Overview

[AddVariable](#)
[Clear](#)
[FindColumn](#)
[FindMaxInColumn](#)
[FindMaxOfVariableInRow](#)
[FindMinInColumn](#)
[FindMinOfVariableInRow](#)
[FinishWriting](#)
[Flush](#)
[GetDescription](#)
[GetFirstValidObject](#)
[GetFirstValidObjectVariable](#)
[GetFirstValidVariable](#)
[GetNextValidObject](#)
[GetNextValidObjectVariable](#)
[GetNextValidVariable](#)
[GetNumberOfColumns](#)
[GetNumberOfRows](#)
[GetObj](#)
[GetObject](#)
[GetObjectValue](#)
[GetRelCase](#)
[GetSubElmRes](#)
[GetUnit](#)
[GetValue](#)
[GetVariable](#)
[InitialiseWriting](#)
[Load](#)
[Release](#)
[SetAsDefault](#)
[SetObj](#)
[SetSubElmResKey](#)
[SortAccordingToColumn](#)
[Write](#)
[WriteDraw](#)

AddVariable

Adds a variable to the list of monitored variables for the Result object.

```
None ElmRes.AddVariable(DataObject element,
                        str varname)
```

ARGUMENTS

element An object.
varname Variable name for object O.

DEPRECATED NAMES

AddVars

Clear

Clears all data (calculation results) written to the result file. The Variable definitions stored in the contents of ElmRes are not modified.

```
int ElmRes.Clear()
```

RETURNS

Always 0 and can be ignored.

FindColumn

Returns the index of the first header column matching the given object and/or variable name.

```
int ElmRes.FindColumn(DataObject obj,
                      [str varName]
                      )
int ElmRes.FindColumn(DataObject obj,
                      [int startCol]
                      )
int ElmRes.FindColumn(DataObject obj,
                      str varName
                      )
```

ARGUMENTS

obj (optional)

Object of matching column

varName (optional)

Variable name of matching column

startCol (optional)

Index of first checked column; Search starts at first column if colIndex is not given

RETURNS

≥ 0 column index

< 0 no valid column found

The index can be used in the ElmRes method GetData to retrieve the value of the column.

FindMaxInColumn

Find the maximum value of the variable in the given column.

```
[int row,
float value] ElmRes.FindMaxInColumn(int column)
```

ARGUMENTS

column The column index.

value (optional, out)

The maximum value found. The value is 0. in case that the maximum value was not found.

RETURNS

- < 0 The maximum value of column was not found.
- ≥ 0 The row with the maximum value of the column.

FindMaxOfVariableInRow

Find the maximum value for the given row and variable.

```
[int col,
float maxValue] ElmRes.FindMaxOfVariableInRow(str variable,
                                              int row)
```

ARGUMENTS

- variable* The variable name
- variable* The row
- maxValue* (*optional*)
 The corresponding maximum value.

RETURNS

- < 0 There is no valid value of the corresponding variable in the row.
- ≥ 0 Column index of variable.

FindMinInColumn

Find the minimum value of the variable in the given column.

```
[int row,
float value] ElmRes.FindMinInColumn(int column)
```

ARGUMENTS

- column* The column index.
- value* (*optional, out*)
 The minimum value found. The value is 0. in case that the minimum value was not found.

RETURNS

- < 0 The minimum value of column was not found.
- ≥ 0 The row with the minimum value of the column.

FindMinOfVariableInRow

Find the minimum value for the given row and variable.

```
[int col,
float minValue] ElmRes.FindMinOfVariableInRow(str variable,
                                              int row)
```

ARGUMENTS

- variable* The variable name
- variable* The row
- minValue* (*optional, out*)
The corresponding minimum value.

RETURNS

- < 0 There is no valid value of the corresponding variable in the row.
- ≥ 0 Column index of variable.

FinishWriting

Finishes the writing of values to a result file.

```
None ElmRes.FinishWriting()
```

DEPRECATED NAMES

Close

SEE ALSO

[ElmRes.InitialiseWriting\(\)](#), [ElmRes.Write\(\)](#), [ElmRes.WriteDraw\(\)](#)

Flush

This function is required in scripts which perform both file writing and reading operations. While writing to a results object (ElmRes), a small portion of this data is buffered in memory. This is required for performance reasons. Therefore, all data must be written to the disk before attempting to read the file. 'Flush' copies all data buffered in memory to the disk. After calling 'Flush' all data is available to be read from the file.

```
int ElmRes.Flush()
```

GetDescription

Get the description of a column.

```
str ElmRes.GetDescription([int column],  
                           [int short]  
                           )
```

ARGUMENTS

- column* (*optional*)
The column index. The description name of the default variable is returned if the parameter is not passed to the function.
- short* (*optional*)
- | | |
|----------|----------------------|
| 0 | long desc. (default) |
| 1 | short description |

RETURNS

Returns the description which is empty in case that the column index is not part of the data.

GetFirstValidObject

Gets the index of the column for the first valid variable in the given line. Starts at the beginning of the given line and sets the internal iterator of the result file to the found position.

```
int ElmRes.GetFirstValidObject(int row,
                              [str classNames,]
                              [str variableName,]
                              [float limit,]
                              [int limitOperator,]
                              [float limit2,]
                              [int limitOperator2]
                              )
int ElmRes.GetFirstValidObject(int row,
                              list objects
                              )
```

ARGUMENTS

row Result file row

classNames (optional)

Comma separated list of class names for valid objects. The next object of one of the given classes is searched. If not set all objects are considered as valid (default).

variableName (optional)

Name of the limiting variable. The searched object must have this variable. If not set variables are not considered (default).

limit (optional)

Limiting value for the variable.

limitOperator (optional)

Operator for checking the limiting value:

- | | |
|----------|--------------------------------|
| 0 | all values are valid (default) |
| 1 | valid values must be < limit |
| 2 | valid values must be ≤ limit |
| 3 | valid values must be > limit |
| 4 | valid values must be ≥ limit |

limit2 (optional)

Second limiting value for the variable.

limitOperator2 (optional)

Operator for checking the second limiting value:

- | | |
|-------------|--|
| < 0 | first OR second criterion must match, |
| > 0 | first AND second criterion must match, |
| 0 | all values are valid (default) |
| 1/-1 | valid values must be < limit2 |
| 2/-2 | valid values must be ≤ limit2 |

3/-3 valid values must be $>$ limit2

4/-4 valid values must be \geq limit2

objects Valid objects

RETURNS

≥ 0 column index
 < 0 no valid column found

GetFirstValidObjectVariable

Gets the index of the first valid variable of the current object in the current line. Starts at the internal iterator of the given result file and sets it to the position found.

```
int ElmRes.GetFirstValidObjectVariable([str variableNames])
```

ARGUMENTS

variableNames (optional)

Comma separated list of valid variable names. The next column with one of the given variables is searched. If empty all variables of the current object are considered as valid (default).

RETURNS

≥ 0 column index
 < 0 no valid column found

GetFirstValidVariable

Gets the index of the column for the first valid variable in the given line. Starts at the beginning of the given line and sets the internal iterator of the result file to the found position.

```
int ElmRes.GetFirstValidVariable(int row,
                                [str variableNames]
                                )
```

ARGUMENTS

row Result file row

variableNames (optional)

Comma separated list of valid variable names. The next column with one of the given variables is searched. If not set all variables are considered as valid (default).

RETURNS

≥ 0 column index
 < 0 no valid column found

GetNextValidObject

Gets the index of the column for the next valid variable (after current iterator) in the given line. Sets the internal iterator of the result file to the position found.


```

int ElmRes.GetNextValidObject ([str classNames,]
                               [str variableName,]
                               [float limit,]
                               [int limitOperator,]
                               [float limit2,]
                               [int limitOperator2]
                               )
int ElmRes.GetNextValidObject (list objects)

```

ARGUMENTS

row Result file row

classNames (*optional*)

Comma separated list of class names for valid objects. The next object of one of the given classes is searched. If not set all objects are considered as valid (default).

variableName (*optional*)

Name of the limiting variable. The searched object must have this variable. If not set variables are not considered (default).

limit (*optional*)

Limiting value for the variable.

limitOperator (*optional*)

Operator for checking the limiting value:

- 0** all values are valid (default)
- 1** valid values must be < limit
- 2** valid values must be ≤ limit
- 3** valid values must be > limit
- 4** valid values must be ≥ limit

limit2 (*optional*)

Second limiting value for the variable.

limitOperator2 (*optional*)

Operator for checking the second limiting value:

- < 0 first OR second criterion must match,
- > 0 first AND second criterion must match,
- 0** all values are valid (default)
- 1/-1** valid values must be < limit2
- 2/-2** valid values must be ≤ limit2
- 3/-3** valid values must be > limit2
- 4/-4** valid values must be ≥ limit2

objects Valid objects

RETURNS

- ≥ 0 column index
- < 0 no valid column found

GetNextValidObjectVariable

Gets the index of the column for the next valid variable of the current object in the current line. Starts at the internal iterator of the given result file and sets it to the found position.

```
int ElmRes.GetNextValidObjectVariable([str variableNames])
```

ARGUMENTS

variableNames (optional)

Comma separated list of valid variable names. The next column with one of the given variables is searched. If not set all variables are considered as valid (default).

RETURNS

≥ 0 column index
 < 0 no valid column found

GetNextValidVariable

Gets the index of the column for the next valid variable in the given line. Starts at the internal iterator of the given line and sets the internal iterator of the result file to the found position.

```
int ElmRes.GetNextValidVariable([str variableNames])
```

ARGUMENTS

variableNames (optional)

Comma separated list of valid variable names. The next column with one of the given variables is searched. If not set all variables are considered as valid (default).

RETURNS

≥ 0 column index
 < 0 no valid column found

GetNumberOfColumns

Returns the number of variables (columns) in result file excluding the default variable (e.g. time for time domain simulation).

```
int ElmRes.GetNumberOfColumns()
```

RETURNS

Number of variables (columns) in result file.

GetNumberOfRows

Returns the number of values per column (rows) stored in result object.

```
int ElmRes.GetNumberOfRows()
```

RETURNS

Returns the number of values per column stored in result object.

GetObj

Returns an object used in the result file. Positive index means objects for which parameters are being monitored (i.e. column objects). Negative index means objects which occur in written result rows as values.

```
DataObject ElmRes.GetObj(int index)
```

ARGUMENTS

index index of the object.

RETURNS

The object found or None.

GetObject

Get object of given column.

```
DataObject ElmRes.GetObject([int column])
```

ARGUMENTS

col Column index. Object of default column is returned if col is not passed.

RETURNS

The object of the variable stored in column 'column'.

GetObjectValue

Returns a value from a result object for row iX of curve col.

```
[int error,
DataObject o ] ElmRes.GetObjectValue(int iX,
                                     [int col])
```

ARGUMENTS

o (out) The object retrieved from the data.

iX The row.

col (optional)
The curve number, which equals the variable or column number, first column value (time,index, etc.) is returned when omitted.

RETURNS

- 0** when ok
- 1** when iX out of bound
- 2** when col out of bound

- 3** when invalid value is returned from a sparse file. Sparse files are written e.g. by the contingency analysis, the value is invalid in case that it was not written, because it was below the recording limit. Result files created using DPL/Python are always full and will not return invalid values.

GetRelCase

Get the contingency object for the given case number from the reliability result file.

```
DataObject ElmRes.GetRelCase(int caseNumber)
```

ARGUMENTS

caseNumber
The reliability case number

RETURNS

Returns the contingency of case number. None is returned if there is no corresponding contingency.

GetSubElmRes

Get sub-result file stored inside this.

```
DataObject ElmRes.GetSubElmRes(int value)
DataObject ElmRes.GetSubElmRes(DataObject obj)
```

ARGUMENTS

value The cnttime to look for
obj The pResElm to look for

RETURNS

None The sub result file with value=cnttime (obj=pResElm) was not found.
any other value The sub result file with value=cnttime (obj=pResElm).

GetUnit

Get the unit of a column.

```
str ElmRes.GetUnit([int column])
```

ARGUMENTS

column (optional)
The column index. The unit of the default variable is returned if the parameter is nor passed to the function.

RETURNS

Returns the unit which is empty in case that the column index is not part of the data.

GetValue

Returns a value from a result object for row *iX* of curve *col*.

```
[int error,
float d    ] ElmRes.GetValue(int iX,
                             [int col])
```

ARGUMENTS

- d (out)* The value retrieved from the data.
- iX* The row.
- col (optional)*
The curve number, which equals the variable or column number, first column value (time,index, etc.) is returned when omitted.

RETURNS

- 0** when ok
- 1** when *iX* out of bound
- 2** when *col* out of bound
- 3** when invalid value is returned from a sparse file. Sparse files are written e.g. by the contingency analysis, the value is invalid in case that it was not written, because it was below the recording limit. Result files created using DPL/Python are always full and will not return invalid values.

GetVariable

Get variable name of column

```
str ElmRes.GetVariable([int column])
```

ARGUMENTS

- column (optional)*
The column index. The variable name of the default variable is returned if the parameter is not passed to the function.

RETURNS

Returns the variable name which is empty in case that the column index is not part of the data.

InitialiseWriting

Opens the result object for writing. This function must be called before writing data for result files not stored in the script object. If arguments are passed to the function they specify the variable name, unit... of the default variable (e.g. to be used by plots as x-axis).

```
int ElmRes.InitialiseWriting()
int ElmRes.InitialiseWriting(str variableName,
                             str unit,
                             str description,
                             [str shortDescription]
                             )
```

ARGUMENTS

variableName

The variable name for the default variable (e.g. “distance”)

unit

The unit (e.g. “km”)

description

The description of the variable (e.g. “Distance from infeed”)

shortDescription

The short description (e.g. “Dist. Infeed”)

RETURNS

Always 0 and can be ignored

DEPRECATED NAMES

Init

SEE ALSO

[ElmRes.FinishWriting\(\)](#), [ElmRes.Write\(\)](#), [ElmRes.WriteDraw\(\)](#)**Load**Loads the data of a result object (**ElmRes**) in memory for reading.`None ElmRes.Load()`**Release**

Releases the data loaded to memory. This function should be used whenever several result objects are processed in a loop. Data is always released from memory automatically after execution of the current script.

`None ElmRes.Release()`**SetAsDefault**

Sets this results object as the default results object. Plots using the default result file will use this file for displaying data.

`None ElmRes.SetAsDefault()`**SetObj**

Adds an object to the objects assigned to the result file

`int ElmRes.SetObj(DataObject element)`

ARGUMENTS

element Element to store in result file

RETURNS

The index which can be used to retrieve the object from the results file. The index is < 0 if no results are recorded for the given object (e.g. a contingency in reliability calculation). The index is \geq if variables are recorded for the object.

SetSubElmResKey

Assigns a value or an object to the according ElmRes parameter.

```
None ElmRes.SetSubElmResKey(int value)
None ElmRes.SetSubElmResKey(DataObject obj)
```

ARGUMENTS

value Value to be assigned to parameter cnttime of ElmRes

value Object to be assigned to parameter pResElm of ElmRes

SortAccordingToColumn

Sorts all rows in the data loaded according to the given column. The ElmRes itself remains unchanged.

```
int ElmRes.SortAccordingToColumn(int column)
```

ARGUMENTS

col The column number.

RETURNS

0 The function executed correctly, the data was sorted correctly according to the given column.

1 The column with index column does not exist.

Write

Writes the current results to the result object.

```
int ElmRes.Write([float defaultValue])
```

RETURNS

0 on success

SEE ALSO

[ElmRes.WriteDraw\(\)](#), [ElmRes.InitialiseWriting\(\)](#), [ElmRes.FinishWriting\(\)](#)

WriteDraw

Writes current results to the result objects and updates all plots that display values from the result object.

```
int ElmRes.WriteDraw()
```

RETURNS

0 on success

5.2.24 ElmShnt

Overview

[GetGroundingImpedance](#)

GetGroundingImpedance

Returns the impedance of the internal grounding. Single phase shunts connected to neutral are considered as grounding devices themselves; i.e. instead of the dedicated grounding parameters, the shunt parameters are used.

```
[int valid,
float resistance,
float reactance ] ElmShnt.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.

resistance (out)
Real part of the grounding impedance in Ohm.

reactance (out)
Imaginary part of the grounding impedance in Ohm.

RETURNS

0 The values are invalid (e.g. because there is no internal grounding)
1 The values are valid.

5.2.25 ElmStactrl

Overview

[GetControlledHVNode](#)
[GetControlledLVNode](#)
[GetStepupTransformer](#)
[Info](#)

GetControlledHVNode

Returns the corresponding voltage controlled HV node for the machine at the specified index. Switch status are always considered.

```
DataObject ElmStactrl.GetControlledHVNode([int index = 0])
```

ARGUMENTS

index (optional)
Index of machine (starting from 0 – ...). Default is 0.

RETURNS

object Busbar/Terminal ()
None not found

GetControlledLVNode

Returns the corresponding voltage controlled LV node for the machine at specified index. Switch status are always considered.

```
DataObject ElmStactrl.GetControlledLVNode(int index)
```

ARGUMENTS

index Index of machine (starting from 0 – ...).

RETURNS

object Terminal ()
None not found

GetStepupTransformer

Performs a topological search to find the step-up transformer of the machine at the specified index.

```
DataObject ElmStactrl.GetStepupTransformer([int index,]  
                                             [int iBrkMode]  
                                             )
```

ARGUMENTS

index Index of machine (starting from 0 – ...).

iBrkMode (optional)

- 0 (default)** All switch status (open,close) are considered
- 1** Ignore breaker status (jump over open breakers)
- 2** Ignore all switch status (jump over open switches)

RETURNS

object step-up transformer
None step-up transformer not found

Info

Prints the control information in the output window. It is the same information that the button "Info" of the Station Control dialog prints.

```
int ElmStactrl.Info()
```

5.2.26 ElmSubstat

Overview

[ApplyAndResetRA](#)
[GetSplit](#)
[GetSplitCal](#)
[GetSplitIndex](#)
[GetSuppliedElements](#)
[OverwriteRA](#)
[ResetRA](#)
[SaveAsRA](#)
[SetRA](#)

ApplyAndResetRA

This function applies switch statuses of currently selected running arrangement to corresponding switches and resets the running arrangement selection afterwards. Nothing happens if no running arrangement is selected.

```
int ElmSubstat.ApplyAndResetRA()
```

RETURNS

- 1** on success
- 0** otherwise, especially if no running arrangement is selected

GetSplit

A split of a station is a group of topologically connected elements. Such a group is called split if all contained components are energized and there is at least one busbar (terminal of usage 'busbar') contained or it has connections to at least two main components (= all components except switch devices and terminals).

These splits are ordered according to the count of nodes contained and according to their priority. So each split becomes a unique index.

The function GetSplit offers access to the elements contained in a split. By calling GetSplit with an index from 0 to n, the elements belonging to the corresponding split are filled into given sets and returned.

```
[int error
list mainNodes,
list connectionCubicles,
list allElements ] ElmSubstat.GetSplit(int index)
```

ARGUMENTS

index Index of the split used to access the elements of the corresponding split. Value must be ≥ 0 .

mainNodes (out)

Terminals of same usage considered to form the most important nodes for that group. In most cases, this is the group of contained busbars.

connectionCubicles (optional, out)

All cubicles (of terminals inside the station) that point to an element that sits outside the station or to an element that is connected to a terminal outside the station are filled into the set connectionCubicles. (The connection element (branch) can

be accessed by calling `GetBranch()` on each of these cubicles. The terminals of these cubicles (parents) must not necessarily be contained in any split. They could also be separated by a disconnecting component.)

allElements(optional, out)

All elements (class `Elm*`) of the split that have no connection to elements outside the station are filled into this set.

RETURNS

- 0** success, split of that index exists and is returned.
- 1** indicates that there exists no split with given index. (Moreover, this means that there is no split with index n greater than this value.)

SEE ALSO

[ElmSubstat.GetSplitCal\(\)](#), [ElmSubstat.GetSplitIndex\(\)](#),

GetSplitCal

This function determines the elements that belong to a split. In contrast to [ElmSubstat.GetSplit\(\)](#) it is based on calculation instead of pure edit object topology. This means the returned nodes correspond to the calculation nodes, the interconnecting cubicles are those connecting nodes of different splits.

Note: As this function relies on calculation nodes it can only be executed after a calculation has been performed (e.g. load flow calculation).

```
[int error,
list nodes,
list connectionCubicles,
list elements          ] ElmSubstat.GetSplitCal(int index)
```

ARGUMENTS

index Index of the split used to access the elements of the corresponding split. Refers to same split as index in [ElmSubstat.GetSplit\(\)](#). Value must be ≥ 0 .

nodes (out)

A set that is filled with terminals. There is one terminal returned for each calculation node in the split.

connectionCubicles (optional, out)

This set is filled with all cubicles that point from a calculation node of current split to another calculation node that does not belong to that split. The connecting element can be accessed by calling `GetBranch()` on such a cubicle.

elements (optional, out)

This set is filled with network elements that are connected to a calculation node of current split and have exactly one connection, i.e. these elements are completely contained in the split.

RETURNS

- 0** success, split of that index exists and is returned.
- 1** indicates that there exists no split with given index. (Moreover, this means that there is no split with index n greater than this value.)

SEE ALSO

[ElmSubstat.GetSplit\(\)](#)

GetSplitIndex

This function returns the index of the split that contains passed object.

```
int ElmSubstat.GetSplitIndex(DataObject o)
```

ARGUMENTS

o Object for which the split index is to be determined.

RETURNS

≥ 0 index of split in which element is contained
-1 given object does not belong to any split of that station

SEE ALSO

[ElmSubstat.GetSplit\(\)](#)

GetSuppliedElements

Returns the network components that are supplied by the transformer (located in the station). A network component is considered to be supplied by a transformer if a topological path from the transformer to the component exists. A valid topological path in this sense is a path that starts at the transformer's HV side in direction of transformer (not in direction of HV connected node) and stops at

- network components that are out of calculation,
- network components that are not active (e.g. hidden or those of currently inactive grids),
- open switches,
- connections leading to a higher voltage level.

Generally, all network components of such a path are considered to be supplied by the transformer.

Exceptions are components that are out of calculation or in-active. Those components are never considered to be supplied by any transformer. A transformer is never considered to supply itself. Composite components such as ElmBranch, ElmSubstat, ElmTrfstat are considered to be supplied by a transformer if all energized components inside that composite are supplied by the transformer.

```
list ElmSubstat.GetSuppliedElements([int inclNested])
```

ARGUMENTS

inclNested (optional)

0 Do not include components that are supplied by nested supplying stations
1 (default) Include components that are supplied by nested stations

SEE ALSO

[ElmTr2.GetSuppliedElements\(\)](#), [ElmTr3.GetSuppliedElements\(\)](#)

OverwriteRA

This function overwrites switch statuses stored in an existing running arrangement with actual switch statuses of the substation. This is only possible if the substation has no running arrangement selected and given running arrangement is valid for substation the method was called on.

```
int ElmSubstat.OverwriteRA(DataObject ra)
```

ARGUMENTS

ra Given running arrangement

RETURNS

1 If given running arrangement was successfully overwritten;
0 otherwise

ResetRA

This function resets the running arrangement selection for the substation it was called on.

```
None ElmSubstat.ResetRA()
```

SaveAsRA

When called on a substation that has no running arrangement selected, a new running arrangement is created and all switch statuses of all running arrangement relevant switches (for that substation) are saved in it. The running arrangement is stored in project folder "Running Arrangement" and its name is set to given locname. The new running arrangement is not selected automatically.

(No new running arrangement is created if this method is called on a substation that has currently a running arrangement selected).

```
DataObject ElmSubstat.SaveAsRA(str locname)
```

ARGUMENTS

locname Name of the new running arrangement (if name is already used, an increment (postfix) is added to make it unique).

RETURNS

Newly created 'IntRunarrange' object on success, otherwise None.

SetRA

This function sets the running arrangement selection for the substation it was called on. The switch statuses are now determined by the values stored in the running arrangement.

```
int ElmSubstat.SetRA(DataObject ra)
```

ARGUMENTS

ra running arrangement that is valid for the substation

RETURNS

- 1** If given running arrangement was successfully set;
- 0** otherwise (e.g. given *ra* is not valid for that substation)

5.2.27 ElmSvs

Overview

[GetStepupTransformer](#)

GetStepupTransformer

Performs a topological search to find the step-up transformer of the static VAR system.

```
DataObject ElmSvs.GetStepupTransformer(float voltage,
                                         int swStatus
                                         )
```

ARGUMENTS

voltage voltage level at which the search will stop

swStatus consideration of switch status. Possible values are:

- 0** consider all switch status
- 1** ignore breaker status
- 2** ignore all switch status

RETURNS

Returns the first collected step-up transformer object. It is empty if not found (e.g. start terminal already at *hvVoltage*).

5.2.28 ElmSym

Overview

[CalcEfficiency](#)
[Derate](#)
[Disconnect](#)
[GetAvailableGenPower](#)
[GetGroundingImpedance](#)
[GetMotorStartingFlag](#)
[GetStepupTransformer](#)
[IsConnected](#)
[Reconnect](#)
[ResetDerating](#)

CalcEfficiency

Calculate efficiency for connected storage model at given active power value.

```
float ElmSym.CalcEfficiency(float activePowerMW)
```

ARGUMENTS

activePowerMW
Active power value to calculate efficiency for.

RETURNS

Returns the resulting efficiency in p.u.

Derate

Derates the value of the Max. Active Power Rating according to the specified value given in MW.

The following formula is used: $P_{max_uc} = P_{max_uc} - "Deratingvalue"$.

```
None ElmSym.Derate(float deratingP)
```

ARGUMENTS

deratingP Derating value

Disconnect

Disconnects a synchronous machine by opening the first circuit breaker. The topological search performed to find such a breaker, stops at any busbar.

```
int ElmSym.Disconnect()
```

RETURNS

- 0** breaker already open or successfully opened
- 1** an error occurred (no breaker found, open action not possible (earthing / RA))

GetAvailableGenPower

Returns the available power that can be dispatched from the generator, for the particular study time.

For the case of conventional generators (no wind generation selected), the available power is equal to the nominal power specified.

For wind generators, the available power will depend on the wind model specified:

- No Wind Model: No available power.
- Stochastic Wind Model: Given the specified mean wind speed, the available power is calculated from the Power Curve. If the units of the Power Curve are in MW, the returned value is directly the available power. In the other hand, if the units are in PU, the returned value is multiplied by the nominal power of the generator to return the available power.
- Time Series Characteristics of Active Power Contribution: The available power is the average of the power values (in MW) obtained from all the specified time characteristics for the current study time.

- **Time Series Characteristics of Wind Speed:** The available power is calculated with the average of the power values (in MW) calculated for all the specified time characteristics. A power value for any time characteristic is calculated by obtaining the wind speed for the current study time, and then calculating the power from the specified Power Curve. If the units of the Power Curve are in MW, the returned value is directly the power value. In the other hand, if the units are in PU, the returned value is multiplied by the nominal power of the generator to return the power value.

For motors, the available power is zero.

```
float ElmSym.GetAvailableGenPower()
```

RETURNS

Available generation power

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,  
float resistance,  
float reactance ] ElmSym.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.

resistance (out)

Real part of the grounding impedance in Ohm.

reactance (out)

Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

GetMotorStartingFlag

Returns the starting motor condition.

```
int ElmSym.GetMotorStartingFlag()
```

RETURNS

Returns the motor starting condition. Possible values are:

- 1** in the process of being calculated
- 0** not calculated
- 1** successful start
- 2** unsuccessful start

GetStepupTransformer

Performs a topological search to find the step-up transformer of the synchronous machine.

```
DataObject ElmSym.GetStepupTransformer(float voltage,
                                       int swStatus
                                       )
```

ARGUMENTS

voltage voltage level at which the search will stop

swStatus consideration of switch status. Possible values are:

- 0** consider all switch status
- 1** ignore breaker status
- 2** ignore all switch status

RETURNS

Returns the first collected step-up transformer object. It is empty if not found (e.g. start terminal already at hvVoltage).

IsConnected

Checks if a synchronous machine is already connected to any busbar.

```
int ElmSym.IsConnected()
```

RETURNS

- 0** false, not connected to a busbar
- 1** true, generator is connected to a busbar

Reconnect

Connects a synchronous machine by closing all switches (breakers and isolators) up to the first breaker on the HV side of a transformer. The topological search to find all the switches, stops at any busbar.

```
int ElmSym.Reconnect()
```

RETURNS

- 0** the machine was successfully closed
- 1** a error occurred and the machine could not be connected to any busbar

ResetDerating

Resets the derating value, setting the Max. Active Power Rating according to the rating factor. The following formula is used: $P_{max_uc} = p_{maxratf} * P_n * n_{gnum}$.

```
None ElmSym.ResetDerating()
```

5.2.29 ElmTerm

Overview

[GetBusType](#)
[GetCalcRelevantCubicles](#)
[GetConnectedBrkCubicles](#)
[GetConnectedCubicles](#)
[GetConnectedMainBuses](#)
[GetConnectionInfo](#)
[GetEquivalentTerminals](#)
[GetMinDistance](#)
[GetNextHVBus](#)
[GetNodeName](#)
[GetSepStationAreas](#)
[HasCreatedCalBus](#)
[IsElectrEquivalent](#)
[IsEquivalent](#)
[IsInternalNodeInStation](#)
[UpdateSubstationTerminals](#)

GetBusType

Gets busbar calculation type.

```
int ElmTerm.GetBusType()
```

RETURNS

- 0** No valid calculation (load flow).
- 1** PQ busbar.
- 2** PV busbar.
- 3** Slack busbar.

GetCalcRelevantCubicles

This function gets calculation relevant cubicles of this terminal.

```
list ElmTerm.GetCalcRelevantCubicles()
```

RETURNS

Set of calculation relevant cubicles.

GetConnectedBrkCubicles

Function gets the set of cubicles connected with the breaker and this terminal.

```
list ElmTerm.GetConnectedBrkCubicles([int ignoreSwitchStates])
```

ARGUMENTS

- ignoreSwitchStates (optional)*
Ignore switch status flag 1 or not 0 (=default).

RETURNS

Set of cubicles.

GetConnectedCubicles

Function gets the set of cubicles connected with this terminal.

```
list ElmTerm.GetConnectedCubicles([int ignoreSwitchStates])
```

ARGUMENTS

ignoreSwitchStates (optional)
Ignore switch status flag 1 or not 0 (=default).

RETURNS

Set of cubicles.

GetConnectedMainBuses

Function gets the set of connected main buses.

```
list ElmTerm.GetConnectedMainBuses([int considerSwitches])
```

ARGUMENTS

considerSwitches (optional)
Consider switch state (default 1).

RETURNS

Set of main buses connected to the terminal.

GetConnectionInfo

Gets connection information of this terminal. Requires valid load flow calculation. Input arguments are filled with the value after function call.

```
[int error,  
float closedSwitches,  
float allSwitches,  
float nonSwitchingDevices,  
float closedAndNonSwitchingDevices,  
float allDevices,  
float connectedNodes,  
float mainNodes] ElmTerm.GetConnectionInfo()
```

ARGUMENTS

closedSwitches
Number of closed switch devices.

allSwitches
Number of total switch devices.

nonSwitchingDevices
Number of non-switch devices.

closedAndNonSwitchingDevices

Number of total closed and non-switch devices (closedSwitches+nonSwitchingDevices).

allDevices

Number of total switch and non-switch devices (allSwitches+nonSwitchingDevices).

connectedNodes

Number of total nodes connected via couplers.

mainNodes

Number of total main nodes.

RETURNS

Return value is always 0 and has no meaning.

GetEquivalentTerminals

Returns a set of all terminals that are equivalent to current one. Equivalent means that those terminals are topologically connected only by

- closed switching devices (ElmCoup, RelFuse) or
- lines of zero length (line droppers).

```
list ElmTerm.GetEquivalentTerminals()
```

RETURNS

All terminals that are equivalent to current one. Current one is also included so the set is never empty.

SEE ALSO

[ElmTerm.IsEquivalent\(\)](#)

GetMinDistance

This function determines the shortest path between the terminal the function was called on and the terminal that was passed as first argument. The distance is determined on network topology regarding the length of the traversed component (i.e. only lines have an influence on distance).

```
float ElmTerm.GetMinDistance(DataObject term)
[float minDistance,
list path          ] ElmTerm.GetMinDistance(DataObject term,
                                             [int considerSwitches,]
                                             [list limitToNodes,]
                                             [list elementsToIgnore,]
                                             )
```

ARGUMENTS

term Terminal to which the shortest path is determined.

considerSwitches (optional)

- | | |
|----------|---|
| 0 | Traverse all components, ignore switch states |
| 1 | Do not traverse open switch devices (default) |

path (optional, out)

If given, all components of the found shortest path are put into this set.

limitToNodes(optional)

If given, the shortest path is searched only within this set of nodes. Please note, when limiting search to a given set of nodes, the start and end terminals (for which the distance is determined) must be part of this set (otherwise distance == -1).

elementsToIgnore(optional)

If given, these objects (e.g. nodes) are ignored in the search and not traversed (=considered as not existing).

RETURNS

- < 0 If there is no path between the two terminals
- ≥ 0 Distance of shortest path in km

GetNextHVBus

This function returns the nearest connected busbar that has a higher voltage level. To detect this bus, a breadth-first search on the net topology is executed. The traversal stops on each element that is out of service and on each opened switch device. The criterion for higher voltage level is passing a transformer to HV side. No junction nor internal nodes shall be considered. Two winding transformers with equal voltage on both sides are ignored (simply passed by the search).

```
DataObject ElmTerm.GetNextHVBus()
```

RETURNS

- object** First busbar found.
- None** If no busbar was found.

GetNodeName

For terminals inside a station, this function returns a unique name for the split the terminal is located in. The name is built on first five characters of the station's short name plus the split index separated by an underscore. E.g. "USTAT_1".

For terminals inside a branch (*ElmBranch*) the returned name is just a concatenation of the branch name and the terminal's name.

For all other terminals not inside a branch or a station the node name corresponds to the terminal's name.

```
str ElmTerm.GetNodeName()
```

RETURNS

- Node name as described above. Never empty.

GetSepStationAreas

Function gets all separate areas within the substation linked to this terminal. In this manner, area is any part between two nodes.

```
list ElmTerm.GetSepStationAreas([int considerSwitches])
```

ARGUMENTS

considerSwitches (optional)
Consider switch state (default 1).

RETURNS

Set of all separate areas in this substation.

HasCreatedCalBus

This function checks if the valid calculation exists for this terminal (i.e. load flow). If it exists, then the calculation parameters could be retrieved.

```
int ElmTerm.HasCreatedCalBus()
```

RETURNS

- 1** Valid calculation exists.
- 0** No valid calculation.

IsElectrEquivalent

Function checks if two terminals are electrically equivalent. Two terminals are said to be electrically equivalent if they are topologically connected only by

- closed switching devices (*ElmCoup*, *RelFuse*) or
- lines of zero length (line droppers) or
- branch components whose impedance is below given thresholds ($R \leq \max R$ and $X \leq \max X$)

```
int ElmTerm.IsElectrEquivalent(DataObject terminal,
                               float maxR,
                               float maxX
                               )
```

ARGUMENTS

terminal Terminal to which the 'method called terminal' is connected to.
maxR Given threshold for the resistance of branch elements (must be given in Ohm).
maxX Given threshold for the reactance of branch elements (must be given in Ohm).

RETURNS

- 1** If terminal on which the method was called is electrical equivalent to terminal that was passed as argument
- 0** Otherwise

SEE ALSO

[ElmTerm.IsEquivalent\(\)](#)

IsEquivalent

Function checks if two terminals are topologically connected only by

- closed switching devices (ElmCoup, RelFuse) or
- lines of zero length (line droppers).

IsEquivalent defines a relation that is

- symmetric (Term1.IsEquivalent(Term2) -> Term2.IsEquivalent(Term1)),
- reflexive (Term1.IsEquivalent(Term1)) and
- transitive (Term1.IsEquivalent(Term2) and Term2.IsEquivalent(Term3) -> Term1.IsEquivalent(Term3));

```
int ElmTerm.IsEquivalent(DataObject terminal)
```

ARGUMENTS

terminal Terminal (object of class ElmTerm) that is checked to be equivalent to the terminal on which the function was called on. Passing None is not allowed and will result in a scripting error.

RETURNS

- 1** If terminal on which the method was called is connected to terminal that was passed as argument only by closed switching devices or by lines of zero length
- 0** Otherwise (terminals are not connected or connected by other components than switching devices / lines of zero length)

SEE ALSO

[ElmTerm.GetEquivalentTerminals\(\)](#) [ElmTerm.IsElectrEquivalent\(\)](#)

IsInternalNodeInStation

Function checks if the terminal is an internal node located in a station (*ElmSubstat*, *ElmTrfstat*).

```
int ElmTerm.IsInternalNodeInSubStation()
```

RETURNS

- 1** Terminal is a node of usage 'internal' and is located in a station.
- 0** Not internal node or not in a station, or both.

UpdateSubstationTerminals

Updates all nodes within the substation to the new voltage and/or phase technology. Applicable for all busbars and junction nodes. The highest voltage is taken as the leading one.

```
None ElmTerm.UpdateSubstationTerminals(int volt,
                                         int phs
                                         )
```

ARGUMENTS

- volt* Updates nominal voltages (<> 0)
- phs* Updates phase technology (<> 0)

5.2.30 ElmTr2**Overview**

[CreateEvent](#)
[GetGroundingImpedance](#)
[GetSuppliedElements](#)
[GetTapPhi](#)
[GetTapRatio](#)
[GetZ0pu](#)
[GetZpu](#)
[IsQuadBooster](#)
[NTap](#)

CreateEvent

For the corresponding transformer, a Tap Event (EvtTap) is created for the simulation.

```
int ElmTr2.CreateEvent([int tapAction,]
                      [int tapPos]
                      )
```

ARGUMENTS

- tapAction* (*optional*)
 0=increase tap; 1=decrease tap; 2=set tap to tapPos; 3>manual; 4=automatic
- tapPos* (*optional*)
 Position of tap.

RETURNS

- 0 on success

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,
float resistance,
float reactance ] ElmTr2.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

- busIdx* Bus index where the grounding should be determined.
- resistance* (*out*)
 Real part of the grounding impedance in Ohm.
- reactance* (*out*)
 Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

GetSuppliedElements

Returns the network components that are supplied by the transformer.

A network component is considered to be supplied by a transformer if a topological path from the transformer to the component exists. A valid topological path in this sense is a path that starts at the transformer's HV side in direction of transformer (not in direction of HV connected node) and stops at

- network components that are out of calculation,
- network components that are not active (e.g. hidden or those of currently inactive grids),
- open switches,
- connections leading to a higher voltage level.

Generally all network components of such a path are considered to be supplied by the transformer. Exceptions are components that are out of calculation or in-active. Those components are never considered to be supplied by any transformer.

A transformer is never considered to supply itself.

Composite components such as *ElmBranch*, *ElmSubstat*, *ElmTrfstat* are considered to be supplied by a transformer if all energized components inside that composite are supplied by the transformer.

```
list ElmTr2.GetSuppliedElements([int inclNested])
```

ARGUMENTS

inclNested (optional)

- 0** Only include components which are directly supplied by the transformer (not nested components)
- 1** Include nested components and components that are directly supplied by the transformer (default)

SEE ALSO

[ElmTr3.GetSuppliedElements\(\)](#), [ElmSubstat.GetSuppliedElements\(\)](#), [ElmTrfstat.GetSuppliedElements\(\)](#)

GetTapPhi

Gets the tap phase shift in deg of the transformer for given tap position.

```
float ElmTr2.GetTapPhi(int itappos,
                      int inclPhaseShift
                      )
```

ARGUMENTS

itappos Tap position

inclPhaseShift

1 = Includes the vector group phase shift, 0 = consider only the tap phase shift

RETURNS

Returns the tap phase shift angle of the transformer for given tap position

GetTapRatio

Gets the voltage ratio of the transformer for given tap position.

```
float ElmTr2.GetTapRatio(int itappos,
                        int onlyTapSide,
                        int includeNomRatio
                        )
```

ARGUMENTS

itappos Tap position

onlyTapSide

1 = ratio only for given side., 0 = total ratio

includeNomRatio

1 = Includes nominal ratio of the transformer, 0 = consider only tap ratio

RETURNS

Returns the voltage ratio of the transformer for given tap position

GetZ0pu

Gets the zero-sequence impedance in p.u. of the transformer for the specified tap position. If the tap position is out of the tap changer range, the respective min. or max. position will be used.

```
[float r0pu,
float x0pu ] ElmTr2.GetZ0pu(int itappos,
                           int systembase)
```

ARGUMENTS

itappos Tap position

r0pu (out)

Resistance in p.u.

x0pu (out)

Reactance in p.u.

systembase

0 p.u. is based on rated power.

1 p.u. is based on system base (e.g. 100MVA).

GetZpu

Gets the impedance in p.u. of the transformer for the specified tap position. If the tap position is out of the tap changer range, the respective min. or max. position will be used.

```
[float rpu,
float xpu ] ElmTr2.GetZpu(int itappos,
                        int systembase)
```

ARGUMENTS

itappos Tap position*rpu (out)* Resistance in p.u.*xpu (out)* Reactance in p.u.*systembase***0** p.u. is based on rated power.**1** p.u. is based on system base (e.g. 100MVA).**IsQuadBooster**Returns whether transformer is a quadbooster; i.e. checks phase shift angle modulus 180° .

```
int ElmTr2.IsQuadBooster()
```

RETURNS

'1' if quadbooster, else '0'

NTap

Gets the transformer tap position.

```
int ElmTr2.NTap()
```

RETURNS

The tap position.

5.2.31 ElmTr3**Overview**

[CreateEvent](#)
[GetGroundingImpedance](#)
[GetSuppliedElements](#)
[GetTapPhi](#)
[GetTapRatio](#)
[GetTapZDependentSide](#)
[GetZ0pu](#)
[GetZpu](#)
[IsQuadBooster](#)
[NTap](#)

CreateEvent

For the corresponding transformer, a Tap Event (EvtTap) is created for the simulation.

```
None ElmTr3.CreateEvent([int tapAction = 2,]
                        [int tapPos = 0,]
                        [int busIdx = 0]
                        )
```

ARGUMENTS

tapAction (optional)

0=increase tap; 1=decrease tap; 2=set tap to tapPos; 3>manual; 4=automatic

tapPos (optional)

Position of tap.

busIdx (optional)

Bus index.

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,
float resistance,
float reactance ] ElmTr3.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.*resistance* (out)

Real part of the grounding impedance in Ohm.

reactance (out)

Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

GetSuppliedElements

Returns the network components that are supplied by the transformer.

A network component is considered to be supplied by a transformer if a topological path from the transformer to the component exists. A valid topological path in this sense is a path that starts at the transformer's HV side in direction of transformer (not in direction of HV connected node) and stops at

- network components that are out of calculation,
- network components that are not active (e.g. hidden or those of currently inactive grids),
- open switches,
- connections leading to a higher voltage level.

Generally all network components of such a path are considered to be supplied by the transformer. Exceptions are components that are out of calculation or in-active. Those components are never considered to be supplied by any transformer.

A transformer is never considered to supply itself.

Composite components such as *ElmBranch*, *ElmSubstat*, *ElmTrfstat* are considered to be supplied by a transformer if all energized components inside that composite are supplied by the transformer.

```
list ElmTr3.GetSuppliedElements([int inclNested])
```

ARGUMENTS

inclNested (optional)

- 0** Only include components which are directly supplied by the transformer (not nested components)
- 1** Include nested components and components that are directly supplied by the transformer (default)

SEE ALSO

[ElmTr2.GetSuppliedElements\(\)](#), [ElmSubstat.GetSuppliedElements\(\)](#), [ElmTrfstat.GetSuppliedElements\(\)](#)

GetTapPhi

Gets the tap phase shift in deg of the transformer for given tap position and side.

```
float ElmTr2.GetTapPhi(int iSide,
                      int itappos,
                      int inclPhaseShift
                      )
```

ARGUMENTS

- iSide* for tap at side (0=Hv, 1=Mv, 2=Lv)
- itappos* Tap position for corresponding side
- inclPhaseShift*
1 = Includes the vector group phase shift, 0 = consider only the tap phase shift

RETURNS

Returns the tap phase shift angle of the transformer for given tap position and side

GetTapRatio

Gets the voltage ratio of the transformer for given tap position and side.

```
float ElmTr2.GetTapRatio(int iSide,
                        int itappos,
                        int includeNomRatio
                        )
```

ARGUMENTS

- iSide* for tap at side (0=Hv, 1=Mv, 2=Lv)
- itappos* Tap position at corresponding side
- includeNomRatio*
1 = Includes nominal ratio of the transformer, 0 = consider only tap ratio

RETURNS

Returns the voltage ratio of the transformer for given tap position and side

GetTapZDependentSide

Get tap side used for the dependent impedance

```
int ElmTr3.GetTapZDependentSide()
```

RETURNS

- 1** if no tap dependent impedance is defined
- 0** for HV tap
- 1** for MV tap
- 2** for LV tap

GetZ0pu

Gets the zero-sequence impedance in p.u. of the transformer for the specified tap position. If the tap position is out of the tap changer range, the respective min. or max. position will be used.

```
[float r0pu,
float x0pu ] ElmTr3.GetZ0pu(int itappos,
                             int iSide,
                             int systembase)
```

ARGUMENTS

itappos Tap position of the z-dependent tap

iSide

- 0** Get the HV-MV impedance.
- 1** Get the MV-LV impedance.
- 2** Get the LV-HV impedance.

r0pu (out)

Resistance in p.u.

x0pu (out)

Reactance in p.u.

systembase

- 0** p.u. is based on rated power.
- 1** p.u. is based on system base (e.g. 100MVA).

GetZpu

Gets the impedance in p.u. of the transformer for the specified tap position. If the tap position is out of the tap changer range, the respective min. or max. position will be used.

```
[float rpu,
float xpu ] ElmTr3.GetZpu(int itappos,
                           int iSide,
                           int systembase)
```

ARGUMENTS

itappos Tap position of the z-dependent tap

iSide

- 0** Get the HV-MV impedance.
- 1** Get the MV-LV impedance.
- 2** Get the LV-HV impedance.

rpu (out) Resistance in p.u.

xpu (out) Reactance in p.u.

systembase

0 p.u. is based on rated power.

1 p.u. is based on system base (e.g. 100MVA).

IsQuadBooster

Returns whether transformer is a quadbooster or not, i.e. checks phase shift angle modulus 180°.

```
int ElmTr3.IsQuadBooster()
```

RETURNS

'1' if the transformer phase shift angle modulus 180° does not equal 0 at any of the sides LV, MV, HV, else '0'

NTap

Gets the transformer tap position of a given bus.

```
int ElmTr3.NTap(int bus)
```

ARGUMENTS

bus 0=HV, 1=MV, 2=LV

RETURNS

The tap position.

5.2.32 ElmTr4

Overview

[CreateEvent](#)
[GetGroundingImpedance](#)
[GetSuppliedElements](#)
[GetTapPhi](#)
[GetTapRatio](#)
[GetTapZDependentSide](#)
[GetZ0pu](#)
[GetZpu](#)
[IsQuadBooster](#)
[NTap](#)

CreateEvent

For the corresponding transformer, a Tap Event (EvtTap) is created for the simulation.

```
None ElmTr4.CreateEvent([int tapAction = 2,]  
                        [int tapPos = 0,])
```

```
[int busIdx = 0]
)
```

ARGUMENTS

tapAction (optional)

0=increase tap; 1=decrease tap; 2=set tap to tapPos; 3>manual; 4=automatic

tapPos (optional)

Position of tap.

busIdx (optional)

Bus index.

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,
float resistance,
float reactance ] ElmTr4.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.*resistance* (out)

Real part of the grounding impedance in Ohm.

reactance (out)

Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

GetSuppliedElements

Returns the network components that are supplied by the transformer.

A network component is considered to be supplied by a transformer if a topological path from the transformer to the component exists. A valid topological path in this sense is a path that starts at the transformer's HV side in direction of transformer (not in direction of HV connected node) and stops at

- network components that are out of calculation,
- network components that are not active (e.g. hidden or those of currently inactive grids),
- open switches,
- connections leading to a higher voltage level.

Generally all network components of such a path are considered to be supplied by the transformer. Exceptions are components that are out of calculation or in-active. Those components are never considered to be supplied by any transformer.

A transformer is never considered to supply itself.

Composite components such as *ElmBranch*, *ElmSubstat*, *ElmTrfstat* are considered to be supplied by a transformer if all energized components inside that composite are supplied by the transformer.

```
list ElmTr4.GetSuppliedElements([int inclNested])
```

ARGUMENTS

inclNested (optional)

- 0** Only include components which are directly supplied by the transformer (not nested components)
- 1** Include nested components and components that are directly supplied by the transformer (default)

SEE ALSO

[ElmTr2.GetSuppliedElements\(\)](#), [ElmSubstat.GetSuppliedElements\(\)](#), [ElmTrfstat.GetSuppliedElements\(\)](#)

GetTapPhi

Gets the tap phase shift in deg of the transformer for given tap position and side.

```
float ElmTr4.GetTapPhi(int iSide,
                      int itappos,
                      int inclPhaseShift
                      )
```

ARGUMENTS

iSide for tap at side (0=HV, 1=LV1, 2=Lv2, 3=Lv3)

itappos Tap position for corresponding side

inclPhaseShift

1 = Includes the vector group phase shift, 0 = consider only the tap phase shift

RETURNS

Returns the tap phase shift angle of the transformer for given tap position and side

GetTapRatio

Gets the voltage ratio of the transformer for given tap position and side.

```
float ElmTr4.GetTapRatio(int iSide,
                        int itappos,
                        int includeNomRatio
                        )
```

ARGUMENTS

iSide for tap at side (0=HV, 1=LV1, 2=Lv2, 3=Lv3)

itappos Tap position at corresponding side

includeNomRatio

1 = Includes nominal ratio of the transformer, 0 = consider only tap ratio

RETURNS

Returns the voltage ratio of the transformer for given tap position and side

GetTapZDependentSide

Get tap side used for the dependent impedance

```
int ElmTr4.GetTapZDependentSide()
```

RETURNS

- 1** if no tap dependent impedance is defined
- 0** for HV tap
- 1** for LV1 tap
- 2** for LV2 tap
- 2** for LV3 tap

GetZ0pu

Gets the zero-sequence impedance in p.u. of the transformer for the specified tap position. If the tap position is out of the tap changer range, the respective min. or max. position will be used.

```
[float r0pu,
float x0pu ] ElmTr4.GetZ0pu(int itappos,
                             int iSide,
                             int systembase)
```

ARGUMENTS

itappos Tap position of the z-dependent tap

iSide

- 0** Get the HV-LV1 impedance.
- 1** Get the HV-LV2 impedance.
- 2** Get the HV-LV3 impedance.
- 3** Get the LV1-LV2 impedance.
- 4** Get the LV1-LV3 impedance.
- 5** Get the LV2-LV3 impedance.

r0pu (out)

Resistance in p.u.

x0pu (out)

Reactance in p.u.

systembase

- 0** p.u. is based on rated power.
- 1** p.u. is based on system base (e.g. 100MVA).

GetZpu

Gets the impedance in p.u. of the transformer for the specified tap position. If the tap position is out of the tap changer range, the respective min. or max. position will be used.

```
[float rpu,
float xpu ] ElmTr4.GetZpu(int itappos,
                           int iSide,
                           int systembase)
```

ARGUMENTS

itappos Tap position of the z-dependent tap

iSide

- | | |
|----------|----------------------------|
| 0 | Get the HV-LV1 impedance. |
| 1 | Get the HV-LV2 impedance. |
| 2 | Get the HV-LV3 impedance. |
| 3 | Get the LV1-LV2 impedance. |
| 4 | Get the LV1-LV3 impedance. |
| 5 | Get the LV2-LV3 impedance. |

rpu (out) Resistance in p.u.

xpu (out) Reactance in p.u.

systembase

- | | |
|----------|---|
| 0 | p.u. is based on rated power. |
| 1 | p.u. is based on system base (e.g. 100MVA). |

IsQuadBooster

Returns whether transformer is a quadbooster or not, i.e. checks phase shift angle modulus 180°.

```
int ElmTr4.IsQuadBooster()
```

RETURNS

'1' if the transformer phase shift angle modulus 180° does not equal 0 at any of the sides HV, LV1, LV2, LV3, else '0'

NTap

Gets the transformer tap position.

```
int ElmTr4.NTap(float busIdx)
```

ARGUMENTS

busIdx 0=HV, 1=MV, 2=LV

RETURNS

The tap position.

5.2.33 ElmTrfstat

Overview

[GetSplit](#)
[GetSplitCal](#)
[GetSplitIndex](#)
[GetSuppliedElements](#)

GetSplit

A split of a station is a group of topologically connected elements. Such a group is called split if all contained components are energized and there is at least one busbar (terminal of usage 'busbar') contained or it has connections to at least two main components (= all components except switch devices and terminals).

These splits are ordered according to the count of nodes contained and according to their priority. So each split becomes a unique index.

The function GetSplit offers access to the elements contained in a split. By calling GetSplit with an index from 0 to n, the elements belonging to the corresponding split are filled into given sets and returned.

```
[int error
list mainNodes,
list connectionCubicles,
list allElements      ] ElmTrfstat.GetSplit(int index)
```

ARGUMENTS

index Index of the split used to access the elements of the corresponding split. Value must be ≥ 0 .

mainNodes (out)

Terminals of same usage considered to form the most important nodes for that group. In most cases, this is the group of contained busbars.

connectionCubicles (optional, out)

All cubicles (of terminals inside the station) that point to an element that sits outside the station or to an element that is connected to a terminal outside the station are filled into the set connectionCubicles. (The connection element (branch) can be accessed by calling GetBranch() on each of these cubicles. The terminals of these cubicles (parents) must not necessarily be contained in any split. They could also be separated by a disconnecting component.)

allElements(optional, out)

All elements (class Elm*) of the split that have no connection to elements outside the station are filled into this set.

RETURNS

- 0** success, split of that index exists and is returned.
- 1** indicates that there exists no split with given index. (Moreover, this means that there is no split with index n greater than this value.)

SEE ALSO

[ElmTrfstat.GetSplitCal\(\)](#), [ElmTrfstat.GetSplitIndex\(\)](#),

GetSplitCal

This function determines the elements that belong to a split. In contrast to [ElmTrfstat.GetSplit\(\)](#) it is based on calculation instead of pure edit object topology. This means the returned nodes correspond to the calculation nodes, the interconnecting cubicles are those connecting nodes of different splits.

Note: As this function relies on calculation nodes it can only be executed after a calculation has been performed (e.g. load flow calculation).

```
[int error,
list nodes,
list connectionCubicles,
list elements ] ElmTrfstat.GetSplitCal(int index)
```

ARGUMENTS

index Index of the split used to access the elements of the corresponding split. Refers to same split as index in [ElmTrfstat.GetSplit\(\)](#). Value must be ≥ 0 .

nodes (out) A set that is filled with terminals. There is one terminal returned for each calculation node in the split.

connectionCubicles (optional, out) This set is filled with all cubicles that point from a calculation node of current split to another calculation node that does not belong to that split. The connecting element can be accessed by calling [GetBranch\(\)](#) on such a cubicle.

elements (optional, out) This set is filled with network elements that are connected to a calculation node of current split and have exactly one connection, i.e. these elements are completely contained in the split.

RETURNS

- 0** success, split of that index exists and is returned.
- 1** indicates that there exists no split with given index. (Moreover, this means that there is no split with index n greater than this value.)

SEE ALSO

[ElmTrfstat.GetSplit\(\)](#)

GetSplitIndex

This function returns the index of the split that contains passed object.

```
int ElmTrfstat.GetSplitIndex(DataObject o)
```

ARGUMENTS

o Object for which the split index is to be determined.

RETURNS

- ≥ 0 index of split in which element is contained
- 1 given object does not belong to any split of that station

SEE ALSO

[ElmTrfstat.GetSplit\(\)](#)

GetSuppliedElements

Returns the network components that are supplied by the transformer (located in the station). A network component is considered to be supplied by a transformer if a topological path from the transformer to the component exists. A valid topological path in this sense is a path that starts at the transformer's HV side in direction of transformer (not in direction of HV connected node) and stops at

- network components that are out of calculation,
- network components that are not active (e.g. hidden or those of currently inactive grids),
- open switches,
- connections leading to a higher voltage level.

Generally, all network components of such a path are considered to be supplied by the transformer.

Exceptions are components that are out of calculation or in-active. Those components are never considered to be supplied by any transformer. A transformer is never considered to supply itself. Composite components such as ElmBranch, ElmSubstat, ElmTrfstat are considered to be supplied by a transformer if all energized components inside that composite are supplied by the transformer.

```
list ElmTrfstat.GetSuppliedElements([int inclNested])
```

ARGUMENTS

inclNested (optional)

- 0** Do not include components that are supplied by nested supplying stations
- 1** (default) Include components that are supplied by nested stations

SEE ALSO

[ElmTr2.GetSuppliedElements\(\)](#), [ElmTr3.GetSuppliedElements\(\)](#)

5.2.34 ElmVac

Overview

[GetGroundingImpedance](#)

GetGroundingImpedance

Returns the impedance of the internal grounding. Single phase voltage source connected to neutral are considered as grounding devices themselves; i.e. instead of the dedicated grounding parameters, the R1,X1 parameters are used.

```
[int valid,
float resistance,
float reactance ] ElmVac.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.

resistance (out)
Real part of the grounding impedance in Ohm.

reactance (out)
Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

5.2.35 ElmVoltreg

Overview

[CreateEvent](#)
[GetGroundingImpedance](#)
[GetZpu](#)
[NTap](#)

CreateEvent

For the corresponding voltage regulator, a Tap Event (EvtTap) is created for the simulation.

```
None ElmVoltreg.CreateEvent([int tapAction = 2,]
                             [float tapPos = 0]
                             )
```

ARGUMENTS

tapAction (optional)
0=increase tap; 1=decrease tap; 2=set tap to tapPos; 3>manual; 4=automatic

tapPos (optional)
Position of tap

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,
float resistance,
float reactance ] ElmVoltreg.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.

resistance (out)
Real part of the grounding impedance in Ohm.

reactance (out)
Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

GetZpu

Gets the impedance in p.u. of the voltage regulator for the specified tap position. If the tap position is out of the tap changer range, the respective min. or max. position will be used.

```
[float rpu,
float xpu ] ElmVltreg.GetZpu(int itappos,
                             int systembase)
```

ARGUMENTS

itappos Tap position

rpu (out) Resistance in p.u.

xpu (out) Reactance in p.u.

systembase

- 0** p.u. is based on rated power.
- 1** p.u. is based on system base (e.g. 100MVA).

NTap

Gets the voltage regulator tap position.

```
int ElmVltreg.NTap(int tap)
```

ARGUMENTS

tap 0=Tap 1, 1=Tap 2, 2=Tap 3

RETURNS

The tap position.

5.2.36 ElmXnet**Overview**

[CalcEfficiency](#)
[Disconnect](#)
[GetGroundingImpedance](#)
[GetStepupTransformer](#)
[Reconnect](#)

CalcEfficiency

Calculate efficiency for connected storage model at given active power value.

```
float ElmXnet.CalcEfficiency(float activePowerMW)
```

ARGUMENTS

activePowerMW
Active power value to calculate efficiency for.

RETURNS

Returns the resulting efficiency in p.u.

Disconnect

Disconnects a static generator by opening the first circuit breaker. The topological search performed to find such a breaker, stops at any busbar.

```
int ElmXnet.Disconnect()
```

RETURNS

- 0** breaker already open or successfully opened
- 1** an error occurred (no breaker found, open action not possible (earthing / RA))

GetGroundingImpedance

Returns the impedance of the internal grounding.

```
[int valid,  
float resistance,  
float reactance ] ElmXnet.GetGroundingImpedance(int busIdx)
```

ARGUMENTS

busIdx Bus index where the grounding should be determined.

resistance (out)
Real part of the grounding impedance in Ohm.

reactance (out)
Imaginary part of the grounding impedance in Ohm.

RETURNS

- 0** The values are invalid (e.g. because there is no internal grounding)
- 1** The values are valid.

GetStepupTransformer

Performs a topological search to find the step-up transformer of an external grid

```
DataObject ElmXnet.GetStepupTransformer(float voltage,  
                                         int swStatus  
                                         )
```

ARGUMENTS

- voltage* voltage level at which the search will stop
- swStatus* consideration of switch status. Possible values are:
- | | |
|----------|----------------------------|
| 0 | consider all switch status |
| 1 | ignore breaker status |
| 2 | ignore all switch status |

RETURNS

Returns the first collected step-up transformer object. It is empty if not found (e.g. start terminal already at hvVoltage).

Reconnect

Connects a static generator by closing all switches (breakers and isolators) up to the first breaker on the HV side of a transformer. The topological search to find all the switches, stops at any busbar.

```
int ElmXnet.Reconnect()
```

RETURNS

- | | |
|----------|---|
| 0 | the machine was successfully closed |
| 1 | a error occurred and the machine could not be connected to any busbar |

5.2.37 ElmZone**Overview**

[CalculateInterchangeTo](#)
[DefineBoundary](#)
[GetAll](#)
[GetBranches](#)
[GetBuses](#)
[GetObjs](#)
[SetLoadScaleAbsolute](#)

CalculateInterchangeTo

Calculates interchange power to the given zone (calculated quantities are: Pinter, Qinter, Pexport, Qexport, Pimport, Qimport). Prior the calculation the valid load flow calculation is required.

```
int ElmZone.CalculateInterchangeTo(DataObject zone)
```

ARGUMENTS

- zone* zone to which the interchange is calculated

RETURNS

- | | |
|---------------|--|
| < 0 | calculation error (i.e. no valid load flow, empty zone...) |
| 0 | no interchange power to the given zone |
| 1 | interchange power calculated |

DefineBoundary

Defines boundary with this zone as interior part. Resulting cubicles of boundary are busbar-oriented towards the zone.

```
DataObject ElmZone.DefineBoundary(int shift)
```

ARGUMENTS

shift Elements outside the zone that are within a distance of *shift* many elements to a boundary cubicle of the zone are added to the interior part of the resulting boundary

RETURNS

The defined boundary is returned in case of success. Otherwise NULL, if an error appeared in the definition of the boundary.

GetAll

Returns all objects which belong to this zone.

```
list ElmZone.GetAll()
```

RETURNS

The set of objects.

GetBranches

Returns all branches which belong to this zone.

```
list ElmZone.GetBranches()
```

RETURNS

The set of branch objects.

GetBuses

Returns all buses which belong to this zone.

```
list ElmZone.GetBuses()
```

RETURNS

The set of objects.

GetObjs

Returns all objects of the given class which belong to this zone.

```
list ElmZone.GetObjs(str classname)
```

ARGUMENTS

classname
name of the class (i.e. "ElmTr2")

RETURNS

The set of contained objects.

SetLoadScaleAbsolute

Readjusts zonal load scaling factor to the given active power. The zonal load scaling factor is the ratio of the given active power and the loads total actual power.

```
None ElmZone.SetLoadScaleAbsolute(float Pin)
```

ARGUMENTS

Pin active power in MW used for the load scaling factor.

5.3 Station Elements

5.3.1 StaCt

Overview

[SetPrimaryTap](#)

SetPrimaryTap

Determines the best matching primary tap for the connected branch, so that $I_{Nom,Branch} \cdot mltFactor \leq I_{pri}$. If no tap satisfies the equation, the largest tap is used.

```
int StaCt.SetPrimaryTap([float mltFactor])
```

ARGUMENTS

mltFactor (optional)
Multiplication factor (default 1.0)

RETURNS

0 Correctly set.
1 Error.

5.3.2 StaCubic

Overview

[GetAll](#)
[GetBranch](#)
[GetConnectedMajorNodes](#)
[GetConnections](#)
[GetNearestBusbars](#)
[GetPathToNearestBusbar](#)
[IsClosed](#)
[IsConnected](#)

GetAll

This function returns a set of network components that are collected by a topological traversal starting from this cubicle.

```
list StaCubic.GetAll([int direction = 1,]
                    [int ignoreOpenSwitches = 0]
                    )
```

ARGUMENTS

direction (optional)

Specifies the direction in which the network topology is traversed.

- 1** Traversal to the branch element (default).
- 0** Traversal to the terminal element.

ignoreOpenSwitches (optional)

Determines whether to pass open switches or to stop at them.

- 0** The traversal stops in a direction if an open switch is reached (default).
- 1** Ignore all switch statuses and pass every switch.

RETURNS

A set of network components that are collected by a topological traversal starting at the cubicle (StaCubic) where the function is called.

GetBranch

Function gets the branch of this cubicle.

```
DataObject StaCubic.GetBranch()
```

RETURNS

Branch object.

GetConnectedMajorNodes

This function returns all busbars being part of a split (inside a station) that can be reached by starting a topology search from the cubicle in direction of the branch element.

```
list StaCubic.GetConnectedMajorNodes ([float swtStat])
```

ARGUMENTS

swtStat

- 0 (default)** First perform a search that respects switch states (stopping at open switches). If no switches are found, an additional search with ignoring switch states.
- 1** Perform one search ignoring switch states (passing open and closed switches).
- 2** Search with respecting switch states. But do no additional search when switch is found.

RETURNS

A set of all busbars that can be reached starting a topology search from the cubicle in direction of the branch element.

GetConnections

Function gets all elements connected with this cubicle.

```
list StaCubic.GetConnections(int swtStat)
```

ARGUMENTS

swtStat Consider switch status (1) or not (0).

RETURNS

Set of elements.

GetNearestBusbars

Function searches for connected and connectable nearest busbars starting at the cubicle. Search stops at the nearest busbars and out of service elements. Internal and junction nodes, all types of branch elements and all types of switches - i.e. circuit breakers and disconnectors - are passed.

Connected busbars are all busbars which are topologically connected to the start cubicle without passing open switches. Connectable busbars are all busbars which are connectable to the start cubicle by closing switches. If the start cubicles terminal is a busbar then this busbar is not included in the result sets.

If more than one path exists between cubicle and a nearest busbar the relevant busbar is added only once to the result set.

```
[list connectedBusbars,  
list connectableBusbars] StaCubic.GetNearestBusbars(int searchDirection, [int excludeZPUs])
```

ARGUMENTS

connectedBusbars (out)

Found connected busbars.

connectableBusbars (out)

Found connectable busbars.

searchDirection

Direction of the search relative to the cubicle. Possible values are

- 0** search in all directions
- 1** search in direction of cubicles terminal
- 2** search towards connected branch element

excludeZPUs (optional)

Whether ZPU (ElmZpu) should be ignored in the search. Default=0.

GetPathToNearestBusbar

Function determines the path from the cubicle to the given busbar. The busbar must be connected or connectable to the start cubicle without passing additional busbars. If the given busbar is not a nearest busbar in relation to the cubicle an empty path is returned.

If more than one closed path exists between cubicle and busbar the elements of all these paths are combined.

```
list StaCubic.GetPathToNearestBusbar(DataObject nearestBusbar, [int excludeZPUs])
```

ARGUMENTS

nearestBusbar

Nearest busbar in relation to cubicle.

excludeZPUs (optional)

Whether ZPU (ElmZpu) should be ignored in the search. Default=0.

RETURNS

Net elements of the path from cubicle to busbar.

IsClosed

Function checks if this cubicle is directly connected with the busbar, considering the switch status.

```
int StaCubic.IsClosed()
```

RETURNS

- 0** Disconnected cubicle.
- 1** Connected cubicle.

IsConnected

Function checks if the cubicle is connected to the passed terminal or coupler.

```
int StaCubic.IsConnected(DataObject elm,
                        int          swtStat
                        )
```

ARGUMENTS

- elm* Terminal or coupler to check connection with.
- swtStat* Consider switch status (1) or not (0).

RETURNS

- 0** Not connected.
- 1** Connected.

5.3.3 StaExtbrkmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtbrkmea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for the switch position currently stored in the measurement object.

```
[int error,  
float value] StaExtbrkmea.GetMeaValue()
```

ARGUMENTS

value (out)
Value for switch status.

RETURNS

Return value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtbrkmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtbrkmea.GetStatuts()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtbrkmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtbrkmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtbrkmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtbrkmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtbrkmea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtbrkmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtbrkmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtbrkmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtbrkmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0 New status flags are applied in memory only

1 Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtbrkmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtbrkmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value for the switch position currently stored in the measurement object.

```
int StaExtbrkmea.SetMeaValue(int value)
```

ARGUMENTS

value New value for switch status.

RETURNS

Return value has no meaning. It is always 0.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtbrkmea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement
bit2 0x00000004 Disturbance
bit3 0x00000008 Protection
bit4 0x00000010 Marked suspect
bit5 0x00000020 Violated constraint
bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtbrkmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

- 0** New status flags are applied in memory only
- 1** Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtbrkmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtbrkmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

- 0** New status flags are applied in memory only
- 1** Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtbrkmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtbrkmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtbrkmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtbrkmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtbrkmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- | | |
|----------|---|
| 0 | on success |
| 1 | on error, e.g. target object does not have an attribute with given name |

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtbrkmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- 0** on success
- 1** on error, e.g. target object does not have an attribute with given name

5.3.4 StaExtcmdmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtcmdmea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for command interpreted as floating point value.

```
[int error,
float value] StaExtcmdmea.GetMeaValue()
```

ARGUMENTS

value (out)

Value obtained by parsing stored command string as floating point value.

RETURNS

Return value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtcmdmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtcmdmea.GetStatuts()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtcmdmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtcmdmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtcmdmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtcmdmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtcmdmea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtcmdmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtcmdmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtcmdmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtcmdmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0 New status flags are applied in memory only

1 Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtcmdmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtcmdmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtcmdmea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtcmdmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtcmdmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtcmdmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtcmdmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtcmdmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtcmdmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtcmdmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtcmdmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtcmdmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

5.3.5 StaExtdatmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[CreateEvent](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtdatmea.CopyExtMeaStatusToStatusTmp()
```

CreateEvent

Creates a “parameter change” event for controller object ('pCtrl') and attribute ('varName'). The event is stored in simulation event list and executed immediately.

```
None StaExtdatmea.CreateEvent()
```

GetMeaValue

Returns the value stored in the measurement object.

```
[int error,
float value] StaExtdatmea.GetMeaValue(int unused,
                                     int applyMultiplier)
```

ARGUMENTS

value (out)

Value, optionally modified by configured multiplier

unused

Not used.

applyMultiplier

If 1 (default), returned value will be modified by the multiplier stored in the measurement object (depending on Mode incremental/absolute). If 0, raw value will be returned.

RETURNS

Return value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtdatmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtdatmea.GetStatus()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtdatmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtdatmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtdatmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtdatmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtdatmea.IsStatusBitSet(int mask)
```

RETURNS

- | | |
|---|--|
| 0 | if at least one bit in mask is not set |
| 1 | if all bit(s) in mask are set |

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtdatmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtDatmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtDatmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtDatmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

- mask* Mask of bits to set to 0. A bit is unchanged if already unset before.
- dbSync (optional)*
 - 0** New status flags are applied in memory only
 - 1** Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtDatmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtDatmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

- mask* Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtDatmea.SetMeaValue(float value)
```

ARGUMENTS

- value* New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

- bit0 0x00000001** Manually entered data

bit1 0x00000002 Tele-Measurement
bit2 0x00000004 Disturbance
bit3 0x00000008 Protection
bit4 0x00000010 Marked suspect
bit5 0x00000020 Violated constraint
bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtDatmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

- 0** New status flags are applied in memory only
- 1** Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtDatmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtDatmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

- 0** New status flags are applied in memory only
- 1** Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtDatmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtDatmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtdatmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtdatmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtdatmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- | | |
|----------|---|
| 0 | on success |
| 1 | on error, e.g. target object does not have an attribute with given name |

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtdatmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- 0** on success
- 1** on error, e.g. target object does not have an attribute with given name

5.3.6 StaExtfmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtfmea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for frequency currently stored in the measurement object.

```
[int error,
float value] StaExtfmea.GetMeaValue()
```

ARGUMENTS

- value (out)*
Value for frequency.

RETURNS

- Return value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtfmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtfmea.GetStatuts()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtfmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtfmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtfmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtfmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtfmea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtfmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtfmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtfmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtfmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0 New status flags are applied in memory only

1 Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtfmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtfmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtfmea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtfmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtfmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtfmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtfmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtfmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtfmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtfmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtfmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtfmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

5.3.7 StaExtfuelmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtfuelmea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for fuel currently stored in the measurement object.

```
[int error,
float value] StaExtfuelmea.GetMeaValue(int unused,
                                         int applyMultiplier)
```

ARGUMENTS

value (out)

Value for fuel, optionally multiplied by configured multiplier

unused

Not used.

applyMultiplier

If 1 (default), returned value will be multiplied by the multiplier stored in the measurement object. If 0, raw value will be returned.

RETURNS

Return value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtfuelmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtfuelmea.GetStatus()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtfuelmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtfuelmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtfuelmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtfuelmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtfuelmea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtfuelmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtfuelmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtfuelmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtfuelmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtfuelmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtfuelmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtfuelmea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtfuelmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtfuelmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtfuelmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtfuelmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtfuelmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtfuelmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtfuelmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtfuelmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtfuelmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

5.3.8 StaExtimea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtimea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for current currently stored in the measurement object.

```
[int error,
float value] StaExtimea.GetMeaValue(int unused,
                                     int applyMultiplier)
```

ARGUMENTS

value (out)

Value for current, optionally multiplied by configured multiplier

unused Not used.

applyMultiplier

If 1 (default), returned value will be multiplied by the multiplier stored in the measurement object. If 0, raw value will be returned.

RETURNS

Return value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtimea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtimea.GetStatuts()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtimea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtimea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtimea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtimea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtimea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtimea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtimea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtimea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtimea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0 New status flags are applied in memory only

1 Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtimea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtimea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtimea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtimea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above
dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtimea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtimea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.
dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtimea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtimea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtimea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtimea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtimea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtimea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

5.3.9 StaExtpfmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtpfmea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for power factor currently stored in the measurement object.

```
[int error,
float value] StaExtpfmea.GetMeaValue(int unused,
                                     int applyMultiplier)
```

ARGUMENTS

value (out)

Value for current, optionally multiplied by configured multiplier

unused Not used.

applyMultiplier

If 1 (default), returned value will be multiplied by the multiplier stored in the measurement object. If 0, raw value will be returned.

RETURNS

Return value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtpfmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtpfmea.GetStatuts()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtpfmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtpfmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtpfmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtpfmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtpfmea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtpfmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtpfmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtpfmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtpfmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0 New status flags are applied in memory only

1 Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtpfmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtpfmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtpfmea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtpfmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtpfmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtpfmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtpfmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtpfmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtpfmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtpfmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtpfmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtpfmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

5.3.10 StaExtpmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtpmea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for active power stored in the measurement object.

```
[int error,
float value] StaExtpmea.GetMeaValue(int unused,
                                     int applyMultiplier)
```

ARGUMENTS

value (out)

Value for active power, optionally multiplied by configured multiplier

unused Not used.

applyMultiplier

If 1 (default), returned value will be multiplied by the multiplier stored in the measurement object. If 0, raw value will be returned.

RETURNS

Return value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtpmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtpmea.GetStatuts()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtpmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtpmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtpmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtpmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtpmea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtpmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtpmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtpmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtpmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0 New status flags are applied in memory only

1 Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtpmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtpmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtpmea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtPmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtPmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtPmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtPmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtPmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtpmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtpmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtpmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- | | |
|----------|---|
| 0 | on success |
| 1 | on error, e.g. target object does not have an attribute with given name |

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtpmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- | | |
|----------|---|
| 0 | on success |
| 1 | on error, e.g. target object does not have an attribute with given name |

5.3.11 StaExtqmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtqmea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for reactive power currently stored in the measurement object.

```
[int error,
float value] StaExtqmea.GetMeaValue(int unused,
                                     int applyMultiplier)
```

ARGUMENTS

value (out)

Value for reactive power, optionally multiplied by configured multiplier

unused Not used.

applyMultiplier

If 1 (default), returned value will be multiplied by the multiplier stored in the measurement object. If 0, raw value will be returned.

RETURNS

Return value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtqmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtqmea.GetStatuts()
```


RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtqmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtqmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtqmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtqmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtqmea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtqmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtqmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtqmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtqmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0 New status flags are applied in memory only

1 Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtqmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtqmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtqmea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtqmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtqmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtqmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtqmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtqmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtqmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtqmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtqmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtqmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

5.3.12 StaExtsmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtsmea.CopyExtMeaStatusToStatusTmp()
```

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtsmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtsmea.GetStatuts()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtsmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtsmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtsmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtsmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtsmea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtsmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtsmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtsmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtsmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

- 0** New status flags are applied in memory only
- 1** Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtsmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtsmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtsmea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event

bit7 0x00000080 Event Block.

bit8 0x00000100 Alarm Block.

bit9 0x00000200 Update Block.

bit10 0x00000400 Control Block.

bit29 0x20000000 Read

bit30 0x40000000 Write

bit31 0x80000000 Neglected by SE

```
None StaExtsmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

- 0** New status flags are applied in memory only
- 1** Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtsmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtsmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

- 0** New status flags are applied in memory only
- 1** Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtsmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtsmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtsmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtsmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtsmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- | | |
|----------|---|
| 0 | on success |
| 1 | on error, e.g. target object does not have an attribute with given name |

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtsmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- | | |
|----------|---|
| 0 | on success |
| 1 | on error, e.g. target object does not have an attribute with given name |

5.3.13 StaExttapmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)

[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExttapmea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for tap position and tap info currently stored in the measurement object.

```
[int error,
float value] StaExttapmea.GetMeaValue(int unused,
                                     int applyMultiplier)
```

ARGUMENTS

value (out)

Value.

type

type of value to return

- 0** tap position
- 1** operation mode
- 2** tap changer command
- 3** tap operation mode
- 4** tap operation mode command

useTranslationTable

Only supported if type=0 (tap step), if 1 (default) returned value will be translated according to given table. If 0 is passed, the raw value will be returned.

RETURNS

- 0** on success
- 1** on error, e.g. unsupported type

Returns 1 on errorReturn value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExttapmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExttapmea.GetStatuts()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExttapmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExttapmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExttapmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExttapmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExttapmea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExttapmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExttapmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExttapmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExttapmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0 New status flags are applied in memory only

1 Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExttapmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExttapmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExttapmea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event

bit7 0x00000080 Event Block.

bit8 0x00000100 Alarm Block.

bit9 0x00000200 Update Block.

bit10 0x00000400 Control Block.

bit29 0x20000000 Read

bit30 0x40000000 Write

bit31 0x80000000 Neglected by SE

```
None StaExttapmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

- | | |
|----------|---|
| 0 | New status flags are applied in memory only |
| 1 | Default, new status flags are stored on db (persistent) |

SetStatusBit

Sets specific bits in the status bitfield. See [StaExttapmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExttapmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

- | | |
|----------|---|
| 0 | New status flags are applied in memory only |
| 1 | Default, new status flags are stored on db (persistent) |

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExttapmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExttapmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExttapmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExttapmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExttapmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- | | |
|----------|---|
| 0 | on success |
| 1 | on error, e.g. target object does not have an attribute with given name |

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExttapmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- | | |
|----------|---|
| 0 | on success |
| 1 | on error, e.g. target object does not have an attribute with given name |

5.3.14 StaExtv3mea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtv3mea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for voltage currently stored in the measurement object.

```
[int error,
float value] StaExtv3mea.GetMeaValue(int unused,
                                     int applyMultiplier)
```

ARGUMENTS

value (out)

Value for voltage, optionally multiplied by configured multiplier

phase

Index of desired phase. Index must be 0, 1 or 2.

applyMultiplier

If 1 (default), returned value will be multiplied by the multiplier stored in the measurement object. If 0, raw value will be returned.

RETURNS

- 0** on success
- 1** on error, e.g. phase index does not exist

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtv3mea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtv3mea.GetStatuts()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtv3mea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtv3mea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtv3mea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtv3mea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtv3mea.IsStatusBitSet(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtv3mea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtv3mea.IsStatusBitSetTmp(int mask)
```

RETURNS

- 0** if at least one bit in mask is not set
- 1** if all bit(s) in mask are set

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtv3mea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtv3mea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0 New status flags are applied in memory only

1 Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtv3mea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtv3mea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtv3mea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtv3mea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above
dbSync (optional)
 0 New status flags are applied in memory only
 1 Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtv3mea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtv3mea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.
dbSync (optional)
 0 New status flags are applied in memory only
 1 Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtv3mea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtv3mea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtv3mea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtv3mea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtv3mea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- | | |
|----------|---|
| 0 | on success |
| 1 | on error, e.g. target object does not have an attribute with given name |

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtv3mea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

- | | |
|----------|--|
| 0 | Value is copied in memory only |
| 1 | Default, copied value is stored on db (persistent) |

RETURNS

- | | |
|----------|---|
| 0 | on success |
| 1 | on error, e.g. target object does not have an attribute with given name |

5.3.15 StaExtvmea

Overview

[CopyExtMeaStatusToStatusTmp](#)
[GetMeaValue](#)
[GetStatus](#)
[GetStatusTmp](#)
[InitTmp](#)
[IsStatusBitSet](#)
[IsStatusBitSetTmp](#)
[ResetStatusBit](#)
[ResetStatusBitTmp](#)
[SetMeaValue](#)
[SetStatus](#)
[SetStatusBit](#)
[SetStatusBitTmp](#)
[SetStatusTmp](#)
[UpdateControl](#)
[UpdateCtrl](#)

CopyExtMeaStatusToStatusTmp

Copies the (persistent) status of current measurement object to temporary (in memory) status.

```
None StaExtvmea.CopyExtMeaStatusToStatusTmp()
```

GetMeaValue

Returns the value for voltage currently stored in the measurement object.

```
[int error,  
float value] StaExtvmea.GetMeaValue(int unused,  
                                     int applyMultiplier)
```

ARGUMENTS

value (out)

Value for voltage, optionally multiplied by configured multiplier

unused Not used.

applyMultiplier

If 1 (default), returned value will be multiplied by the multiplier stored in the measurement object. If 0, raw value will be returned.

RETURNS

Return value has no meaning. It is always 0.

GetStatus

Returns the status flags. Please note, this value is interpreted as a bitfield. See [StaExtvmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtvmea.GetStatuts()
```

RETURNS

Status bitfield as an integer value.

GetStatusTmp

Returns the temporary (in memory) status flags. Please note, this value is interpreted as a bitfield. See [StaExtvmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtvmea.GetStatusTmp()
```

RETURNS

Status bitfield as an integer value.

InitTmp

Initialises the temporary (in memory) fields of the measurement object with the values currently stored in the corresponding persistent fields. This affects temporary measurement value and temporary status fields. The temporary measurement value is used internally for comparison of new and old values for deadband violation. The temporary status is used during calculation in order to not modify initial value.

This function should be called once after the link has been established and before the calculation loop is executed.

```
None StaExtvmea.InitTmp()
```

IsStatusBitSet

Checks if specific bit(s) are set in the status bitfield. See [StaExtvmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtvmea.IsStatusBitSet(int mask)
```

RETURNS

- | | |
|----------|--|
| 0 | if at least one bit in mask is not set |
| 1 | if all bit(s) in mask are set |

IsStatusBitSetTmp

Checks if specific bit(s) are set in the temporary (in memory) status bitfield. See [StaExtvmea.SetStatus\(\)](#) for details on the status bits.

```
int StaExtvmea.IsStatusBitSetTmp(int mask)
```

RETURNS

- | | |
|----------|--|
| 0 | if at least one bit in mask is not set |
| 1 | if all bit(s) in mask are set |

ResetStatusBit

Resets specific bits in the status bitfield. See [StaExtvmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtvmea.ResetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

dbSync (optional)

0 New status flags are applied in memory only

1 Default, new status flags are stored on db (persistent)

ResetStatusBitTmp

Resets specific bits in the temporary (in memory) status bitfield. See [StaExtvmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtvmea.ResetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 0. A bit is unchanged if already unset before.

SetMeaValue

Sets the value stored in the measurement object.

```
int StaExtvmea.SetMeaValue(float value)
```

ARGUMENTS

value New value.

RETURNS

Return value has no meaning. It is always 0.

SetStatus

Sets the status flags of the measurement object. Please note, this value is interpreted as a bitfield where the bits have the following meaning. An option is considered enabled if the corresponding bit is set to 1.

bit0 0x00000001 Manually entered data

bit1 0x00000002 Tele-Measurement

bit2 0x00000004 Disturbance

bit3 0x00000008 Protection

bit4 0x00000010 Marked suspect

bit5 0x00000020 Violated constraint

bit6 0x00000040 On Event
bit7 0x00000080 Event Block.
bit8 0x00000100 Alarm Block.
bit9 0x00000200 Update Block.
bit10 0x00000400 Control Block.
bit29 0x20000000 Read
bit30 0x40000000 Write
bit31 0x80000000 Neglected by SE

```
None StaExtvmea.SetStatus(int status, int dbSync)
```

ARGUMENTS

status Bitfield for status flags, see above

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBit

Sets specific bits in the status bitfield. See [StaExtvmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtvmea.SetStatusBit(int mask, int dbSync)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

dbSync (optional)

0	New status flags are applied in memory only
1	Default, new status flags are stored on db (persistent)

SetStatusBitTmp

Sets specific bits in the temporary (in memory) status bitfield. See [StaExtvmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtvmea.SetStatusBitTmp(int mask)
```

ARGUMENTS

mask Mask of bits to set to 1. A bit is unchanged if already set before.

SetStatusTmp

Sets the temporary (in memory) status flags of the measurement object. This temporary value is used during calculations so that changes do not lead to object modifications and initial value remains unchanged.

Please note, this value is interpreted as a bitfield. See [StaExtvmea.SetStatus\(\)](#) for details on the status bits.

```
None StaExtvmea.SetStatusTmp(int status)
```

ARGUMENTS

status Bitfield for status flags, see above

UpdateControl

Transfers the value of current measurement object to the controller object (target object 'pCtrl' and target attribute 'varName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtvmea.UpdateControl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

UpdateCtrl

Transfers the value of current measurement object to the controlled object (target object 'pObject' and target attribute 'variabName'). If target object is a command, it is automatically executed afterwards.

Note: Calculation results will not be reset by this value transfer.

```
int StaExtvmea.UpdateCtrl(int dbSync)
```

ARGUMENTS

dbSync (optional)

0	Value is copied in memory only
1	Default, copied value is stored on db (persistent)

RETURNS

0	on success
1	on error, e.g. target object does not have an attribute with given name

5.3.16 StaSwitch

Overview

[Close](#)
[IsClosed](#)
[IsOpen](#)
[Open](#)

Close

Closes the switch by changing its status to 'close'. This action will fail if the status is currently determined by an active running arrangement.

```
int StaSwitch.Close()
```

RETURNS

0	On success
≠ 0	On error

SEE ALSO

[StaSwitch.Open\(\)](#)

IsClosed

Returns information about current switch state.

```
int StaSwitch.IsClosed()
```

RETURNS

1	switch is closed
0	switch is open

SEE ALSO

[StaSwitch.IsOpen\(\)](#)

IsOpen

Returns information about current switch state.

```
int StaSwitch.IsOpen()
```

RETURNS

1	switch is open
0	switch is closed

SEE ALSO

[StaSwitch.IsClosed\(\)](#)

Open

Opens the switch by changing its status to 'open'. This action will fail if the status is currently determined by an active running arrangement.

```
int StaSwitch.Open()
```

RETURNS

0 On success
≠ 0 On error

SEE ALSO

[StaSwitch.Close\(\)](#)

5.4 Commands

Overview

[Execute](#)

Execute

Executes the command.

```
int Com*.Execute()
```

5.4.1 ComAddlabel

Overview

[Execute](#)

Execute

This function executes the Add Statistic Labels command itself for a given plot and curve.

```
int ComAddlabel.Execute(DataObject plot, int curveIndex)
```

ARGUMENTS

plot The plot to modify.

curveIndex
 The index of the curve inside the plot's table. The index is zero based, therefore the index of the first curve is 0.

RETURNS

0 The function executed without any errors.
1 The plot is visible on a single line graphic only.
2 The parameter plot is None.
3 The parameter plot is not a virtual instrument (classname should start with Vis).
4 The object plot was found in any open graphic.

- 5 The object plot is not a diagram.
- 6 An internal error occurred (plot is incomplete).

5.4.2 ComAddon

Overview

[CreateModule](#)
[DefineDouble](#)
[DefineDoubleMatrix](#)
[DefineDoublePerConnection](#)
[DefineDoubleVector](#)
[DefineDoubleVectorPerConnection](#)
[DefineInteger](#)
[DefineIntegerPerConnection](#)
[DefineIntegerVector](#)
[DefineIntegerVectorPerConnection](#)
[DefineObject](#)
[DefineObjectPerConnection](#)
[DefineObjectVector](#)
[DefineObjectVectorPerConnection](#)
[DefineString](#)
[DefineStringPerConnection](#)
[DeleteModule](#)
[FinaliseModule](#)
[GetActiveModule](#)
[ModuleExists](#)
[SetActiveModule](#)

CreateModule

Creates the calculation module of this AddOn. Volatile object parameters are created for all variable definitions stored inside this command. They are accessible like any other built in object parameter.

```
int ComAddon.CreateModule()
```

RETURNS

- 0 Ok, module was created.
- 1 An error occurred.

SEE ALSO

[ComAddon.FinaliseModule\(\)](#) [ComAddon.DeleteModule\(\)](#)

DefineDouble

Creates a new floating-point-number parameter for the given type of objects.

```
int ComAddon.DefineDouble(str class,
                           str name,
                           str desc,
                           str unit,
                           float initial)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. <code>ElmLne</code> for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>unit</i>	Parameter's unit
<i>initial</i>	Default value of new parameter

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.

SEE ALSO

[ComAddon.DefineDoublePerConnection\(\)](#)

DefineDoubleMatrix

Creates a new floating-point-matrix parameter for the given type of objects.

```
int ComAddon.DefineDoubleMatrix(string class,
                                str name,
                                str desc,
                                str unit,
                                float initial,
                                int rows,
                                int columns)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. <code>ElmLne</code> for the line.
<i>name</i>	Name of the new parameter.
<i>desc</i>	Parameter description.
<i>unit</i>	Parameter's unit.
<i>initial</i>	Default value for all entries of new parameter.
<i>rows</i>	Number of initial rows. Number of rows will be 0 if a value smaller than 0 is given.
<i>columns</i>	Number of initial columns. Number of columns will be 0 if a value smaller than 0 is given.

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.

DefineDoublePerConnection

Creates a new floating-point-number parameter for every connection for the given type of objects.

```
int ComAddon.DefineDoublePerConnection(str class,  
                                       str name,  
                                       str desc,  
                                       str unit,  
                                       float initial)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. ElmLne for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>unit</i>	Parameter's unit
<i>initial</i>	Default value of new parameter

RETURNS

- 0** Ok, Parameter was created.
- Other than 0** An error occurred, possible reasons:
- The module of this add on does not exist.
 - An object with the given class does not exist in PowerFactory.
 - The parameter name for the given class already exists in the module.
 - The elements of class are not branch elements. Therefore there is no connection count. DefineDouble shall be used instead.

SEE ALSO

[ComAddon.DefineDouble\(\)](#)

DefineDoubleVector

Creates a new floating-point-number vector parameter for the given type of objects.

```
int ComAddon.DefineDoubleVector(str class,  
                                str name,  
                                str desc,  
                                str unit,  
                                float initial,  
                                int size)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. ElmLne for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>unit</i>	Parameter's unit
<i>initial</i>	Default value of new parameter
<i>size</i>	Initial size of vector. Size will be 0 if a value smaller than 0 is given.

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.

SEE ALSO

[ComAddon.DefineDouble\(\)](#) [ComAddon.DefineDoublePerConnection\(\)](#)

DefineDoubleVectorPerConnection

Creates a new floating-point-number vector parameter for the given type of objects for every connection of the object.

```
int ComAddon.DefineDoubleVectorPerConnection(str class,
                                             str name,
                                             str desc,
                                             str unit,
                                             float initial,
                                             int size)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. Elmlne for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>unit</i>	Parameter's unit
<i>initial</i>	Default value of new parameter
<i>size</i>	Initial size of vector. Size will be 0 if a value smaller than 0 is given.

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.
- The elements of class are not branch elements. Therefore there is no connection count. DefineDoubleVector shall be used instead.

SEE ALSO

[ComAddon.DefineDoubleVector\(\)](#)

DefineInteger

Creates a new integer parameter for the given type of objects.

```
int ComAddon.DefineInteger(str class,
                           str name,
                           str desc,
                           str unit,
                           int initial)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. ElmLne for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>unit</i>	Parameter's unit
<i>initial</i>	Default value of new parameter

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.

SEE ALSO

[ComAddon.DefineIntegerPerConnection\(\)](#)

DefineIntegerPerConnection

Creates a new integer parameter for every connection for the given type of objects.

```
int ComAddon.DefineIntegerPerConnection(str class,
                                         str name,
                                         str desc,
                                         str unit,
                                         int initial)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. ElmLne for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>unit</i>	Parameter's unit
<i>initial</i>	Default value of new parameter

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.
- The elements of class are not branch elements. Therefore there is no connection count. DefineInteger shall be used instead.

SEE ALSO

[ComAddon.DefineInteger\(\)](#)

DefineIntegerVector

Creates a new integer vector parameter for the given type of objects.

```
int ComAddon.DefineIntegerVector(str class,
                                str name,
                                str desc,
                                str unit,
                                int initial,
                                int size)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. ElmLne for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>unit</i>	Parameter's unit
<i>initial</i>	Default value of new parameter
<i>size</i>	Initial size of vector. Size will be 0 if a value smaller than 0 is given.

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.

SEE ALSO

[ComAddon.DefineInteger\(\)](#) [ComAddon.DefineIntegerPerConnection\(\)](#)

DefineIntegerVectorPerConnection

Creates a new integer vector parameter for the given type of objects for every connection of the object.

```
int ComAddon.DefineIntegerVectorPerConnection(str class,
                                                str name,
                                                str desc,
                                                str unit,
                                                int initial,
                                                int size)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. ElmLne for the line.
<i>name</i>	Name of the new parameter

<i>desc</i>	Parameter description
<i>unit</i>	Parameter's unit
<i>initial</i>	Default value of new parameter
<i>size</i>	Initial size of vector. Size will be 0 if a value smaller than 0 is given.

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.
- The elements of class are not branch elements. Therefore there is no connection count. DefineIntegerVector shall be used instead.

SEE ALSO

[ComAddon.DefineIntegerVector\(\)](#)

DefineObject

Creates a new object parameter for the given type of objects.

```
int ComAddon.DefineObject(str class,
                        str name,
                        str desc,
                        DataObject initial)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. Elmlne for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>initial</i>	Default object of new parameter

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.

SEE ALSO

[ComAddon.DefineObjectPerConnection\(\)](#)

DefineObjectPerConnection

Creates a new object parameter for every connection for the given type of objects.

```
int ComAddon.DefineObjectPerConnection(str class,  
                                       str name,  
                                       str desc,  
                                       DataObject initial)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. ElmLne for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>initial</i>	Default value of new parameter

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.
- The elements of class are not branch elements. Therefore there is no connection count. DefineObject shall be used instead.

SEE ALSO

[ComAddon.DefineObject\(\)](#)

DefineObjectVector

Creates a new object vector parameter for the given type of objects.

```
int ComAddon.DefineObjectVector(str class,  
                                str name,  
                                str desc,  
                                DataObject initial,  
                                int size)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. ElmLne for the line.
<i>name</i>	Name of the new parameter.
<i>desc</i>	Parameter description.
<i>initial</i>	Default value of new parameter.
<i>size</i>	Initial size of vector. Size will be 0 if a value smaller than 0 is given.

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.

SEE ALSO

[ComAddon.DefineObject\(\)](#) [ComAddon.DefineObjectPerConnection\(\)](#)

DefineObjectVectorPerConnection

Creates a new object vector parameter for the given type of objects for every connection of the object.

```
int ComAddon.DefineObjectVectorPerConnection(str class,
                                             str name,
                                             str desc,
                                             DataObject initial,
                                             int size)
```

ARGUMENTS

class The type of objects for which the new parameter is to be created, e.g. Elmlne for the line.

name Name of the new parameter.

desc Parameter description.

initial Default value of new parameter.

size Initial size of vector. Size will be 0 if a value smaller than 0 is given.

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.
- The elements of class are not branch elements. Therefore there is no connection count. DefineObjectVector shall be used instead.

SEE ALSO

[ComAddon.DefineObjectVector\(\)](#)

DefineString

Creates a new text parameter for the given type of objects.

```
int ComAddon.DefineString(str class,
                          str name,
                          str desc,
                          str unit,
                          str initial)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. <code>ElmLne</code> for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>unit</i>	Parameter's unit
<i>initial</i>	Default value of new parameter

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.

SEE ALSO

[ComAddon.DefineStringPerConnection\(\)](#)

DefineStringPerConnection

Creates a new text parameter for every connection for the given type of objects.

```
int ComAddon.DefineStringPerConnection(str class,
                                       str name,
                                       str desc,
                                       str unit,
                                       str initial)
```

ARGUMENTS

<i>class</i>	The type of objects for which the new parameter is to be created, e.g. <code>ElmLne</code> for the line.
<i>name</i>	Name of the new parameter
<i>desc</i>	Parameter description
<i>unit</i>	Parameter's unit
<i>initial</i>	Default value of new parameter

RETURNS

0 Ok, Parameter was created.

Other than 0 An error occurred, possible reasons:

- The module of this add on does not exist.
- An object with the given class does not exist in PowerFactory.
- The parameter name for the given class already exists in the module.
- The elements of class are not branch elements. Therefore there is no connection count. `DefineString` shall be used instead.

SEE ALSO

[ComAddon.DefineString\(\)](#)

DeleteModule

Deletes the module of this add on.

```
int ComAddon.DeleteModule()
```

- 0** Success. The module is deleted completely.
- 1** Failure. The module does not exist and can therefore not be deleted.

SEE ALSO

[ComAddon.CreateModule\(\)](#)

FinaliseModule

Finalises a user defined module which was created using the method CreateModule. All user defined variables defined for this module are read-only after the call of finalise module. The module is the one being used in the flexible data, single line graphic text boxes and colouring. It can be reset like any other built-in calculation using the reset button.

```
int ComAddon.FinaliseModule()
```

RETURNS

- 0** Ok, module was finalised.
- 1** An error occurred, this command is not the one being currently active.

SEE ALSO

[ComAddon.CreateModule\(\)](#)

GetActiveModule

Gets the key of the module being currently active. An empty string is returned if there is no active module.

```
str ComAddon.GetActiveModule()
```

RETURNS

The key of the active module. an empty string is returned if there is no active module.

SEE ALSO

[ComAddon.SetActiveModule\(\)](#)

ModuleExists

Checks if the module for this add-on was already created using the method CreateModule.

```
int ComAddon.ModuleExists()
```

RETURNS

- 0** The module was not created yet.
- 1** The module was already created.

SEE ALSO

[ComAddon.CreateModule\(\)](#) [ComAddon.FinaliseModule\(\)](#) [ComAddon.DeleteModule\(\)](#)

SetActiveModule

Set this module as active module. This method is required only if several modules are created concurrently. In case that only one module is being used, there is no need to use this method, because CreateModule sets the created module automatically as active module.

```
int ComAddon.SetActiveModule()
```

RETURNS

- 0** Success. This command is set as active module.
- 1** Failure. This command is already the active module.

SEE ALSO

[ComAddon.CreateModule\(\)](#) [ComAddon.FinaliseModule\(\)](#) [ComAddon.DeleteModule\(\)](#)

5.4.3 ComAmpacity**Overview**

[ExecuteAmpacityCalc](#)

ExecuteAmpacityCalc

The function executes ampacity calculation with or without the tabular report at the end.

```
int ComAmpacity.ExecuteAmpacityCalc([int ireport = 1])
```

ARGUMENTS

ireport (optional)
Show report or not, default = 1.

5.4.4 ComAuditlog**Overview**

[Check](#)

Check

Checks integrity of Audit Log.

```
int ComAuditlog.Check()
```

RETURNS

number of errors

5.4.5 ComBoundary

Overview

[GetCreatedBoundaries](#)

GetCreatedBoundaries

Gets created boundaries after the command has been executed.

```
list ComBoundary.GetCreatedBoundaries()
```

RETURNS

Container of created boundaries.

5.4.6 ComCapo

Overview

[ConnectShuntToBus](#)

[LossCostAtBusTech](#)

[TotalLossCost](#)

ConnectShuntToBus

Connects the equivalent shunt in the specified terminal and executes the load flow command. The shunt is not physically added in the database, just the susceptance is added for the calculation.

```
int ComCapo.ConnectShuntToBus(DataObject terminal,
                               float phtech,
                               float Q
                              )
```

ARGUMENTS

terminal The terminal to which the shunt will be connected

phtech Phase technology. Possible values are

0	three-phase
1	ph-ph a-b
2	ph-ph b-c
3	ph-ph a-c
4	ph-e a
5	ph-e b
6	ph-e c

Note: In balanced load flow, the technology will always be three-phase.

Q Reactive power value in Mvar

RETURNS

- 0** On success.
- 1** An error occurred during load flow execution.

LossCostAtBusTech

Returns the losses cost of the selected terminal and configuration calculated during the sensitivity analysis or the optimization.

```
float ComCapo.LossCostAtBusTech(DataObject terminal,
                                float phtech
                                )
```

ARGUMENTS

- terminal* Specified bus
- phtech* Phase technology. Possible values are
- 0** three-phase
 - 1** ph-ph a-b
 - 2** ph-ph b-c
 - 3** ph-ph a-c
 - 4** ph-e a
 - 5** ph-e b
 - 6** ph-e c

RETURNS

Returns the losses cost

TotalLossCost

Returns the total cost calculated after the sensitivity analysis or the optimization.

```
float ComCapo.TotalLossCost([int iopt])
```

ARGUMENTS

- iopt (optional)*
Type of cost. Possible values are
- 0** Losses in MW (default)
 - 1** Cost of losses
 - 2** Cost of voltage violations
 - 3** Cost of shunts

RETURNS

Returns losses in MW or cost value.

5.4.7 ComCimdbexp

Overview

[Execute](#)

Execute

Executes the export. In case of a validation error the export is not performed.

```
int ComCimdbexp.Execute([int validate])
```

ARGUMENTS

validate (optional)

Option to execute CIM Data Validation before export. If not provided, the validation is executed. Possible values are

0	Do not validate
1	Validate

RETURNS

0	OK
1	Error: export failed

5.4.8 ComCimdbimp

Overview

[Execute](#)

[ImportAndConvert](#)

Execute

Executes the import.

```
int ComCimdbimp.Execute([int validate])
```

ARGUMENTS

validate (optional)

Option to execute CIM Data Validation after import. If not provided, the validation is executed. Possible values are

0	Do not validate
1	Validate

RETURNS

0	OK
1	Error: import failed

ImportAndConvert

Imports CIM data from file path provided in the 'CIM Data Import' command without storing CIM data into database, and converts CIM to Grid using the 'CIM to Grid Conversion' command object provided in the function call as template. The CIM to Grid Conversion will be executed with default settings if the command object is not provided in the function call.

```
int ComCimdbimp.ImportAndConvert([int validate],
                                [DataObject cimToGrid])
```

ARGUMENTS

validate (optional)

Option to execute CIM Data Validation after import. If not provided, the validation is executed. Possible values are

- | | |
|----------|-----------------|
| 0 | Do not validate |
| 1 | Validate |

cimToGrid (optional)

ComCimtoGrid object with preconfigured settings for converting imported CIM data. If not provided, the default CIM to Grid Conversion settings will be used.

RETURNS

- | | |
|----------|--------------------------|
| 0 | OK |
| 1 | Error: import failed |
| 2 | Error: conversion failed |

5.4.9 ComCimvalidate

Overview

[Execute](#)
[GetClassType](#)
[GetDescriptionText](#)
[GetInputObject](#)
[GetModel](#)
[GetModelId](#)
[GetNumberOfValidationMessages](#)
[GetObject](#)
[GetObjectId](#)
[GetProfile](#)
[GetSeverity](#)
[GetType](#)

Execute

Executes the validation. Creates no validation report if no errors, or warnings were found.

```
int ComCimvalidate.Execute([int openReport])
```

ARGUMENTS

openReport (optional)

Option to open report after validation. If not provided, the report is opened. Possible values are

0	Do not open report
1	Open report

RETURNS

0	OK
1	Error: validation failed

GetClassType

Returns the object class type from the selected validation message.

```
str ComCimvalidate.GetClassType(int messageIndex)
```

ARGUMENTS

messageIndex
Index of the validation message.

RETURNS

Object class type.

GetDescriptionText

Returns the description from the selected validation message.

```
str ComCimvalidate.GetDescriptionText(int messageIndex)
```

ARGUMENTS

messageIndex
Index of the validation message.

RETURNS

Message description.

GetInputObject

Returns the object selected for validation.

```
DataObject ComCimvalidate.GetInputObject()
```

RETURNS

Pointer to *CimArchive*, *CimModel*, or *SetSelect*.

GetModel

Returns the *CimModel* object from the selected validation message.

```
DataObject ComCimvalidate.GetModel(int messageIndex)
```

ARGUMENTS

messageIndex
Index of the validation message.

RETURNS

Pointer to *CimModel*.

GetModelId

Returns the model ID from the selected validation message.

```
str ComCimvalidate.GetModelId(int messageIndex)
```

ARGUMENTS

messageIndex
Index of the validation message.

RETURNS

Model ID.

GetNumberOfValidationMessages

Returns the number of validation messages generated.

```
int ComCimvalidate.GetNumberOfValidationMessages()
```

RETURNS

Number of validation messages.

GetObject

Returns the *CimObject* object from the selected validation message.

```
DataObject ComCimvalidate.GetObject(int messageIndex)
```

ARGUMENTS

messageIndex
Index of the validation message.

RETURNS

Pointer to *CimObject*.

GetObjectId

Returns the object ID from the selected validation message.

```
str ComCimvalidate.GetObjectId(int messageIndex)
```

ARGUMENTS

messageIndex
Index of the validation message.

RETURNS

Object ID.

GetProfile

Returns the model profile from the selected validation message.

```
str ComCimvalidate.GetProfile(int messageIndex)
```

ARGUMENTS

messageIndex
Index of the validation message.

RETURNS

Model profile.

GetSeverity

Returns the severity of the selected validation message.

```
str ComCimvalidate.GetSeverity(int messageIndex)
```

ARGUMENTS

messageIndex
Index of the validation message.

RETURNS

Message severity.

GetType

Returns the type of the selected validation message.

```
str ComCimvalidate.GetType(int messageIndex)
```

ARGUMENTS

messageIndex
Index of the validation message.

RETURNS

Message type.

5.4.10 ComConreq**Overview**

[Execute](#)

Execute

Performs a Connection Request Assessment according to the selected method. Results are provided for connection request elements in the single line graphic, and are summarised in a report in the output window.

```
int ComConreq.Execute()
```

RETURNS

- | | |
|----------|--|
| 0 | OK |
| 1 | Error: calculation function |
| 2 | Error: settings/initialisation/load flow |

5.4.11 ComContingency

Overview

[ContinueTrace](#)
[CreateRecoveryInformation](#)
[GetGeneratorEvent](#)
[GetInterruptedPowerAndCustomersForStage](#)
[GetInterruptedPowerAndCustomersForTimeStep](#)
[GetLoadEvent](#)
[GetNumberOfGeneratorEventsForTimeStep](#)
[GetNumberOfLoadEventsForTimeStep](#)
[GetNumberOfSwitchEventsForTimeStep](#)
[GetNumberOfTimeSteps](#)
[GetObj](#)
[GetSwitchEvent](#)
[GetTimeOfStepInSeconds](#)
[GetTotalInterruptedPower](#)
[JumpToLastStep](#)
[RemoveEvents](#)
[StartTrace](#)
[StopTrace](#)

ContinueTrace

Continues trace execution for this contingency.

```
int ComContingency.ContinueTrace()
```

RETURNS

- | | |
|----------|-------------|
| 0 | On success. |
| 1 | On error. |

CreateRecoveryInformation

Creates recovery information for a contingency. The recovery information can later be retrieved e.g. via [ComContingency.GetInterruptedPowerAndCustomersForStage\(\)](#).

Can only save one contingency at the same time.

```
int ComContingency.CreateRecoveryInformation(DataObject resultFileInput)
```

ARGUMENTS

resultFileInput

Read from this result file.

RETURNS

- 0** On success.
- 1** On error.

GetGeneratorEvent

Gets generator event of a certain time step during recovery.

[ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
[ DataObject generator,
float changedP,
float changedQ ) ]
ComContingency.GetGeneratorEvent(int currentTimeStep,
int loadEvent)
```

ARGUMENTS

*currentTimeStep*Input: Number of time steps are get via [ComContingency.GetNumberOfTimeSteps\(\)](#)*switchEvent*Input: Number of generator events for a certain time step are get via [ComContingency.GetNumberOfSwitchEventsForTimeStep\(\)](#)*generator (out)*

Output: Generator that dispatched

changedP (out)

Output: Changed active power

changedQ (out)

Output: Changed reactive power

GetInterruptedPowerAndCustomersForStage

Gets recovery information of a contingency.

[ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
[ int error,
float interruptedPower,
float newInterruptedPower,
float interruptedCustomers,
float newInterruptedCustomers ]
ComContingency.GetInterruptedPowerAndCustomersForStage(float timeOfStageInMinutes)
```

ARGUMENTS

timeOfStageInMinutes

Input: Get Information for this time.

interruptedPower (out)

Output: Interrupted Power at this time.

newInterruptedPower (out)

Output: New interrupted Power at this time.

interruptedCustomers (out)

Output: Interrupted Customers at this time.

newInterruptedCustomers (out)

Output: New interrupted Customers at this time.

RETURNS

0 On success.

1 On error.

GetInterruptedPowerAndCustomersForTimeStep

Gets recovery information of a contingency.

[ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
[ float interruptedPower,
float newInterruptedPower,
float interruptedCustomers,
float newInterruptedCustomers ]
ComContingency.GetInterruptedPowerAndCustomersForTimeStep(int currentTimeStep)
```

ARGUMENTS

currentTimeStep

Input: Number of time steps are get via [ComContingency.GetNumberOfTimeSteps\(\)](#)

interruptedPower (out)

Output: Interrupted Power at this time.

newInterruptedPower (out)

Output: New interrupted Power at this time.

interruptedCustomers (out)

Output: Interrupted Customers at this time.

newInterruptedCustomers (out)

Output: New interrupted Customers at this time.

GetLoadEvent

Gets load event of a certain time step during recovery.

[ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
[ DataObject load,
float changedP,
float changedQ,
int isTransfer ) ]
ComContingency.GetLoadEvent(int currentTimeStep,
int loadEvent)
```


ARGUMENTS

*currentTimeStep*Input: Number of time steps are get via [ComContingency.GetNumberOfTimeSteps\(\)](#)*switchEvent*Input: Number of load events for a certain time step are get via [ComContingency.GetNumberOfSwitchEventsForTimeStep\(\)](#)*load (out)* Output: Load that is shed or transfered*changedP (out)*

Output: Changed active power

changedQ (out)

Output: Changed reactive power

isTransfer (out)

Output: = 0 : is load shedding event. >0 is load transfer event.

GetNumberOfGeneratorEventsForTimeStep

Returns the number of generator events of a certain step during recovery.

[ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
int ComContingency.GetNumberOfGeneratorEventsForTimeStep([int currentTimeStep])
```

ARGUMENTS

*currentTimeStep*Input: Number of time steps are get via [ComContingency.GetNumberOfTimeSteps\(\)](#)**GetNumberOfLoadEventsForTimeStep**

Returns the number of load events of a certain step during recovery.

[ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
int ComContingency.GetNumberOfLoadEventsForTimeStep([int currentTimeStep])
```

ARGUMENTS

*currentTimeStep*Input: Number of time steps are get via [ComContingency.GetNumberOfTimeSteps\(\)](#)**GetNumberOfSwitchEventsForTimeStep**

Returns the number of switch events of a certain step during recovery.

[ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
int ComContingency.GetNumberOfSwitchEventsForTimeStep([int currentTimeStep])
```

ARGUMENTS

*currentTimeStep*Input: Number of time steps are get via [ComContingency.GetNumberOfTimeSteps\(\)](#)**GetNumberOfTimeSteps**

Returns the number of time steps during recovery.

[ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
int ComContingency.GetNumberOfTimeSteps()
```

GetObj

Gets interrupted element by index (zero based).

```
DataObject ComContingency.GetObj(int index)
```

ARGUMENTS

index Element order index, 0 for the first object.

RETURNS

object Interrupted element for given index.**None** Index out of range.**GetSwitchEvent**

Gets switch event of a certain time step during recovery.

[ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
[ DataObject switchToBeActuated,
  int isClosed,
  int sectionalizingStep ]
ComContingency.GetSwitchEvent(int currentTimeStep,
                              int switchEvent)
```

ARGUMENTS

*currentTimeStep*Input: Number of time steps are get via [ComContingency.GetNumberOfTimeSteps\(\)](#)*switchEvent*Input: Number of switch event for a certain time step are get via [ComContingency.GetNumberOfSwitchEventsForTimeStep\(\)](#)*switchToBeActuated (out)*

Output: Switch to be actuated

isClosed (out)

Output: > 0 if switch is closed

sectionalizingStep (out)

Output: sectionalizing step when this switch is actuated

GetTimeOfStepInSeconds

Returns the time of the current step during recovery.

[ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
int ComContingency.GetTimeOfStepInSeconds(int currentTimeStep)
```

ARGUMENTS

currentTimeStep

Input: Number of time steps are get via [ComContingency.GetNumberOfTimeSteps\(\)](#)

GetTotalInterruptedPower

Gets the total interrupted power (in kW) during restoration. [ComContingency.CreateRecoveryInformation\(\)](#) has to be called beforehand to collect the data.

```
float ComContingency.GetTotalInterruptedPower()
```

JumpToLastStep

Gets the last trace execution for this contingency.

```
int ComContingency.JumpToLastStep([int timeDelay])
```

ARGUMENTS

timeDelay (optional)

time delay in seconds between trace steps

RETURNS

0 On success.

1 On error.

RemoveEvents

Removes events from this contingency.

```
None ComContingency.RemoveEvents([float emitMessage])
None ComContingency.RemoveEvents(str whichEvents)
None ComContingency.RemoveEvents(float emitMessage,
                                   str whichEvents
                                   )
None ComContingency.RemoveEvents(str whichEvents,
                                   float emitMessage
                                   )
```

ARGUMENTS

emitMessage(optional)

0: no info message shall be issued after event removal

whichEvents(optional)

'lod' removed load events, 'gen' removes generator events, 'switch' removes switching events

StartTrace

Starts trace execution for this contingency.

```
int ComContingency.StartTrace()
```

RETURNS

- 0** On success.
- 1** Error, e.g. Contingency is not in trace.
- 2** On error.

StopTrace

Stops trace execution for this contingency.

```
int ComContingency.StopTrace([int emitMessage])
```

ARGUMENTS

emitMessage (optional)
= 0: no trace-stop info messages shall be issued

RETURNS

- 0** On Success.
- 1** Contingency is not in Trace.

5.4.12 ComCoordreport**Overview**

[DevicesToReport](#)
[HasResultsForDirectionalBackup](#)
[HasResultsForNonDirectionalBackup](#)
[HasResultsForOverreach](#)
[HasResultsForZone](#)
[MaxZoneNumberFor](#)
[ResultForDirectionalBackupVariable](#)
[ResultForNonDirectionalBackupVariable](#)
[ResultForOverreachVariable](#)
[ResultForZoneVariable](#)
[TopologyForDirectionalBackupVariable](#)
[TopologyForNonDirectionalBackupVariable](#)
[TopologyForOverreachVariable](#)
[TopologyForZoneVariable](#)
[TransferDirectionalBackupResultsTo](#)
[TransferNonDirectionalBackupResultsTo](#)
[TransferOverreachResultsTo](#)
[TransferResultsTo](#)
[TransferZoneResultsTo](#)

DevicesToReport

Returns the devices with stored results, i.e. which can be reported.

```
list ComCoordreport.DevicesToReport()
```

RETURNS

Container with reportable devices.

HasResultsForDirectionalBackup

Checks whether there is a stored directional backup result for the given device.

```
int ComCoordreport.HasResultsForDirectionalBackup(DataObject device)
```

ARGUMENTS

device Device for which to check.

RETURNS

- 1** A directional backup result is stored for this device.
- 0** The device has no stored results or no result for the directional backup.

SEE ALSO

[ComCoordreport.HasResultsForZone\(\)](#), [ComCoordreport.HasResultsForOverreach\(\)](#), [ComCoordreport.HasResultsForNonDirectionalBackup\(\)](#)

HasResultsForNonDirectionalBackup

Checks whether there is a stored non-directional backup result for the given device.

```
int ComCoordreport.HasResultsForNonDirectionalBackup(object device)
```

ARGUMENTS

device Device for which to check.

RETURNS

- 1** A non-directional backup result is stored for this device.
- 0** The device has no stored results or no result for the non-directional backup.

SEE ALSO

[ComCoordreport.HasResultsForZone\(\)](#), [ComCoordreport.HasResultsForOverreach\(\)](#), [ComCoordreport.HasResultsForDirectionalBackup\(\)](#)

HasResultsForOverreach

Checks whether there is a stored overreach zone result for the given device.

```
int ComCoordreport.HasResultsForOverreach(DataObject device)
```

ARGUMENTS

device Device for which to check.

RETURNS

- 1** An overreach zone result is stored for this device.
- 0** The device has no stored results or no result for the overreach zone.

SEE ALSO

[ComCoordreport.HasResultsForZone\(\)](#), [ComCoordreport.HasResultsForDirectionalBackup\(\)](#), [ComCoordreport.HasResultsForNonDirectionalBackup\(\)](#)

HasResultsForZone

Checks whether there is a stored result for the given device and zone number.

```
int ComCoordreport.HasResultsForZone(DataObject device,
                                     int zoneNumber)
```

ARGUMENTS

- device* Device for which to check.
- zoneNumber* Zone number to check (1-4).

RETURNS

- 1** A result is stored for this device and zone number.
- 0** The device has no stored results or no result for this zone number.

SEE ALSO

[ComCoordreport.HasResultsForOverreach\(\)](#), [ComCoordreport.HasResultsForDirectionalBackup\(\)](#), [ComCoordreport.HasResultsForNonDirectionalBackup\(\)](#)

MaxZoneNumberFor

Returns the highest zone number in the stored results for the given device.

```
int ComCoordreport.MaxZoneNumberFor(DataObject device)
```

ARGUMENTS

- device* Device for which to retrieve the zone number.

RETURNS

Highest zone number in the stored results.

ResultForDirectionalBackupVariable

Provides access to the directional backup result for a given device and variable.

```
[ int error,
  float result ] ComCoordreport.ResultForDirectionalBackupVariable(DataObject device,
                                                                    str variable)
```

ARGUMENTS

device Device for which to retrieve the result.

variable Variable for which to retrieve the result:

dir Tripping direction

Tp Polygonal delay

Tc Circular delay

result (out)

Value of the stored result.

RETURNS

0 The result is valid.

1 The result was not calculated or not found.

2 The corresponding topological search failed.

3 The corresponding topological search ended prematurely (e.g. end of network reached).

SEE ALSO

[ComCoordreport.ResultForZoneVariable\(\)](#), [ComCoordreport.ResultForOverreachVariable\(\)](#), [ComCoordreport.ResultForNonDirectionalBackupVariable\(\)](#)

ResultForNonDirectionalBackupVariable

Provides access to the non-directional backup result for a given device and variable.

```
[ int error,
float result ] ComCoordreport.ResultForNonDirectionalBackupVariable(DataObject device,
                                                                    str variable)
```

ARGUMENTS

device Device for which to retrieve the result.

variable Variable for which to retrieve the result:

dir Tripping direction

Tp Polygonal delay

Tc Circular delay

result (out)

Value of the stored result.

RETURNS

0 The result is valid.

1 The result was not calculated or not found.

2 The corresponding topological search failed.

3 The corresponding topological search ended prematurely (e.g. end of network reached).

SEE ALSO

[ComCoordreport.ResultForZoneVariable\(\)](#), [ComCoordreport.ResultForOverreachVariable\(\)](#),
[ComCoordreport.ResultForDirectionalBackupVariable\(\)](#)

ResultForOverreachVariable

Provides access to the overreach result for a given device and variable.

```
[ int error,
float result ] ComCoordreport.ResultForOverreachVariable(DataObject device,
                                                         str variable)
```

ARGUMENTS

device Device for which to retrieve the result.

variable Variable for which to retrieve the result:

X	Polygonal reactance
R	Polygonal phase-phase resistance
RE	Polygonal phase-earth resistance
Z	Circular impedance
phi	Circular angle
dir	Tripping direction
Tp	Polygonal delay
Tc	Circular delay

result (out)
 Value of the stored result.

RETURNS

0	The result is valid.
1	The result was not calculated or not found.
2	The corresponding topological search failed.
3	The corresponding topological search ended prematurely (e.g. end of network reached).

SEE ALSO

[ComCoordreport.ResultForZoneVariable\(\)](#), [ComCoordreport.ResultForDirectionalBackupVariable\(\)](#),
[ComCoordreport.ResultForNonDirectionalBackupVariable\(\)](#)

ResultForZoneVariable

Provides access to the result for a given device, zone number and variable.

```
[ int error,
float result ] ComCoordreport.ResultForZoneVariable(DataObject device,
                                                    int zoneNumber,
                                                    str variable)
```


ARGUMENTS

<i>device</i>	Device for which to retrieve the result.
<i>zoneNumber</i>	Zone number for which to retrieve the result (1-4).
<i>variable</i>	Variable for which to retrieve the result:
X	Polygonal reactance
R	Polygonal phase-phase resistance
RE	Polygonal phase-earth resistance
Z	Circular impedance
phi	Circular angle
dir	Tripping direction
Tp	Polygonal delay
Tc	Circular delay

result (out)
Value of the stored result.

RETURNS

- 0** The result is valid.
- 1** The result was not calculated or not found.
- 2** The corresponding topological search failed.
- 3** The corresponding topological search ended prematurely (e.g. end of network reached).

SEE ALSO

[ComCoordreport.ResultForOverreachVariable\(\)](#), [ComCoordreport.ResultForDirectionalBackupVariable\(\)](#), [ComCoordreport.ResultForNonDirectionalBackupVariable\(\)](#)

TopologyForDirectionalBackupVariable

Returns the associated directional backup topology for a given device and variable.

```
list ComCoordreport.TopologyForDirectionalBackupVariable(DataObject device,
                                                         str variable)
```

ARGUMENTS

<i>device</i>	Device for which to retrieve the topology.
<i>variable</i>	Variable for which to retrieve the topology:
Tp	Polygonal delay
Tc	Circular delay

RETURNS

Elements traversed by the topological search determining this variables result.

SEE ALSO

[ComCoordreport.TopologyForZoneVariable\(\)](#), [ComCoordreport.TopologyForOverreachVariable\(\)](#), [ComCoordreport.TopologyForNonDirectionalBackupVariable\(\)](#)

TopologyForNonDirectionalBackupVariable

Returns the associated non-directional backup topology for a given device and variable.

```
list ComCoordreport.TopologyForNonDirectionalBackupVariable(DataObject device,
                                                             str variable)
```

ARGUMENTS

device Device for which to retrieve the topology.

variable Variable for which to retrieve the topology:

Tp	Polygonal delay
Tc	Circular delay

RETURNS

Elements traversed by the topological search determining this variables result.

SEE ALSO

[ComCoordreport.TopologyForZoneVariable\(\)](#), [ComCoordreport.TopologyForOverreachVariable\(\)](#), [ComCoordreport.TopologyForDirectionalBackupVariable\(\)](#)

TopologyForOverreachVariable

Returns the associated overreach zone topology for a given device and variable.

```
list ComCoordreport.TopologyForOverreachVariable(DataObject device,
                                                  str variable)
```

ARGUMENTS

device Device for which to retrieve the topology.

variable Variable for which to retrieve the topology:

X	Polygonal reactance
R	Polygonal phase-phase resistance
RE	Polygonal phase-earth resistance
Z	Circular impedance
phi	Circular angle
Tp	Polygonal delay
Tc	Circular delay

RETURNS

Elements traversed by the topological search determining this variables result.

SEE ALSO

[ComCoordreport.TopologyForZoneVariable\(\)](#), [ComCoordreport.TopologyForDirectionalBackupVariable\(\)](#), [ComCoordreport.TopologyForNonDirectionalBackupVariable\(\)](#)

TopologyForZoneVariable

Returns the associated topology for a given device, zone number and variable.

```
list ComCoordreport.TopologyForZoneVariable(DataObject device,
                                             int zoneNumber,
                                             str variable)
```

ARGUMENTS

<i>device</i>	Device for which to retrieve the topology.
<i>zoneNumber</i>	Zone number for which to retrieve the topology (1-4).
<i>variable</i>	Variable for which to retrieve the topology:
X	Polygonal reactance
R	Polygonal phase-phase resistance
RE	Polygonal phase-earth resistance
Z	Circular impedance
phi	Circular angle
Tp	Polygonal delay
Tc	Circular delay

RETURNS

Elements traversed by the topological search determining this variables result.

SEE ALSO

[ComCoordreport.TopologyForOverreachVariable\(\)](#), [ComCoordreport.TopologyForDirectionalBackupVariable\(\)](#), [ComCoordreport.TopologyForNonDirectionalBackupVariable\(\)](#)

TransferDirectionalBackupResultsTo

Transfers the results of the directional backup for the given variables to the device.

```
int ComCoordreport.TransferDirectionalBackupResultsTo(DataObject device,
                                                    str variables)
```

ARGUMENTS

<i>device</i>	Device to transfer the results to.
<i>variable</i>	Variables to transfer as semi-colon separated string:
dir	Tripping direction
Tp	Polygonal delay
Tc	Circular delay

RETURNS

0	Transfer did non succeed.
1	Result transfer successful.

SEE ALSO

[ComCoordreport.TransferZoneResultsTo\(\)](#), [ComCoordreport.TransferOverreachResultsTo\(\)](#), [ComCoordreport.TransferNonDirectionalBackupResultsTo\(\)](#)

TransferNonDirectionalBackupResultsTo

Transfers the results of the non-directional backup for the given variables to the device.

```
int ComCoordreport.TransferNonDirectionalBackupResultsTo(DataObject device,
                                                         str variables)
```

ARGUMENTS

<i>device</i>	Device to transfer the results to.
<i>variable</i>	Variables to transfer as semi-colon separated string:
dir	Tripping direction
Tp	Polygonal delay
Tc	Circular delay

RETURNS

0	Transfer did non succeed.
1	Result transfer successful.

SEE ALSO

[ComCoordreport.TransferZoneResultsTo\(\)](#), [ComCoordreport.TransferOverreachResultsTo\(\)](#), [ComCoordreport.TransferDirectionalBackupResultsTo\(\)](#)

TransferOverreachResultsTo

Transfers the results of the overreach zone for the given variables to the device.

```
int ComCoordreport.TransferOverreachResultsTo(DataObject device,
                                              str variables)
```

ARGUMENTS

<i>device</i>	Device to transfer the results to.
<i>variable</i>	Variables to transfer as semi-colon separated string:
X	Polygonal reactance
R	Polygonal phase-phase resistance
RE	Polygonal phase-earth resistance
Z	Circular impedance
phi	Circular angle
dir	Tripping direction
Tp	Polygonal delay
Tc	Circular delay

RETURNS

0	Transfer did non succeed.
1	Result transfer successful.

SEE ALSO

[ComCoordreport.TransferZoneResultsTo\(\)](#), [ComCoordreport.TransferDirectionalBackupResultsTo\(\)](#), [ComCoordreport.TransferNonDirectionalBackupResultsTo\(\)](#)

TransferResultsTo

Transfers the results for the given variables to one or more devices.

```
int ComCoordreport.TransferResultsTo(DataObject device,
                                     str variables)

int ComCoordreport.TransferResultsTo(list devices,
                                     str variables)
```

ARGUMENTS

- devices* Devices to transfer the results to.
- variable* Variables to transfer as semi-colon separated string:
- | | |
|------------|----------------------------------|
| X | Polygonal reactance |
| R | Polygonal phase-phase resistance |
| RE | Polygonal phase-earth resistance |
| Z | Circular impedance |
| phi | Circular angle |
| dir | Tripping direction |
| Tp | Polygonal delay |
| Tc | Circular delay |

RETURNS

- 0** At least one transfer did non succeed.
- 1** Result transfer successful.

TransferZoneResultsTo

Transfers the results of a particular zone for the given variables to the device.

```
int ComCoordreport.TransferZoneResultsTo(DataObject device,
                                         int zoneNumber,
                                         str variables)
```

ARGUMENTS

- device* Device to transfer the results to.
- zoneNumber* Zone number for which to to transfer the results (1-4).
- variable* Variables to transfer as semi-colon separated string:
- | | |
|------------|----------------------------------|
| X | Polygonal reactance |
| R | Polygonal phase-phase resistance |
| RE | Polygonal phase-earth resistance |
| Z | Circular impedance |
| phi | Circular angle |
| dir | Tripping direction |
| Tp | Polygonal delay |
| Tc | Circular delay |

RETURNS

- | | |
|----------|-----------------------------|
| 0 | Transfer did non succeed. |
| 1 | Result transfer successful. |

SEE ALSO

[ComCoordreport.TransferOverreachResultsTo\(\)](#), [ComCoordreport.TransferDirectionalBackupResultsTo\(\)](#), [ComCoordreport.TransferNonDirectionalBackupResultsTo\(\)](#)

5.4.13 ComDllmanager

Overview

[Report](#)

Report

Prints a status report of currently available external user-defined dlls (e.g. dpl, exdyn) to the output window. (Same as pressing the 'Report' button in the dialog.)

```
None ComDllmanager.Report()
```

5.4.14 ComDpl

Overview

[CheckSyntax](#)
[Encrypt](#)
[Execute](#)
[GetExternalObject](#)
[GetInputParameterDouble](#)
[GetInputParameterInt](#)
[GetInputParameterString](#)
[IsEncrypted](#)
[ResetThirdPartyModule](#)
[SetExternalObject](#)
[SetInputParameterDouble](#)
[SetInputParameterInt](#)
[SetInputParameterString](#)
[SetThirdPartyModule](#)

CheckSyntax

Checks the syntax and input parameter of the DPL script and all its subscripts.

```
int ComDpl.CheckSyntax()
```

RETURNS

- | | |
|----------|-------------|
| 0 | On success. |
| 1 | On error. |

SEE ALSO

[ComDpl.Execute\(\)](#)

Encrypt

Encrypts a script and all its subscripts. An encrypted script can be executed without password but decrypted only with password. If no password is given a 'Choose Password' dialog appears.

```
int ComDpl.Encrypt([str password = ""],
                  [int removeObjectHistory = 1],
                  [int masterCode = 0])
```

ARGUMENTS

password (optional)

Password for decryption. If no password is given a 'Choose Password' dialog appears.

removeObjectHistory (optional)

Handling of unencrypted object history in database, e.g. used by project versions or by undo:

- | | |
|----------|----------------------|
| 0 | Do not remove. |
| 1 | Do remove (default). |
| 2 | Show dialog and ask. |

masterCode (optional)

Used for re-selling scripts. 3rd party licence codes already set in the script will be overwritten by this value (default = 0).

RETURNS

- | | |
|----------|-------------|
| 0 | On success. |
| 1 | On error. |

SEE ALSO

[ComDpl.IsEncrypted\(\)](#)

Execute

Executes the DPL script as subscript.

```
int ComDpl.Execute([input parameter])
```

ARGUMENTS

input parameter (optional)

All input parameter from the 'Basic Options' page of the 'Edit' dialog can be given as arguments. If a parameter is not given then the value from the dialog is used. The values from the dialog itself are not modified. These can be modified via

- [ComDpl.SetInputParameterInt\(\)](#)
- [ComDpl.SetInputParameterDouble\(\)](#)
- [ComDpl.SetInputParameterString\(\)](#).

The arguments are not given by reference. Thus when a subscript changes the value of a variable then the value from the main script is not changed.

RETURNS

For scripts without the use of `Application.exit()` the following values are returned:

- 0** On a successful execution.
- 1** An error occurred.
- 6** User hit the break button.

SEE ALSO

[ComDpl.CheckSyntax\(\)](#)

GetExternalObject

Gets the external object defined in the ComDpl edit dialog.

```
[int error,  
DataObject value] ComDpl.GetExternalObject(str name)
```

ARGUMENTS

- name* Name of the external object parameter.
- value (out)*
The external object.

RETURNS

- 0** On success.
- 1** On error.

SEE ALSO

[ComDpl.SetExternalObject\(\)](#), [ComDpl.GetInputParameterInt\(\)](#), [ComDpl.GetInputParameterDouble\(\)](#), [ComDpl.GetInputParameterString\(\)](#)

GetInputParameterDouble

Gets the double input parameter value defined in the ComDpl edit dialog.

```
[int error,  
float value] ComDpl.GetInputParameterDouble(str name)
```

ARGUMENTS

- name* Name of the double input parameter.
- value (out)*
Value of the double input parameter.

RETURNS

- 0** On success.
- 1** On error.

SEE ALSO

[ComDpl.SetInputParameterDouble\(\)](#), [ComDpl.GetInputParameterInt\(\)](#), [ComDpl.GetInputParameterString\(\)](#), [ComDpl.GetExternalObject\(\)](#)

GetInputParameterInt

Gets the integer input parameter value defined in the ComDpl edit dialog.

```
[int error,  
int value ] ComDpl.GetInputParameterInt(str name)
```

ARGUMENTS

name Name of the integer input parameter.

value (out)
Value of the integer input parameter.

RETURNS

0 On success.

1 On error.

SEE ALSO

[ComDpl.SetInputParameterInt\(\)](#), [ComDpl.GetInputParameterDouble\(\)](#), [ComDpl.GetInputParameterString\(\)](#),
[ComDpl.GetExternalObject\(\)](#)

GetInputParameterString

Gets the string input parameter value defined in the ComDpl edit dialog.

```
[int error,  
str value ] ComDpl.GetInputParameterString(str name)
```

ARGUMENTS

name Name of the string input parameter.

value (out)
Value of the string input parameter.

RETURNS

0 On success.

1 On error.

SEE ALSO

[ComDpl.SetInputParameterString\(\)](#), [ComDpl.GetInputParameterInt\(\)](#), [ComDpl.GetInputParameterDouble\(\)](#),
[ComDpl.GetExternalObject\(\)](#)

IsEncrypted

Returns the encryption state of the script.

```
int ComDpl.IsEncrypted()
```

RETURNS

1 Script is encrypted.

0 Script is not encrypted.

SEE ALSO

[ComDpl.Encrypt\(\)](#)

ResetThirdPartyModule

Resets the third party licence. Only possible for non-encrypted scripts. Requires masterkey licence for third party module currently set.

```
int ComDpl.ResetThirdPartyModule()
```

RETURNS

0	On success.
1	On error.

SetExternalObject

Sets the external object defined in the ComDpl edit dialog.

```
int ComDpl.SetExternalObject(str name,  
                             DataObject value  
                             )
```

ARGUMENTS

<i>name</i>	Name of the external object parameter.
<i>value</i>	The external object.

RETURNS

0	On success.
1	On error.

SEE ALSO

[ComDpl.GetExternalObject\(\)](#), [ComDpl.SetInputParameterInt\(\)](#), [ComDpl.SetInputParameterDouble\(\)](#), [ComDpl.SetInputParameterString\(\)](#)

SetInputParameterDouble

Sets the double input parameter value defined in the ComDpl edit dialog.

```
int ComDpl.SetInputParameterDouble(str name,  
                                    float value  
                                    )
```

ARGUMENTS

<i>name</i>	Name of the double input parameter.
<i>value</i>	Value of the double input parameter.

RETURNS

0	On success.
1	On error.

SEE ALSO

[ComDpl.GetInputParameterDouble\(\)](#), [ComDpl.SetInputParameterInt\(\)](#), [ComDpl.SetInputParameterString\(\)](#), [ComDpl.SetExternalObject\(\)](#)

SetInputParameterInt

Sets the integer input parameter value defined in the ComDpl edit dialog.

```
int ComDpl.SetInputParameterInt(str name,
                                int value
                                )
```

ARGUMENTS

name Name of the integer input parameter.
value Value of the integer input parameter.

RETURNS

0 On success.
1 On error.

SEE ALSO

[ComDpl.GetInputParameterInt\(\)](#), [ComDpl.SetInputParameterDouble\(\)](#), [ComDpl.SetInputParameterString\(\)](#), [ComDpl.SetExternalObject\(\)](#)

SetInputParameterString

Sets the string input parameter value defined in the ComDpl edit dialog.

```
int ComDpl.SetInputParameterString(str name,
                                    str value
                                    )
```

ARGUMENTS

name Name of the string input parameter.
value Value of the string input parameter.

RETURNS

0 On success.
1 On error.

SEE ALSO

[ComDpl.GetInputParameterString\(\)](#), [ComDpl.SetInputParameterInt\(\)](#), [ComDpl.SetInputParameterDouble\(\)](#), [ComDpl.SetExternalObject\(\)](#)

SetThirdPartyModule

Sets the third party licence to a specific value. Only possible for non-encrypted scripts with no third party licence set so far. Requires masterkey licence for third party module to be set.

```
int ComDpl.SetThirdPartyModule(str companyCode,
                                str moduleCode
                                )
```

ARGUMENTS

companyCode

Display name or numeric value of company code.

moduleCode

Display name or numeric value of third party module.

RETURNS

0 On success.**1** On error.

5.4.15 ComFlickermeter

Overview

[Execute](#)

Execute

Calculates the short- and long-term flicker according to IEC 61000-4-15.

```
int ComFlickermeter.Execute()
```

RETURNS

0 OK**1** Error: column not found in file; other internal errors**2** Error: empty input file**3** Error: cannot open file**4** Internal error: matrix empty

5.4.16 ComGenrelinc

Overview

[GetCurrentIteration](#)[GetMaxNumIterations](#)

GetCurrentIteration

The command returns the current iteration number of the 'Run Generation Adequacy' command (ComGenrel).

```
int ComGenrelinc.GetCurrentIteration()
```

RETURNS

Returns the current iteration number.

GetMaxNumIterations

The command returns the maximum number of iterations specified in the 'Run Generation Adequacy' command (ComGenrel).

```
int ComGenrelinc.GetMaxNumIterations()
```

RETURNS

Returns the maximum number of iterations.

5.4.17 ComGridtocim

Overview

[ConvertAndExport](#)
[SetAuthorityUri](#)
[SetBoundaries](#)
[SetGridsToExport](#)

ConvertAndExport

Convert Grid to CIM and export CIM data to given file path without storing into database. If no file path is provided, the file path from the corresponding CIM Data Export command in the study will be used. In case of a validation error the export is not performed.

```
int ComGridtocim.ConvertAndExport([int validate])
int ComGridtocim.ConvertAndExport(str filePath,
                                   [int validate])
```

ARGUMENTS

filePath File path for CIM data.

validate (optional)

Option to execute CIM Data Validation before export. If not provided, the validation is executed. Possible values are

0	Do not validate
1	Validate

RETURNS

0	OK
1	Error: conversion failed
2	Error: export failed

SetAuthorityUri

Sets the authority uri for a specific grid.

```
None ComGridtocim.SetAuthorityUri(DataObject grid,
                                   str uri)
```

ARGUMENTS

- grid* Grid to set the URI for.
- uri* Model authority URI to be set.

SetBoundaries

Sets the grids as "Boundary Grid" and clears any previous setting.

```
None ComGridtocim.SetBoundaries(list grids)
```

ARGUMENTS

- grids* The grids to be considered as boundaries.

SetGridsToExport

Sets the grids as "Selected" and clears any previous setting.

```
None ComGridtocim.SetGridsToExport(list grids)
```

ARGUMENTS

- grids* The grids to be selected.

5.4.18 ComHostcap**Overview**

[CalcMaxHostedPower](#)

CalcMaxHostedPower

The function executes predefined hosting capacity analysis command and returns the max. hosted power (P, Q) at the given terminal. In addition, object where the violation has occurred is returned.

```
[int returnValue,
float P,
float Q,
violatedElement ] ComHostcap.CalcMaxHostedPower(DataObject terminal)
```

ARGUMENTS

- terminal* Hosting site.
- P (out)* Max. active power.
- Q (out)* Max. reactive power.
- violatedElement (out)*
Element where the limiting violation occurs.

RETURNS

- 1 Selected object is not a terminal.
- 0 No violations. Calculation OK.
- 1 Calculation failed.
- 2 Thermal violation.
- 3 Voltage violation.
- 4 Protection violation.
- 5 Power quality violation.

5.4.19 ComImport

Overview

[GetCreatedObjects](#)
[GetModifiedObjects](#)

GetCreatedObjects

Returns the newly created objects after execution of a DGS import.

```
list ComImport.GetCreatedObjects()
```

RETURNS

Collection of objects that have been created during DGS import.

GetModifiedObjects

Returns the modified objects after execution of a DGS import.

```
list ComImport.GetModifiedObjects()
```

RETURNS

Collection of objects that have been modified during DGS import.

5.4.20 ComInc

Overview

[ZeroDerivative](#)

ZeroDerivative

This function returns 1 if the state variable derivatives are less than the tolerance for the initial conditions, provided that the *Simulation method* is set to *RMS values* and the *Verify initial conditions* option is selected in the *Calculation of initial conditions* command. The tolerance is defined on the *Solver options* page in the *Maximum error for dynamic model equations* field. The function returns 0 if the aforementioned conditions are not met, or if at least one state variable has a derivative larger than the tolerance.

```
int ComInc.ZeroDerivative()
```

RETURNS

- 0** At least one state variable has a derivative larger than the tolerance, or the required command options have not been set.
- 1** All state variable derivatives are less than the tolerance.

5.4.21 ComLdf

Overview

[CheckControllers](#)
[DoNotResetCalc](#)
[EstimateOutage](#)
[Execute](#)
[IsAC](#)
[IsBalanced](#)
[IsDC](#)
[PrintCheckResults](#)
[SetOldDistributeLoadMode](#)

CheckControllers

Check the conditions of all controllers based on available load flow results. The report will be printed out in output window.

```
int ComLdf.CheckControllers()
```

RETURNS

Always return 1.

DoNotResetCalc

The load flow results will not be reset even the load flow calculation fails.

```
int ComLdf.DoNotResetCalc(int doNotReset)
```

ARGUMENTS

doNotReset

Specifies whether the results shall be reset or not.

- 0** Reset load flow results if load flow fails.
- 1** Load flow results will remain even load flow fails.

RETURNS

Always return 0.

EstimateOutage

Estimate the loading of all branches with outages of given set of branch elements.

```
int ComLdf.EstimateOutage(list branches,
                          int init
                          )
```


ARGUMENTS

branches The branch elements to be in outage.

init Initialisation of sensitivities.

- | | |
|----------|---|
| 0 | No need to calculate sensitivities; it assumes that sensitivities have been calculated before hand. |
| 1 | Sensitivities will be newly calculated. |

RETURNS

- | | |
|----------|-------------|
| 0 | On success. |
| 1 | On error. |

Execute

Performs a load flow analysis on a network. Results are displayed in the single line graphic and available in relevant elements.

```
int ComLdf.Execute()
```

RETURNS

- | | |
|----------|--|
| 0 | OK |
| 1 | Load flow failed due to divergence of inner loops. |
| 2 | Load flow failed due to divergence of outer loops. |

IsAC

Check whether this load flow is configured as AC method or not.

```
int ComLdf.IsAC()
```

RETURNS

- | | |
|----------|------------------|
| 0 | Is a DC method. |
| 1 | Is an AC method. |

IsBalanced

Check whether this load flow command is configured as balanced or unbalanced.

```
int ComLdf.IsBalanced()
```

RETURNS

Returns true if the load flow is balanced.

IsDC

Check whether this load flow is configured as DC method or not.

```
int ComLdf.IsDC()
```

RETURNS

- 0** Is an AC method.
- 1** Is a DC method.

PrintCheckResults

Shows the verification report in the output window.

```
int ComLdf.PrintCheckResults()
```

RETURNS

Always return 1.

SetOldDistributeLoadMode

Set the old scaling mode in case of Distributed Slack by loads.

```
None ComLdf.SetOldDistributeLoadMode(int iOldMode)
```

ARGUMENTS

iOldMode The flag showing if the old model is used.

- 0** Use standard mode.
- 1** Use old mode.

5.4.22 ComLink**Overview**

[LoadMicroSCADAFile](#)
[ReceiveData](#)
[SendData](#)
[SentDataStatus](#)
[SetOPCReceiveQuality](#)
[SetSwitchShcEventMode](#)

LoadMicroSCADAFile

Reads in a MicroSCADA snapshot file.

```
int ComLink.LoadMicroSCADAFile(str filename,
                                [int populate]
                                )
```

ARGUMENTS

filename name of the file to read

populate (optional)

determines whether new values should be populated to the network elements
(0=no, 1=yes)

RETURNS

- 0** On success.
- 1** On error.

ReceiveData

Reads and processes values for all (in PowerFactory configured) items from OPC server (OPC only).

```
int ComLink.ReceiveData([int force])
```

ARGUMENTS

force (optional)

- 0** (default) Processes changed values (asynchronously) received by PowerFactory via callback
- 1** Forces (synchronous) reading and processing of all values (independent of value changes)

RETURNS

Number of read items

SendData

Sends values from configured measurement objects to OPC server (OPC only).

```
int ComLink.SendData([int force])
```

ARGUMENTS

force (optional)

- 0** (default) Send only data that have been changed and difference between old and new value is greater than configured deadband
- 1** Forces writing of all values (independent of previous value)

RETURNS

Number of written items

SentDataStatus

Outputs status of all items marked for sending to output window.

```
int ComLink.SentDataStatus()
```

RETURNS

Number of items configured for sending.

SetOPCReceiveQuality

Allows to override the actual OPC receive quality by this value. (Can be used for testing.)

```
int ComLink.SetOPCReceiveQuality(int quality)
```

ARGUMENTS

quality new receive quality (bitmask)

RETURNS

0 On success.
1 On error.

SetSwitchShcEventMode

Configures whether value changes for switches are directly transferred to the object itself or whether shc switch events shall be created instead.

```
None ComLink.SetSwitchShcEventMode(int enabled)
```

ARGUMENTS

enabled

0 Values are directly written to switches
1 For each value change a switch event will be created

5.4.23 ComMerge**Overview**

[CheckAssignments](#)
[Compare](#)
[CompareActive](#)
[ExecuteRecording](#)
[ExecuteWithActiveProject](#)
[GetCorrespondingObject](#)
[GetModification](#)
[GetModificationResult](#)
[GetModifiedObjects](#)
[Merge](#)
[PrintComparisonReport](#)
[PrintModifications](#)
[Reset](#)
[SetAutoAssignmentForAll](#)
[SetObjectsToCompare](#)
[ShowBrowser](#)
[WereModificationsFound](#)

CheckAssignments

Checks if all assignments are correct and merge can be done.

```
int ComMerge.CheckAssignments()
```

RETURNS

0 On success.
1 Cancelled by user.

- 2 Missing assignments found.
- 3 Conflicts found.
- 4 On other errors.

Compare

Starts a comparison according to the settings in this ComMerge object. The merge browser is not shown.

```
int ComMerge.Compare()
```

CompareActive

Starts a comparison according to the settings in this ComMerge object. The merge browser is not shown. Can compare with the active project.

```
int ComMerge.CompareActive()
```

ExecuteRecording

Starts a comparison according to the settings in this ComMerge object and shows the merge browser. Records modifications in the active scenario and/or expansion stage of the target project.

```
int ComMerge.ExecuteRecording()
```

ExecuteWithActiveProject

Starts a comparison according to the settings in this ComMerge object and shows the merge browser. Can compare with the active project.

```
None ComMerge.ExecuteWithActiveProject()
```

GetCorrespondingObject

Searches corresponding object for given object.

```
DataObject ComMerge.GetCorrespondingObject(DataObject sourceObj,
                                           [int target]
                                           )
```

ARGUMENTS

sourceObj

Object for which corresponding object is searched.

target

- 0 Get corresponding object from “Base” (default)
- 1 Get corresponding object from “1st”
- 2 Get corresponding object from “2nd”

RETURNS

- object** Corresponding object.
- None** Corresponding object not found.

GetModification

Gets kind of modification between corresponding objects of “Base” and “1st” or “2nd”.

```
int ComMerge.GetModification(DataObject sourceObj,
                             [int target]
                             )
```

ARGUMENTS

- sourceObj*
Object from any source for which modification is searched.
- target*
 - 1** Get modification from “Base” to “1st” (default)
 - 2** Get modification from “Base” to “2nd”

RETURNS

- 0** On error.
- 1** No modifications (equal).
- 2** Modified.
- 3** Added in “1st”/“2nd”.
- 4** Removed in “1st”/“2nd”.

GetModificationResult

Gets kind of modifications between compared objects in “1st” and “2nd”.

```
int ComMerge.GetModificationResult(DataObject obj)
```

ARGUMENTS

- obj* Object from any source for which modification is searched.

RETURNS

- 0** On error.
- 1** No modifications (equal).
- 2** Same modifications in “1st” and “2nd” (no conflict).
- 3** Different modifications in “1st” and “2nd” (conflict).

GetModifiedObjects

Gets all objects with a certain kind of modification.

```
list ComMerge.GetModifiedObjects(int modType,
                                  [int modSource]
                                  )
```

ARGUMENTS

modType

- | | |
|----------|------------------------|
| 1 | get unmodified objects |
| 2 | get modified objects |
| 3 | get added objects |
| 4 | get removed objects |

modSource

- | | |
|----------|--|
| 1 | consider modification between “Base” and “1st” (default) |
| 2 | consider modification between “Base” and “2nd” |

RETURNS

Set with matching objects.

Unmodified, modified and added objects are always from given “modSource”, removed objects are always from “Base”.

Merge

Checks assignments, merges modifications according to assignments into target and prints merge report to output window.

```
None ComMerge.Merge(int printReport)
```

ARGUMENTS

printReport

- | | |
|----------|------------------------------|
| 1 | print merge report (default) |
| 0 | do not print merge report |
- always set to 0 in paste and split mode

PrintComparisonReport

Prints the modifications of all compared objects as a report to the output window.

```
None ComMerge.PrintComparisonReport(int mode)
```

ARGUMENTS

mode

- | | |
|----------|-------------------------------|
| 0 | no report |
| 1 | only modified compare objects |
| 2 | all compare objects |

PrintModifications

Prints modifications of given objects (if any) to the output window.

```
int ComMerge.PrintModifications(list objs)
int ComMerge.PrintModifications(DataObject obj)
```

ARGUMENTS

- objs* Set of objects for which the modifications are printed.
- obj* Object for which the modifications are printed.

RETURNS

- 0** On error: object(s) not found in comparison.
- 1** On success: modifications were printed.

Reset

Resets/clears and deletes all temp. object sets, created internally for the comparison.

```
None ComMerge.Reset()
```

SetAutoAssignmentForAll

Sets the assignment of all compared objects automatically.

```
None ComMerge.SetAutoAssignmentForAll(int conflictVal)
```

ARGUMENTS

- conflictVal* Assignment of compared objects with undefined automatic values (e.g. conflicts)
- 0** no assignment
 - 1** assign from “Base”
 - 2** assign from 1st
 - 3** assign from 2nd

SetObjectsToCompare

Sets top level objects for comparison.

```
None ComMerge.SetObjectsToCompare(DataObject base,
                                   [DataObject first,]
                                   [DataObject second]
                                   )
```

ARGUMENTS

- base* Top level object to be set as “Base”
- first* Top level object to be set as “1st”
- second* Top level object to be set as “2nd”

ShowBrowser

Shows merge browser with initialized settings and all compared objects. Can only be called after a comparison was executed.

```
int ComMerge.ShowBrowser()
```


RETURNS

- 0** The browser was left with ok button.
- 1** The browser was left with cancel button.
- 2** On error.

WereModificationsFound

Checks, if modifications were found in comparison.

```
int ComMerge.WereModificationsFound()
```

RETURNS

- 0** All objects in comparison are equal.
- 1** Modifications found in comparison.

5.4.24 ComMot

Overview

[GetMotorConnections](#)
[GetMotorSwitch](#)
[GetMotorTerminal](#)

GetMotorConnections

Finds the cables connecting the motor to the switch.

```
list ComMot.GetMotorConnections(DataObject motor)
```

ARGUMENTS

motor The motor element

RETURNS

Returns the set of cables connecting the motor to the switch.

GetMotorSwitch

Finds the switch which will connect the motor to the network.

```
DataObject ComMot.GetMotorSwitch(DataObject motor)
```

ARGUMENTS

motor The motor element

RETURNS

Returns the switch element.

GetMotorTerminal

Finds the terminal to which the motor will be connected.

```
DataObject ComMot.GetMotorTerminal(DataObject motor)
```

ARGUMENTS

motor The motor element

RETURNS

Returns the terminal element.

5.4.25 ComNmink**Overview**

[AddRef](#)
[Clear](#)
[GenerateContingenciesForAnalysis](#)
[GetAll](#)

AddRef

Adds shortcuts to the objects to the existing selection.

```
None ComNmink.AddRef(DataObject O)  
None ComNmink.AddRef(list S)
```

ARGUMENTS

O(optional)
 an object

S(optional)
 a Set of objects

Clear

Delete all contents, i.e. to empty the selection.

```
None ComNmink.Clear()
```

GenerateContingenciesForAnalysis

Generates Contingencies for Contingency Analysis. Similar to calling 'Execute' but does not pop-up the contingency command.

RETURNS

0 On success.
1 On error.

GetAll

Returns all objects which are of the class 'ClassName'.

```
list ComNmink.GetAll(str className)
```

ARGUMENTS

className
The object class name.

RETURNS

The set of objects

5.4.26 ComOmr**Overview**

[GetFeeders](#)
[GetOMR](#)
[GetRegionCount](#)

GetFeeders

Get all feeders for which optimal manual switches have been determined. This function can be used after execution of an Optimal Manual Restoration command only.

```
list ComOmr.GetFeeders()
```

RETURNS

The set of all feeders used for optimisation.

GetOMR

Get terminal and connected optimal manual switches determined by the optimisation for the given feeder and its region(pocket) of the given index. For a detailed description of a pocket, please consult the manual. This function can be used after execution of an Optimal Manual Restoration command only.

```
list ComOmr.GetOMR(DataObject arg0,  
                    int arg1  
                    )
```

ARGUMENTS

arg0 The feeder to derive the resulting optimal terminal with its connected (optimal) manual switches for.

arg1 The index of the region(pocket) inside the given feeder to derive the resulting optimal terminal with its connected (optimal) manual switches for.

RETURNS

The resulting optimal terminal with its connected (optimal) manual switches for the region in the feeder.

GetRegionCount

Get total number of regions(pockets) separated by infeeding point, feeder ends and certain switches for the provided feeder. For a detailed description of a pocket, please consult the manual. This function can be used after execution of an Optimal Manual Restoration command only.

```
int ComOmr.GetRegionCount(DataObject feeder)
```

ARGUMENTS

feeder Feeder to derive number of regions(pockets) for.

RETURNS

Number of regions(pockets) for the feeder.

5.4.27 ComOpc

Overview

[ReceiveData](#)
[SendData](#)

ReceiveData

Reads and processes values for all (in PowerFactory configured) items from OPC server (OPC only).

```
int ComOpc.ReceiveData([int force])
```

ARGUMENTS

force (optional)

- | | |
|----------|--|
| 1 | Forces (synchronous) reading and processing of all values (independent of value changes) |
|----------|--|

RETURNS

1 if successfully received data -1 if an error occurred -2 if the link is not connected

SendData

Sends values from configured measurement objects to OPC server (OPC only).

```
int ComOpc.SendData([int force])
```

ARGUMENTS

force (optional)

- | | |
|----------|--|
| 0 | (default) Send only data that have been changed and difference between old and new value is greater than configured deadband |
| 1 | Forces writing of all values (independent of previous value) |

RETURNS

1 if successfully received data -1 if an error occurred -2 if the link is not connected

5.4.28 ComOutage

Overview

[ContinueTrace](#)
[ExecuteTime](#)
[GetObject](#)
[RemoveEvents](#)
[SetObjs](#)
[StartTrace](#)
[StopTrace](#)

ContinueTrace

Continue the next step of the trace.

```
int ComOutage.ContinueTrace()
```

RETURNS

0 On success.
 $\neq 0$ On error.

ExecuteTime

Execute contingency (with multiple time phase) for the given time.

```
int ComOutage.ExecuteTime(float time)
```

ARGUMENTS

time the given time to be executed.

RETURNS

$= 0$ On success.
 $\neq 0$ On error.

GetObject

Get the element stored in line number “line” in the table of ComOutage. The line index starts with 0.

```
DataObject ComOutage.GetObject(int line)
```

ARGUMENTS

line line index, if index exceeds the range None is returned

RETURNS

the element of line “line” in the table.

RemoveEvents

Remove all events defined in this contingency.

```
None ComOutage.RemoveEvents ([int info])
None ComOutage.RemoveEvents (str type)
None ComOutage.RemoveEvents (int info, str type)
None ComOutage.RemoveEvents (str type, int info)
```

ARGUMENTS

type

none Hidden objects are ignored and not added to the set
'Lod' remove all EvtLod
'Gen' remove all EvtGen
'Switch' remove all EvtSwitch

info(optional)

1 show info message in output window (default)
0 do not show info message

SetObjs

To fill up the "interrupted components" with given elements.

Sets the list of objects according to S.

```
int ComOutage.SetObjs(list S)
```

ARGUMENTS

S the set of objects

RETURNS

0 On success.
1 On error.

StartTrace

Start trace all post fault events of this contingency.

```
int ComOutage.StartTrace()
```

RETURNS

= 0 On success.
≠ 0 On error.

StopTrace

To stop the trace.

```
int ComOutage.StopTrace([int msg])
```

ARGUMENTS

msg (optional)

Emit messages or not.

0 Suppress messages.**1** Emit messages.

RETURNS

= 0 On success.**≠ 0** On error.

5.4.29 ComPfdimport

Overview

[GetImportedObjects](#)

GetImportedObjects

Returns the imported objects of last execution of command.

```
list ComPfdimport.GetImportedObjects()
```

RETURNS

Returns the imported objects of the last execution of the command.

5.4.30 ComPrjconnector

Overview

[GetSuccessfullyConnectedItems](#)[GetUnsuccessfullyConnectedItems](#)

GetSuccessfullyConnectedItems

Returns a list of elements which were correctly processed in the last run of the command

```
list ComPrjconnector.GetSuccessfullyConnectedItems()
```

GetUnsuccessfullyConnectedItems

Returns a list of elements which were not correctly processed in the last run of the command

```
list ComPrjconnector.GetUnsuccessfullyConnectedItems()
```

5.4.31 ComProtgraphic

Overview

[AddToUpdatePages](#)
[ClearUpdatePages](#)

AddToUpdatePages

Adds pages (*.SetVipage) to the user-defined selection of plot pages to update.

```
None ComProtgraphic.AddToUpdatePages(list pages)
```

ClearUpdatePages

Clears the user-defined selection of plot pages to update.

```
None ComProtgraphic.ClearUpdatePages()
```

5.4.32 ComPvcurves

Overview

[FindCriticalBus](#)

FindCriticalBus

Returns the critical bus for a given PV-Curve result file. The critical bus is the one with the highest gradient at the last iteration. If a bus is found, whose voltage drop is twice as high, as the one with the highest gradient, the bus with the higher voltage drop is the critical one.

```
DataObject ComPvcurves.FindCriticalBus(DataObject resultfile)
```

5.4.33 ComPython

Overview

[GetExternalObject](#)
[GetInputParameterDouble](#)
[GetInputParameterInt](#)
[GetInputParameterString](#)
[SetExternalObject](#)
[SetInputParameterDouble](#)
[SetInputParameterInt](#)
[SetInputParameterString](#)

GetExternalObject

Gets the external object defined in the ComPython edit dialog.

```
[int error,  
DataObject value] ComPython.GetExternalObject(str name)
```


ARGUMENTS

name Name of the external object parameter.

value (out)
The external object.

RETURNS

0 On success.

1 On error.

SEE ALSO

[ComPython.SetExternalObject\(\)](#), [ComPython.GetInputParameterInt\(\)](#),
[ComPython.GetInputParameterDouble\(\)](#), [ComPython.GetInputParameterString\(\)](#)

GetInputParameterDouble

Gets the double input parameter value defined in the ComPython edit dialog.

```
[int error,  
float value] ComPython.GetInputParameterDouble(str name)
```

ARGUMENTS

name Name of the double input parameter.

value (out)
Value of the double input parameter.

RETURNS

0 On success.

1 On error.

SEE ALSO

[ComPython.SetInputParameterDouble\(\)](#), [ComPython.GetInputParameterInt\(\)](#),
[ComPython.GetInputParameterString\(\)](#), [ComPython.GetExternalObject\(\)](#)

GetInputParameterInt

Gets the integer input parameter value defined in the ComPython edit dialog.

```
[int error,  
int value ] ComPython.GetInputParameterInt(str name)
```

ARGUMENTS

name Name of the integer input parameter.

value (out)
Value of the integer input parameter.

RETURNS

0 On success.

1 On error.

SEE ALSO

[ComPython.SetInputParameterInt\(\)](#), [ComPython.GetInputParameterDouble\(\)](#),
[ComPython.GetInputParameterString\(\)](#), [ComPython.GetExternalObject\(\)](#)

GetInputParameterString

Gets the string input parameter value defined in the ComPython edit dialog.

```
[int error,  
str value ] ComPython.GetInputParameterString(str name)
```

ARGUMENTS

name Name of the string input parameter.

value (out)
Value of the string input parameter.

RETURNS

0 On success.
1 On error.

SEE ALSO

[ComPython.SetInputParameterString\(\)](#), [ComPython.GetInputParameterInt\(\)](#),
[ComPython.GetInputParameterDouble\(\)](#), [ComPython.GetExternalObject\(\)](#)

SetExternalObject

Sets the external object defined in the ComPython edit dialog.

```
int ComPython.SetExternalObject(str name,  
                                DataObject value  
                                )
```

ARGUMENTS

name Name of the external object parameter.

value The external object.

RETURNS

0 On success.
1 On error.

SEE ALSO

[ComPython.GetExternalObject\(\)](#), [ComPython.SetInputParameterInt\(\)](#),
[ComPython.SetInputParameterDouble\(\)](#), [ComPython.SetInputParameterString\(\)](#)

SetInputParameterDouble

Sets the double input parameter value defined in the ComPython edit dialog.

```
int ComPython.SetInputParameterDouble(str name,  
                                       float value  
                                       )
```

ARGUMENTS

<i>name</i>	Name of the double input parameter.
<i>value</i>	Value of the double input parameter.

RETURNS

0	On success.
1	On error.

SEE ALSO

[ComPython.GetInputParameterDouble\(\)](#), [ComPython.SetInputParameterInt\(\)](#),
[ComPython.SetInputParameterString\(\)](#), [ComPython.SetExternalObject\(\)](#)

SetInputParameterInt

Sets the integer input parameter value defined in the ComPython edit dialog.

```
int ComPython.SetInputParameterInt(str name,  
                                   int value  
                                   )
```

ARGUMENTS

<i>name</i>	Name of the integer input parameter.
<i>value</i>	Value of the integer input parameter.

RETURNS

0	On success.
1	On error.

SEE ALSO

[ComPython.GetInputParameterInt\(\)](#), [ComPython.SetInputParameterDouble\(\)](#),
[ComPython.SetInputParameterString\(\)](#), [ComPython.SetExternalObject\(\)](#)

SetInputParameterString

Sets the string input parameter value defined in the ComPython edit dialog.

```
int ComPython.SetInputParameterString(str name,  
                                       str value  
                                       )
```

ARGUMENTS

<i>name</i>	Name of the string input parameter.
<i>value</i>	Value of the string input parameter.

RETURNS

0	On success.
1	On error.

SEE ALSO

[ComPython.GetInputParameterString\(\)](#), [ComPython.SetInputParameterInt\(\)](#),
[ComPython.SetInputParameterDouble\(\)](#), [ComPython.SetExternalObject\(\)](#)

5.4.34 ComRed

Overview

[ReductionInMemory](#)
[ResetReductionInMemory](#)

ReductionInMemory

Execute network reduction in memory, no change to database. Note: afterwards, the function `ResetReductionInMemory()` must be called to revert the reduction.

```
int ComRed.ReductionInMemory()
```

RETURNS

Returns 0 if successfully executed.

ResetReductionInMemory

Restore the network back to original after reduction in memory.

```
void ComRed.ResetReductionInMemory()
```

5.4.35 ComRel3

Overview

[AnalyseElmRes](#)
[ExeEvt](#)
[OvlAlleviate](#)
[RemoveEvents](#)
[RemoveOutages](#)
[ValidateConstraints](#)

AnalyseElmRes

Evaluate the results object created by the last calculation. Performs exactly the same as pressing the button 'Perform Evaluation of Result File' in the dialogue box of the command.

```
int ComRel3.AnalyseElmRes([int error])
```

ARGUMENTS

<i>error (optional)</i>	
0	do not display an error message (default)
1	display error messages in case of errors

RETURNS

= 0 On success.

$\neq 0$ On error.

ExeEvt

Executes a given event.

```
None ComRel3.ExeEvt([DataObject event])
```

ARGUMENTS

event The event that shall be executed.

OvlAlleviate

Performs an overload alleviation for given events.

```
int ComRel3.OvlAlleviate([list preCalcEvents])
```

ARGUMENTS

preCalcEvents (optional)
The events which will be executed before the calculation.

RETURNS

0 On success.
1 Failure in load flow.
2 No overloading detected.
> 2 On error.

RemoveEvents

Removes all events stored in all contingencies (*.ComContingency) inside the reliability command.

```
None ComRel3.RemoveEvents()
```

RemoveOutages

Removes all contingency definitions (*.ComContingencies) stored inside the reliability command.

```
None ComRel3.RemoveOutages([int msg])
```

ARGUMENTS

msg(optional)
1 Show info message in output window (default value).
0 Do not emit messages.

ValidateConstraints

Checks if the restoration of a contingency violates any constraint according to the current settings of the reliability calculation. These do not necessarily have to be the settings used during calculation. Of course the selected calculation method of ComRel3 has to be 'Load flow analysis' to check for constraint violations.

```
int ComRel3.ValidateConstraints(DataObject contingency)
```

ARGUMENTS

contingency

The contingency which will be checked for constraint violations.

RETURNS

- 0 No constraint violations, or all constraint violations could be solved.
- 1 Constraints are violated.
- −1 Contingency not valid.

5.4.36 ComRelpost

Overview

[CalcContributions](#)

[GetContributionOfComponent](#)

CalcContributions

Calculates the contributions to load interruptions of the loads that are passed to this function. The loads can be e.g. inside a feeder or a zone as well. If nothing is passed as input all loads will be analysed.

```
int ComRelpost.CalcContributions([list elements])
```

ARGUMENTS

elements (optional)

Elements (Loads) for which the contributions shall be calculated (default: all loads, if no argument is passed).

RETURNS

- 0** Calculation successful.
- 1** On error.

GetContributionOfComponent

Gets the contributions of a component to a certain reliability indice.

```
float ComRelpost.GetContributionOfComponent(int componentNr,  
                                             str indice)
```

ARGUMENTS

componentNr 1. Lines

2. Cables

3. Transformers
4. Busbars
5. Generators
6. Common Modes
7. Double Earth Faults

indice Available indices are: 'SAIFI', 'SAIDI', 'ASIFI', 'ASIDI', 'ENS', 'EIC'

RETURNS

The contribution of this component to this reliability indice.

5.4.37 ComRelreport

Overview

[GetContingencies](#)
[GetContributionOfComponent](#)

GetContingencies

Gets all contingencies of reliability for reporting.

```
list ComRelpost.GetContingencies()
```

RETURNS

All contingencies of reliability for reporting.

GetContributionOfComponent

Is described in [ComRelpost.GetContributionOfComponent\(\)](#).

```
float ComRelreport.GetContributionOfComponent(int componentNr,  
                                              str indice)
```

5.4.38 ComRes

Overview

[ExportFullRange](#)
[FileNmResNm](#)

ExportFullRange

Executes the export command for the whole data range.

```
None ComRes.ExportFullRange()
```

FileNmResNm

Sets the filename for the data export to the name of the result object being exported (classes: ElmRes, IntComtrade)

```
None ComRes.FileNmResNm()
```

5.4.39 ComShc**Overview**

[ExecuteRXSweep](#)
[GetFaultType](#)
[GetOverLoadedBranches](#)
[GetOverLoadedBuses](#)

ExecuteRXSweep

Calculates RX Sweep. If no impedance passed, the value from the command shall be used. If argument passed then the impedance changes are stored to the command (Rf, Xf).

```
int ComShc.ExecuteRXSweep()
int ComShc.ExecuteRXSweep(float Zr,
                           float Zi
                           )
```

ARGUMENTS

Zr Impedance real part
Zi Impedance imaginary part

RETURNS

= 0 On success.
 ≠ 0 On error.

GetFaultType

Returns the short-circuit fault type.

```
int ComShc.GetFaultType()
```

RETURNS

0 three phase fault
1 single phase to ground
2 two phase fault
3 two phase to ground fault
4 three phase unbalanced fault
5 single phase to neutral fault
6 single phase, neutral to ground fault
7 two phase to neutral fault
8 two phase, neutral to ground fault

- 9 three phase to neutral fault
- 10 three phase, neutral to ground fault
- 20 DC fault

GetOverLoadedBranches

Get overloaded branches after a short-circuit calculation.

```
[int error,
list branches] ComShc.GetOverLoadedBranches(float ip,
                                              float ith)
```

ARGUMENTS

- ip* Max. peak-current loading, in %
- ith* Max. thermal loading, in %
- branches (out)*
Set of branches which are checked

RETURNS

- = 0 On error or 0 branches found.
- ≠ 0 Number of branches.

EXAMPLE

GetOverLoadedBuses

Get overloaded buses after a short-circuit calculation.

```
[int error,
list buses] ComShc.GetOverLoadedBuses(float ip,
                                       float ith)
```

ARGUMENTS

- ip* Max. peak-current loading, in %
- ith* Max. thermal loading, in %
- buses (optional, out)*
Set of buses which are checked

RETURNS

- = 0 On error or 0 buses found.
- ≠ 0 Number of buses.

EXAMPLE

5.4.40 ComShctrace

Overview

[BlockSwitch](#)
[ExecuteAllSteps](#)
[ExecuteInitialStep](#)
[ExecuteNextStep](#)
[GetBlockedSwitches](#)
[GetCurrentTimeStep](#)
[GetDeviceSwitches](#)
[GetDeviceTime](#)
[GetNonStartedDevices](#)
[GetStartedDevices](#)
[GetSwitchTime](#)
[GetTrippedDevices](#)
[NextStepAvailable](#)

BlockSwitch

Blocks a switch from operating for the remainder of the trace.

```
int ComShctrace.BlockSwitch(DataObject switchDevice)
```

ARGUMENTS

switchDevice
 Switch device to block.

RETURNS

0 Switch can not be blocked (e.g. because it already operated).
1 Switch is blocked.

ExecuteAllSteps

Executes all steps of the short circuit trace. This function requires the trace to be already running

```
int ComShctrace.ExecuteAllSteps()
```

RETURNS

0 No error occurred, trace is complete.
!=0 An error occurred, calculation was reset.

SEE ALSO

ComShctrce.ExecuteInitialStep()

ExecuteInitialStep

Executes the first step of the short circuit trace.

```
int ComShctrace.ExecuteInitialStep()
```

RETURNS

- 0** No error occurred, the short-circuit trace is now running.
- !=0** An error occurred, calculation was reset.

ExecuteNextStep

Executes the next step of the short circuit trace. This function requires the trace to be already running

```
int ComShctrace.ExecuteNextStep()
```

RETURNS

- 0** No error occurred, step was executed .
- !=0** An error occurred, calculation was reset.

SEE ALSO

ComShctrce.ExecuteInitialStep()

GetBlockedSwitches

Returns all switches which are currently blocked.

```
list ComShctrace.GetBlockedSwitches()
```

RETURNS

All blocked switches.

GetCurrentTimeStep

Returns the current time step of the trace in seconds.

```
int ComShctrace.GetCurrentTimeStep()
```

RETURNS

The current time step in [s].

GetDeviceSwitches

Returns all switches operated by a protection device.

```
list ComShctrace.GetDeviceSwitches(DataObject device)
```

ARGUMENTS

device Protection device to get the switches for.

RETURNS

All switches devices operated by the protection device.

GetDeviceTime

Returns the time a protection device operated or will operate at.

```
int ComShctrace.GetDeviceTime(DataObject device)
```

ARGUMENTS

device Protection device to get the time for.

RETURNS

The tripping time of the device itself, if the device already tripped, or the prospective tripping time.

GetNonStartedDevices

Returns all protection devices which are not started.

```
list ComShctrace.GetNonStartedDevices()
```

RETURNS

All protection devices which are not started.

GetStartedDevices

Returns all started but not yet tripped protection devices.

```
list ComShctrace.GetStartedDevices()
```

RETURNS

All started but not yet tripped protection devices.

GetSwitchTime

Returns the time a switch device operated or will operate at.

```
int ComShctrace.GetSwitchTime(DataObject device,  
                               DataObject switchDevice  
                               )
```

ARGUMENTS

device Reference protection device for the switch.

device Switch device to get the time for.

RETURNS

The tripping time of the switch device, based on the tripping time of the reference protection device. If the switch already operated, the time of operation will be returned.

GetTrippedDevices

Returns all protection devices already tripped.

```
list ComShctrace.GetTrippedDevices()
```

RETURNS

All protection devices already tripped.

NextStepAvailable

Indicates whether or not a next time step can be executed.

```
int ComShctrace.NextStepAvailable()
```

RETURNS

- 0** Next step is not available, the trace is completed.
- 1** A next step is available.

5.4.41 ComSim**Overview**

[GetSimulationTime](#)
[GetTotalWarnA](#)
[GetTotalWarnB](#)
[GetTotalWarnC](#)
[GetViolatedScanModules](#)
[LoadSnapshot](#)
[SaveSnapshot](#)

GetSimulationTime

Get the actual simulation time if the initial conditions are calculated.

```
int ComSim.GetSimulationTime()
```

RETURNS

Returns the simulation time in seconds. If the initial conditions are not calculated, then the function returns 'nan'.

GetTotalWarnA

Returns the total number of type-A (serious) warnings related to the time-domain simulation.

```
int ComSim.GetTotalWarnA()
```

RETURNS

Returns the total number of type-A (serious) warnings.

GetTotalWarnB

Returns the total number of type-B (moderate) warnings related to the time-domain simulation.

```
int ComSim.GetTotalWarnB()
```

RETURNS

Returns the total number of type-B (moderate) warnings.

GetTotalWarnC

Returns the total number of type-C (minor) warnings related to the time-domain simulation.

```
int ComSim.GetTotalWarnC()
```

RETURNS

Returns the total number of type-C (minor) warnings.

GetViolatedScanModules

Returns a set of scan modules of which each have at least one violation.

```
list ComSim.GetViolatedScanModules()
```

RETURNS

Returns a set of scan modules of which each have at least one violation.

LoadSnapshot

Load the state of a dynamic simulation from a file.

```
int ComSim.LoadSnapshot([string snapshotFilePath], [int suppressUserMessage])
```

DEPRECATED NAMES

LoadSimulationState

ARGUMENTS

snapshotFilePath (optional)

The snapshot file to load. If no file is specified, the last snapshot stored in the memory is used. (default = "")

suppressUserMessage (optional)

The pop-up window asking for data overwrite is not displayed if this value is set to 1. (default = 0)

RETURNS

- 0** Saved simulation state has been loaded.
- 1** Saved simulation state cannot be loaded.
- 2** Saved simulation state does not match actual model. Calculation will be reset.

SaveSnapshot

Save the state of a dynamic simulation to memory and to a file.

```
int ComSim.SaveSnapshot(string snapshotFilePath, [int suppressUserMessage])
```

DEPRECATED NAMES

SaveSimulationState

ARGUMENTS

snapshotFilePath (optional)

The snapshot file to save. If no file is specified, the last snapshot is saved in the non-persistent memory slot. (default = ")

suppressUserMessage (optional)

The pop-up window asking for file overwriting is not displayed if this value is set to 1. (default = 0)

RETURNS

0 OK
1 Error

5.4.42 ComSimoutage

Overview

AddCntcy
 AddContingencies
 AddRas
 ClearCont
 CreateFaultCase
 Execute
 ExecuteAndCheck
 GetNTopLoadedElms
 MarkRegions
 RemoveAllRas
 RemoveContingencies
 RemoveRas
 Reset
 SetLimits
 Update

AddCntcy

Executes an (additional) ComOutage, without resetting results. The results of the outage analysis will be added to the intermediate results. Object "O" must be a ComOutage object. If the outage definition has already been analyzed, it will be ignored. The ComOutage will be renamed to "name" when "name" is given.

```
int ComSimoutage.AddCntcy(DataObject O,
                          [str name]
                          )
```

ARGUMENTS

O The ComOutage object
name A name for the outage

RETURNS

0 On success.

1 On error.

AddContingencies

Adds contingencies for fault cases/groups selected by the user to the command. Shows a modal window with the list of available fault cases and groups. Functionality as “Add Cases/Groups” button in dialog.

```
None ComSimoutage.AddContingencies()
```

AddRas

Adds a reference to a given IntRas to the ComSimoutage.

```
int ComSimoutage.AddRas(DataObject ras)
```

ARGUMENTS

ras The IntRas object

RETURNS

- 0** If the reference to the IntRas has been added.
- 1** If the reference to the IntRas has already been there or on error.

ClearCont

Reset existing contingency analysis results and delete existing contingency cases.

```
int ComSimoutage.ClearCont()
```

RETURNS

- 0** On success.
- 1** On error.

CreateFaultCase

Create fault cases from the given elements.

```
int ComSimoutage.CreateFaultCase(list elms,
                                int mode,
                                [int createEvt],
                                [DataObject folder]
                                )
```

ARGUMENTS

elms Selected elements to create fault cases.

mode How the fault cases are created:

- 0** Single fault case containing all elements.
- 1** n-1 (multiple cases).
- 2** n-2 (multiple cases).

- 3** Collecting coupling elements and create fault cases for line couplings.

createEvt (optional)

Switch event:

- 0** Do NOT create switch events.
1 Create switch events.

folder (optional)

Folder in which the fault case is stored.

RETURNS

- 0** On success.
1 On error.

Execute

Execute contingency analysis.

```
int ComSimoutage.Execute()
```

RETURNS

- 0** On success.
1 On error.

ExecuteAndCheck

Executes contingency analysis and checks the violated constraints. The constraints are defined in models, such as maximum and minimum voltage of buses, maximum loading of lines etc. On the first occurred constraint violation, the further contingency execution will be terminated, making the calculation faster, but with less information.

```
int ComSimoutage.ExecuteAndCheck([int workMode,]
                                   [int initMode,]
                                   [int considerOPFflags,]
                                   [int considerVstep]
                                   )
```

ARGUMENTS

workMode (optional)

the mode how the contingency analysis is executed; default is 0

- 0** Normal execution (doesn't check limits, write results file)
1 Check constraints (thermal loading, bus voltage range) and terminate execution if violates any limit, writes result file
2 Check constraints (thermal loading, bus voltage range) and terminate execution if violates any limit, does not write result file
3 Check thermal loading only and terminate execution if violates any limit, writes result file
4 Check thermal loading only and terminate execution if violates any limit, does not write result file
5 Check voltage limits only and terminate execution if violates any limit, writes result file

- 6** Check voltage limits only and terminate execution if violates any limit, does not write result file
- 7** Neither loading nor voltage limits are checked and terminate execution if any contingency fails, writes result file
- 8** Neither loading nor voltage limits are checked and terminate execution if any contingency fails, does not write result file

initMode (optional)

the mode how the contingency analysis is initialised; default is 0

- 0** full initialisation, new topology rebuild, and update all contingencies
- 1** new topology rebuild, but skip updating all contingencies
- 2** no topology rebuild, and no contingency update, which is the fastest mode

considerOPFflags (optional)

whether the voltage limit settings on OPF page will be considered or not; default is 1

- 0** the settings on OPF page are not considered.
- 1** the voltage limits are considered only when the check boxes on OPF page are ticked.

considerVstep (optional)

whether the voltage step limits will be considered or not; default is 1

- 0** the voltage step limits are not considered.
- 1** the voltage step limits are considered.

RETURNS

- 0** contingencies successfully executed without any violation
- 1** calculation was interrupted by the user
- 2** error occurred during contingency analysis
- 3** initialisation failed, such as base case load flow diverged
- 4** for AC/DC combined method, base case is already critical and calculation stopped
- 5** any loading constraint violated
- 6** any contingency cannot be analysed, non-convergent case
- 7** any voltage constraint violated
- 8** any constraint violated already in the base case

GetNTopLoadedElms

To get certain number of top loaded components (most close to its limit).

```
list ComSimoutage.GetNTopLoadedElms(int number)
```

ARGUMENTS

number The number of elements to be found.

elements (out)

The top loaded elements.

MarkRegions

To execute Region marker for certain system status (like prefault, post fault etc.), which will identifies energizing mode for each element.

```
int ComSimoutage.MarkRegions(int stage)
```

ARGUMENTS

stage which system stage to be analyzed, 0=*stage*₁=2

RETURNS

0 On success.
1 On error.

RemoveAllRas

Removes all IntRas from to the ComSimoutage. Only deletes the references not the IntRas itself.

```
None ComSimoutage.RemoveAllRas()
```

RemoveContingencies

Removes all contingencies from the command. Functionality as "Remove All" button in dialog.

```
None ComSimoutage.RemoveContingencies()
```

RemoveRas

Removes the reference to a given IntRas from the ComSimoutage.

```
int ComSimoutage.RemoveRas(DataObject ras)
```

ARGUMENTS

ras The IntRas object

RETURNS

0 If the reference to the IntRas has been successfully removed.
1 If the reference to the IntRas has not been in the container or on error.

Reset

Resets the intermediate results of the outage simulation.

```
int ComSimoutage.Reset()
```

RETURNS

0 On success.
1 On error.

SetLimits

Sets the recording limits for the Contingency Analysis.

```
None ComSimoutage.SetLimits(float vmin,
                             float vmax,
                             float ldmax)
```

ARGUMENTS

<i>vmin</i>	The minimum voltage
<i>vmax</i>	The maximum voltage
<i>ldmax</i>	The maximum loading

Update

To update contingency cases via topology search. It will find interrupted elements, required switch actions for each contingency.

```
int ComSimoutage.Update()
```

RETURNS

0	On success.
1	On error.

5.4.43 ComSvgexport

Overview

[SetFileName](#)
[SetObject](#)
[SetObjects](#)

SetFileName

Sets SVG file for export.

```
None ComSvgexport.SetFileName(str path)
```

ARGUMENTS

<i>path</i>	Path of target SVG file
-------------	-------------------------

SetObject

Sets annotation layer or group for export.

```
None ComSvgexport.SetObject(DataObject obj)
```

ARGUMENTS

<i>obj</i>	Annotation layer (IntGrflayer) or group (IntGrfgroup) to be exported
------------	--

SetObjects

Sets annotation layers and groups for export.

```
None ComSvgexport.SetObjects(set objs)
```

ARGUMENTS

objs Set of annotation layers (IntGrflayer) and/or groups (IntGrfgroup) to be exported

5.4.44 ComSvgimport

Overview

[SetFileName](#)
[SetObject](#)

SetFileName

Sets source SVG file for import.

```
None ComSvgimport.SetFileName(str path)
```

ARGUMENTS

path Path of SVG file to be imported

SetObject

Sets target annotation layer or group for import.

```
None ComSvgimport.SetObject(DataObject obj)
```

ARGUMENTS

obj Target annotation layer (IntGrflayer) or group (IntGrfgroup)

5.4.45 ComTasks

Overview

[AppendCommand](#)
[AppendStudyCase](#)
[GetCommandsForStudyCase](#)
[GetNumberOfCommandsForStudyCase](#)
[GetNumberOfStudyCases](#)
[GetStudyCases](#)
[IsAdditionalResultsFlagSetForCommand](#)
[IsCommandIgnored](#)
[IsStudyCaseIgnored](#)
[RemoveCmdsForStudyCaseRow](#)
[RemoveCommand](#)
[RemoveStudyCase](#)
[RemoveStudyCases](#)

[SetAdditionalResultsFlagForCommand](#)
[SetIgnoreFlagForCommand](#)
[SetIgnoreFlagForStudyCase](#)
[SetResultsFolder](#)

AppendCommand

Appends a command for calculation.

```
int ComTasks.AppendCommand(DataObject command,
                           [int studyCaseRow]
                           )
```

RETURNS

- 0** Command could not be added for calculation.
- 1** Command has been successfully added for calculation.

ARGUMENTS

command

Command to add for calculation.

studyCaseRow

- ≤ 0 Command is added to the list of commands for its study case.
- > 0 Optionally, the row in the study case table containing the study case for which this command shall be added can be passed. This is helpful, e.g., if a study case has been added multiple times for calculation with different command lists.

AppendStudyCase

Appends a study case to the list of study cases for calculation.

```
int ComTasks.AppendStudyCase(DataObject studyCase)
```

RETURNS

- 0** Study case could not be added for calculation.
- 1** Study case has been successfully added for calculation.

ARGUMENTS

studyCase

Study case to add for calculation.

GetCommandsForStudyCase

Get all selected commands to be processed for a study case from the Task Automation command.

```
list ComTasks.GetCommandsForStudyCase(DataObject studyCase,
                                       [int studyCaseRow])
```

RETURNS

Returns ordered set of all selected commands to be processed for a study case.

ARGUMENTS

studyCase

Study case to get all selected commands to be processed from.

studyCaseRow

- ≤ 0 Commands for the first matching study case found will be returned.
- > 0 Optionally, the row in the study case table containing the study case for which commands shall be returned can be passed. This is helpful, e.g., if a study case has been added multiple times for calculation with different command lists.

GetNumberOfCommandsForStudyCase

Get number of all selected commands to be processed for a study case from the Task Automation command.

```
int ComTasks.GetNumberOfCommandsForStudyCase(DataObject studyCase,
                                             [int studyCaseRow])
```

RETURNS

Returns number of all selected commands to be processed for a study case.

ARGUMENTS

studyCase

Study case to get number of selected commands to be processed from.

studyCaseRow

- ≤ 0 Number of commands for the first matching study case found will be returned.
- > 0 Optionally, the row in the study case table containing the study case for which number of commands shall be returned can be passed. This is helpful, e.g., if a study case has been added multiple times for calculation with different command lists.

GetNumberOfStudyCases

Get number of selected study cases from the Task Automation command.

```
int ComTasks.GetNumberOfStudyCases()
```

RETURNS

Returns number of selected study cases from the Task Automation command.

GetStudyCases

Get all selected study cases from the Task Automation command.

```
list ComTasks.GetStudyCases()
```

RETURNS

Returns ordered set of all selected study cases from the Task Automation command, set is empty on failure.

IsAdditionalResultsFlagSetForCommand

Returns whether additional results flag of a command is set or not (using study case row and task row / using a command directly).

```
int ComTasks.IsAdditionalResultsFlagSetForCommand(int studyCaseRow,
                                                  int taskRow)
int ComTasks.IsAdditionalResultsFlagSetForCommand(DataObject command)
```

ARGUMENTS

studyCaseRow ≤ 0

The row in the study cases table of the command's study case for which the additional results flag shall be derived.

taskRow > 0

The row in the commands table for which the additional results flag shall be derived.

command

The command to get the additional results flag for.

RETURNS

- 0** Additional results flag of command is not set.
- 1** Additional results flag of command is set.
- 1** Additional results flag was not found for provided study case row and task row / provided command.

IsCommandIgnored

Returns whether the processing of a command will be ignored or not (using study case row and task row / using provided command).

```
int ComTasks.IsCommandIgnored(int studyCaseRow,
                              int taskRow)
int ComTasks.IsCommandIgnored(DataObject command)
```

ARGUMENTS

studyCaseRow ≤ 0

The row in the study cases table of the command's study case for which the ignore flag shall be derived.

taskRow > 0

The row in the commands table for which the ignore flag shall be derived.

command

The command to get the ignore-flag for.

RETURNS

- 0** Processing of command will be done.
- 1** Processing of command will be ignored.
- 1** Command was not found (for provided study case row and task row / at all).

IsStudyCaseIgnored

Returns whether the command processing for a study case will be ignored or not.

```
int ComTasks.IsStudyCaseIgnored(DataObject studyCase,
                                [int studyCaseRow])
```

RETURNS

- 0** Command processing for study case will be done.
- 1** Command processing for study case will be ignored.
- 1** study case or row was not found.

ARGUMENTS

studyCase

Study case to get ignore-flag from.

studyCaseRow

- ≤ 0 Ignore flag for the first matching study case found will be returned.
- > 0 Optionally, the row in the study case table containing the study case for which ignore-flag shall be returned can be passed. This is helpful, e.g., if a study case has been added multiple times for calculation with different command lists.

RemoveCmdsForStudyCaseRow

Removes all commands selected for calculation for a given row in the study case table.

```
int ComTasks.RemoveCmdsForStudyCaseRow(int studyCaseRow)
```

RETURNS

- 0** Commands could not be removed from calculation.
- 1** All commands of study case row were successfully removed from calculation.

ARGUMENTS

studyCaseRow > 0

The row in the study case table containing the study case for which all commands shall be removed.

RemoveCommand

Remove a command from the calculation (for study case row and task row / for all appearances).

```
int ComTasks.RemoveCommand(int studyCaseRow,
                            int taskRow
                            )
int ComTasks.RemoveCommand(DataObject command)
```

ARGUMENTS

studyCaseRow > 0

The row in the study cases table of the command's study case for which the command shall be removed.

taskRow > 0

The row in the commands table for which the command shall be removed.

command

Command to remove.

RETURNS

- 0** Command could not be removed.
- 1** Command has been successfully removed.

RemoveStudyCase

Removes a study case from the study cases table.

```
int ComTasks.RemoveStudyCase(DataObject studyCase,
                             [int studyCaseRow]
                             )
```

RETURNS

- 0** Study case could not be removed.
- 1** Study case has been successfully removed.

ARGUMENTS

studyCase

Study case which shall be removed.

studyCaseRow

- ≤ 0 Study case is removed for all entries in the study cases table matching the provided study case.
- > 0 Optionally, the row in the study case table containing the study case which shall be removed can be passed. This is helpful, e.g., if a study case has been added multiple times for calculation with different command lists.

RemoveStudyCases

Removes all study cases from calculation.

```
None ComTasks.RemoveStudyCases()
```

SetAdditionalResultsFlagForCommand

Set the flag whether to record an additional result file for a command of the calculation (using study case row and task row / using command).

```

int ComTasks.SetAdditionalResultsFlagForCommand(int studyCaseRow,
                                                int taskRow,
                                                int addResVal
                                                )
int ComTasks.SetAdditionalResultsFlagForCommand(DataObject command,
                                                int addResVal
                                                )

```

ARGUMENTS

studyCaseRow > 0

The row in the study cases table of the command's study case for which the flag to record additional results shall be set.

taskRow > 0

The row in the commands table for which the flag to record additional results shall be set.

command

The command for which the flag to record additional results shall be set.

addResVal

1 Will set the command to record additional results for the calculation.

0 Will set the command to avoid recording of results for calculation.

RETURNS

0 Flag to record (or not record) additional results could not be set.

1 Flag to record (or not record) additional results has been successfully set.

SetIgnoreFlagForCommand

Set the flag whether to ignore a command for the calculation (using study case row and task row / using command).

```

int ComTasks.SetIgnoreFlagForCommand(int studyCaseRow,
                                      int taskRow,
                                      int ignoreVal
                                      )
int ComTasks.SetIgnoreFlagForCommand(DataObject command,
                                      int ignoreVal
                                      )

```

ARGUMENTS

studyCaseRow > 0

The row in the study cases table of the command's study case for which the ignore flag shall be set.

taskRow > 0

The row in the commands table for which the ignore flag shall be set.

command

Command for which the ignore flag shall be set.

ignoreVal

1 Will set the command to be ignored for calculation.

0 Will set the command to be processed for calculation.

RETURNS

- 0** Ignore flag could not be set.
- 1** Ignore flag has been successfully set.

SetIgnoreFlagForStudyCase

Set the flag whether to ignore the processing of commands of a study case for the calculation.

```
int ComTasks.SetIgnoreFlagForStudyCase(DataObject studyCase,
                                       int ignoreVal,
                                       [int studyCaseRow]
                                       )
```

RETURNS

- 0** Ignore flag could not be set.
- 1** Ignore flag has been successfully set.

ARGUMENTS

studyCase

Study case for which the ignore flag shall be set.

ignoreVal

- 1 Will set the flag to ignore the processing of commands of a study case.
- 0 Will set the flag to process the commands of a study case.

studyCaseRow

- ≤ 0 Ignore flag is set for all entries in the study cases table matching the provided study case.
- > 0 Optionally, the row in the study case table containing the study case for which the ignore-flag shall be set can be passed. This is helpful, e.g., if a study case has been added multiple times for calculation with different command lists.

SetResultsFolder

Set a folder to store results for a given row in the study case table.

```
int ComTasks.SetResultsFolder(DataObject folder, int studyCaseRow)
```

RETURNS

- 0** New folder could not be set as results folder for the given row in the study case table.
- 1** Folder was successfully set as results folder for given row in the study case table.

ARGUMENTS

folder The new folder to store results in.

studyCaseRow

The row in the study case table containing the study case for which results folder shall be set.

5.4.46 ComTececo

Overview

[UpdateTablesByCalcPeriod](#)

UpdateTablesByCalcPeriod

Update all calculation points with respect to a new start- and end year

```
int ComTececo.UpdateTablesByCalcPeriod(float start,
                                       float end
                                       )
```

ARGUMENTS

start Start year of the study period
end End year of the study period

RETURNS

0 Calculation points have been successfully set.
1 Invalid input data: end year of study period must be greater or equal to start year.

5.4.47 ComTransfer

Overview

[GetTransferCalcData](#)
[IsLastIterationFeasible](#)

GetTransferCalcData

The function returns the calculated transfer capacity and the total number of iteration after the transfer capacity command has been executed.

```
[float transferCapacity,
 int totalIterations ] ComTransfer.GetTransferCalcData()
```

ARGUMENTS

transferCapacity (out)
 Transfer capacity value at the last feasible iteration.
totalIterations (out)
 Total iteration number.

IsLastIterationFeasible

The function verifies if the last transfer calculation iteration resulted in a feasible solution or not.

```
int ComTransfer.IsLastIterationFeasible()
```

RETURNS

- 1** Last transfer calculation iteration resulted in a feasible solution.
- 0** Last transfer calculation iteration did not result in a feasible solution.

5.4.48 ComUcte

Overview

[SetBatchMode](#)

SetBatchMode

The batch mode allows to suppress all messages except error and warnings. This can be useful when used in scripts where additional output might be confusing.

```
None ComUcte.SetBatchMode(int enabled)
```

ARGUMENTS

enabled

- 0** disables batch mode, all messages are printed to output window (default).
- 1** enables batch mode, only error and warning messages are printed to output window.

5.4.49 ComUcteexp

Overview

[BuildNodeNames](#)
[DeleteCompleteQuickAccess](#)
[ExportAndInitQuickAccess](#)
[GetConnectedBranches](#)
[GetFromToNodeNames](#)
[GetOrderCode](#)
[GetUcteNodeName](#)
[InitQuickAccess](#)
[QuickAccessAvailable](#)
[ResetQuickAccess](#)
[SetGridSelection](#)

BuildNodeNames

Builds the node names as used in UCTE export and makes them accessible via :UcteNodeName attribute. The node names will only be available as long as topology has not been changed. They must be re-build after any topology relevant modification.

Furthermore, the method fills the quick access cache given by the cache index for node names and branch topologies as used in UCTE export. The quick access cache endures also topology changes. The cache index is optional. If no cache index is given the default quick access cache is used.

```
int ComUcteexp.BuildNodeNames([int cacheIndex])
```

ARGUMENTS

cacheIndex (optional)

Index of the quick access cache (must be greater than or equals to 0)

RETURNS

- 0** On success.
- 1** On error (e.g. load flow calculation failed).

DeleteCompleteQuickAccess

Deletes all quick access caches.

```
None ComUcteexp.DeleteCompleteQuickAccess()
```

ExportAndInitQuickAccess

Performs an UCTE export and fills the quick access cache given by the cache index.

```
None ComUcteexp.ExportAndInitQuickAccess(int cacheIndex)
```

ARGUMENTS

cacheIndex

Index of the quick access cache (must be greater than or equals to 0)

GetConnectedBranches

Determines the connected branches for the given terminal from the quick access cache given by the optional cache index. If no cache index is given the default quick access cache is used.

```
list ComUcteexp.GetConnectedBranches(DataObject terminal,
                                     [int cacheIndex])
```

ARGUMENTS

terminal Terminal to determine the connected branches from*connectedBranches (out)*

Connected branches for the given terminal

cacheIndex (optional)

Index of the quick access cache (must be greater than or equals to 0)

GetFromToNodeNames

Determines the UCTE node names of the branch ends from the quick access cache given by the optional cache index. If no cache index is given the default quick access cache is used.

```
[str nodeNameFrom,
str nodeNameTo ] ComUcteexp.GetFromToNodeNames(DataObject branch,
                                               [int cacheIndex])
```

ARGUMENTS

branch Branch to find the UCTE node names from

nodeNameFrom (out)
UCTE node name of start node

nodeNameTo (out)
UCTE node name of end node

cacheIndex (optional)
Index of the quick access cache (must be greater than or equals to 0)

GetOrderCode

Determines the order code of the given branch element as used for UCTE export from the quick access cache given by the optional cache index. If no cache index is given the default quick access cache is used.

```
str ComUcteexp.GetOrderCode(DataObject branch,  
                             [int cacheIndex])
```

ARGUMENTS

branch Branch element to get the UCTE order code from

orderCode (out)
Order code of the given branch element

cacheIndex (optional)
Index of the quick access cache (must be greater than or equals to 0)

GetUcteNodeName

Determines the node name of the given terminal as used for UCTE export from the quick access cache given by the optional cache index. If no cache index is given the default quick access cache is used.

```
None ComUcteexp.GetOrderCode(DataObject terminal,  
                              [int cacheIndex])
```

ARGUMENTS

terminal Terminal to get the UCTE node name from

ucteNodeName (out)
UCTE node name of the given terminal

cacheIndex (optional)
Index of the quick access cache (must be greater than or equals to 0)

InitQuickAccess

Initializes the quick access cache given by the optional cache index. The quick access cache contains node names and branch topologies as used in UCTE export and endures topology changes. *InitQuickAccess()* requires a successful executed UCTE export as pre-condition. The cache index is optional. If no cache index is given the default quick access cache is used.

```
None ComUcteexp.InitQuickAccess([int cacheIndex])
```


ARGUMENTS

cacheIndex (optional)

Index of the quick access cache (must be greater than or equals to 0)

QuickAccessAvailable

Checks if the quick access cache given by the optional cache index is available. If no cache index is given the default quick access cache is checked for availability.

```
int ComUcteexp.QuickAccessAvailable([int cacheIndex])
```

ARGUMENTS

cacheIndex (optional)

Index of the quick access cache (must be greater than or equals to 0)

RETURNS

0 on success and 1 on error.

ResetQuickAccess

Resets the given quick access cache for node names and branch topologies as used in UCTE export. The cache index is optional. If no cache index is given the default quick access cache is reset.

```
None ComUcteexp.ResetQuickAccess([int cacheIndex])
```

ARGUMENTS

cacheIndex (optional)

Index of the quick access cache (must be greater than or equals to 0)

SetGridSelection

Configures the selected grids in the UCTE export command.

```
None ComUcteexp.SetGridSelection(list gridsToExport)
```

ARGUMENTS

gridsToExport

Grids (instances of class ElmNet) to be selected for export. All not contained grids will be de-selected.

5.4.50 ComWktimp**Overview**

[GetCreatedObjects](#)
[GetModifiedObjects](#)

GetCreatedObjects

Returns the newly created objects after execution of a WKT import.

```
list ComWktimp.GetCreatedObjects()
```

RETURNS

Collection of objects that have been created during WKT import.

GetModifiedObjects

Returns the modified objects after execution of a WKT import.

```
list ComWktimp.GetModifiedObjects()
```

RETURNS

Collection of objects that have been modified during WKT import.

5.5 Settings

5.5.1 SetCluster

Overview

[CalcCluster](#)
[GetNumberOfClusters](#)

CalcCluster

Performs a load flow calculation for the cluster index passed to the function. To execute properly this function requires that a valid load flow result is already calculated before calling it.

```
int SetCluster.CalcCluster(int clusterIndex,  
                           [int messageOn]  
                           )
```

ARGUMENTS

clusterIndex

The cluster index. Zero based value, the first cluster has index 0.

messageOn (optional)

Possible values:

- | | |
|----------|--|
| 0 | Do not emit a message in the output window. |
| 1 | Emit a message in the output window in case that the function does not execute properly. |

RETURNS

- | | |
|----------|---|
| 0 | On success. |
| 1 | There are no clusters, the number of clusters is 0. |
| 2 | The cluster index exceeds the number of clusters. |
| 3 | There is no load flow in memory before running CalcCluster. |

GetNumberOfClusters

Get the number of clusters.

```
int SetCluster.GetNumberOfClusters()
```

RETURNS

The number of clusters.

5.5.2 SetColscheme

Overview

[CreateFilter](#)
[SetColouring](#)
[SetFilter](#)

CreateFilter

Creates filter used to determine objects to be colored.

```
int SetColscheme.CreateFilter([int pageNr])
```

ARGUMENTS

<i>pageNr</i>	empty	Create filter for currently valid calculation
	set	Dialog page number for which filter is created (see table below)

Table 5.5.3

Dialog Page Name	"pageNr" value
Basic Data	101
Load Flow	102
AC Load Flow Sensitivities	120
AC Contingency Analysis	121
AC Quasi-dynamic Simulation	137
DC Load Flow	122
DC Load Flow Sensitivities	123
DC Contingency Analysis	124
DC Quasi-dynamic Simulation	138
VDE/IEC Short-Circuit	103
Complete Short-Circuit	111
ANSI Short-Circuit	112
IEC 61363	114
DC Short-Circuit	117
RMS-Simulation	104
Modal Analysis	128
EMT-Simulation	105
Harmonics/Power Quality	106
Frequency Sweep	127
D-A-CH-CZ Connection Request	139
BDEW/VDE Connection Request	142
Optimal Power Flow	108
DC Optimal Power Flow	130
DC OPF with Contingencies	135
State Estimation	113
Reliability	109
General Adequacy	115
Tie Open Point Opt.	116
Motor Starting Calculation	133
Arc Flash Calculation	129
Optimal Capacitor Placement	126
Voltage Profile Optimisation	125
Backbone Calculation	131
Optimal RCS Placement	132
Optimal Manual Restoration	136
Phase Balance Optimisation	141
User defined calculation	142

RETURNS

- 0** On success.
- 1** On error.

SetColouring

Sets colouring for given or currently valid calculation.

```
int SetColscheme.SetColouring(str page,
                              int energizing,
                              [int alarm,]
                              [int normal]
                              )
```

ARGUMENTS

page

- empty** set for currentlx valid calculation
set page for which modes are set (see table below)

energizing

Colouring for Energizing Status

- 2** enable (set to previously selected mode),
-1 do not change
0 disable
>0 set to this mode (see table below)

alarm

Colouring for Alarm

- 2** enable (set to previously selected mode),
-1 do not change (default)
0 disable
>0 set to this mode (see table below)

normal

Other Colouring

- 2** enable (set to previously selected mode),
-1 do not change (default)
0 disable
>0 set to this mode (see table below)

Table 5.5.4

Dialog Page Name	"page" value
Basic Data	basic
Load Flow	ldf
AC Load Flow Sensitivities	acsens
AC Contingency Analysis	accont
AC Quasi-dynamic Simulation	acldfsweep
DC Load Flow	dcldf
DC Load Flow Sensitivities	dcsens
DC Contingency Analysis	dccont
DC Quasi-dynamic Simulation	dcldfsweep
VDE/IEC Short-Circuit	shc
Complete Short-Circuit	shcfull
ANSI Short-Circuit	shcansi
IEC 61363	shc61363
DC Short-Circuit	shcdc
RMS-Simulation	rms
Modal Analysis	modal
EMT-Simulation	emt
Harmonics/Power Quality	harm
Frequency Sweep	fsweep
D-A-CH-CZ Connection Request	dachcz
BDEW/VDE Connection Request	bdewvde
Optimal Power Flow	opf
DC Optimal Power Flow	dcopf
DC OPF with Contingencies	dccontopf
State Estimation	est
Reliability	rel
General Adequacy	genrel
Tie Open Point Opt.	topo
Motor Starting Calculation	motstart
Arc Flash Calculation	arcflash
Optimal Capacitor Placement	optcapo
Voltage Profile Optimisation	mvplan
Backbone Calculation	backbone
Optimal RCS Placement	optrcs
Optimal Manual Restoration	omr
Phase Balance Optimisation	balance
Hosting Capacity Analysis	hostcap
User defined calculation	usercalc

Table 5.5.5

Energizing State Name	"energizing" value
De-energized	33
Out of Calculation	37
De-energised, Planned Outage	61

Table 5.5.6

Alarm Name	"alarm" value
Voltage Violations / Overloading	29
Outages	31
Overloading of Thermal / Peak Short Circuit Current	32
Feeder Radiality Check	38

Table 5.5.7

Other Colouring Name	Group	"normal" value
Voltages / Loading	Results	1
Voltage Levels	Topology	2
Individual	Individual	4
Connected Grid Components	Topology	5
According to Filter	User-defined	see notes below table
Grids	Groupings	7
Modifications in Variations / System Stages	Variations / System Stages	8
Loading of Thermal / Peak Short-Circuit Current	Results	9
Paths	Groupings	10
System Type AC/DC and Phases	Topology	11
Relays, Current and Voltage Transformers	Secondary Equipment	12
Fault Clearing Times	Results	13
Feeders	Topology	14
Switches, Type of Usage	Secondary Equipment	15
Measurement Locations	Secondary Equipment	16
Missing graphical connections	Topology	17
Zones	Groupings	18
State Estimation	Results	19
Boundaries (Interior Region)	Topology	20
Station Connectivity	Topology	21
Outage Check	Topology	22
Energizing Status	Topology	23
Modifications in Recording Expansion Stage	Variations / System Stages	24
Areas	Groupings	25
Owners	Groupings	26
Routes	Groupings	27
Operators	Groupings	28
Original Locations	Variations / System Stages	30
Boundaries (Definition)	Topology	34
Meteo Stations	Groupings	35
Station Connectivity (Beach Balls only)	Topology	36
Power Restoration	Secondary Equipment	43
Connected Components	Topology	39
Connected Components, Voltage Level	Topology	40
Year of Construction	Primary Equipment	41
Cross Section	Primary Equipment	42
Forced Outage Rate	Primary Equipment	44
Forced Outage Duration	Primary Equipment	45
Loads: Yearly interruption frequency	Results	46
Loads: Yearly interruption time	Results	47
Loads: Average Interruption Duration	Results	48
Loads: Load Point Energy Not Supplied	Results	49
Supplied by Substation	Topology	50
Supplied by Secondary Substation	Topology	51
Incident Energy	Results	52
PPE-Category	Results	53
Optimal Manual Restoration	Results	54
Connection Request: Approval Status	Results	55
Voltage Angle	Results	56
Contributions to SAIDI	Results	57
Contributions to SAIFI	Results	58
Contributions to ENS	Results	59
Contributions to EIC	Results	60

Note: User-defined filters can be set with a “normal” value of 1000 or higher. The first filter in the list has the value 1000, the next one has 1001 and so on.

RETURNS

- 0** error (at least one of the given colourings cannot be set, e.g. not available for given page). Nothing is changed.
- 1** ok

SetFilter

Sets filter for given or currently valid calculation.

```
int SetColscheme.SetFilter(int filter,
                           [int page]
                           )
int SetColscheme.SetFilter(DataObject obj,
                           [int page]
                           )
```

ARGUMENTS

- filter* number of filter to be set
- obj* user-defined filter to be set
- page (optional)*
Dialog page number for which filter is set (for numbers see table listed in [SetColscheme.CreateFilter\(\)](#))

RETURNS

- 0** ok
- 1** error (filter or page not found)

SEE ALSO

[SetColscheme.CreateFilter\(\)](#)

5.5.3 SetDataext

Overview

[AddConfiguration](#)
[GetConfiguration](#)
[GetConfigurations](#)
[RemoveAllConfigurations](#)
[RemoveConfiguration](#)

AddConfiguration

Adds a new IntAddonvars configuration object for the given classFilter with the descriptiveName as object name. For the classFilter expressions like Elm* or ElmTr? are possible. If there already is an object matching classFilter and descriptiveName exactly, this object is returned instead.

```
DataObject SetDataext.AddConfiguration(str classFilter, str descriptiveName)
```

ARGUMENTS

classFilter

The class filter of the IntAddonvars object

descriptiveName

The object name of the IntAddonvars object

RETURNS

The IntAddonvars object which exactly matches the classFilter and descriptive name or a newly created one.

SEE ALSO

[IntAddonvars.AddDouble\(\)](#), [IntAddonvars.AddDoubleMatrix\(\)](#), [IntAddonvars.AddDoubleVector\(\)](#), [IntAddonvars.AddInteger\(\)](#), [IntAddonvars.AddIntegerVector\(\)](#), [IntAddonvars.AddObject\(\)](#), [IntAddonvars.AddObjectVector\(\)](#), [IntAddonvars.AddString\(\)](#), [IntAddonvars.RemoveParameter\(\)](#)

GetConfiguration

Returns the IntAddonvars object which exactly matches the classFilter and descriptiveName, if the latter is specified. If there are multiple matches the first object will be returned. Otherwise nothing is returned.

```
DataObject SetDataext.GetConfiguration(str classFilter, [str descriptiveName])
```

ARGUMENTS

classFilter

The class filter of the IntAddonvars object

descriptiveName (optional)

The object name of the IntAddonvars object

RETURNS

The IntAddonvars object which exactly matches the classFilter and optionally the descriptiveName or nothing.

SEE ALSO

[IntAddonvars.AddDouble\(\)](#), [IntAddonvars.AddDoubleMatrix\(\)](#), [IntAddonvars.AddDoubleVector\(\)](#), [IntAddonvars.AddInteger\(\)](#), [IntAddonvars.AddIntegerVector\(\)](#), [IntAddonvars.AddObject\(\)](#), [IntAddonvars.AddObjectVector\(\)](#), [IntAddonvars.AddString\(\)](#), [IntAddonvars.RemoveParameter\(\)](#)

GetConfigurations

Returns all IntAddonvars objects by a given classFilter.

```
list SetDataext.GetConfigurations(str classFilter)
list SetDataext.GetConfigurations()
```

ARGUMENTS

classFilter

The class filter for the IntAddonvars object

RETURNS

A list of IntAddonvars objects matching the classFilter exactly. If no filter is specified all IntAddonvars are returned.

SEE ALSO

[IntAddonvars.AddDouble\(\)](#), [IntAddonvars.AddDoubleMatrix\(\)](#), [IntAddonvars.AddDoubleVector\(\)](#), [IntAddonvars.AddInteger\(\)](#), [IntAddonvars.AddIntegerVector\(\)](#), [IntAddonvars.AddObject\(\)](#), [IntAddonvars.AddObjectVector\(\)](#), [IntAddonvars.AddString\(\)](#), [IntAddonvars.RemoveParameter\(\)](#)

RemoveAllConfigurations

Removes all IntAddonvars objects effectively removing all Data Extensions.

```
None SetDataext.RemoveAllConfigurations()
```

RemoveConfiguration

Removes the IntAddonvars object exactly matching classFilter and descriptive name.

```
None SetDataext.RemoveConfiguration(str classFilter, [str descriptiveName])
```

ARGUMENTS

classFilter

The class filter of the IntAddonvars object

descriptiveName (optional)

The object name of the IntAddonvars object

5.5.4 SetDeskpage**Overview**

[Close](#)

[Show](#)

Close

Closes the graphic page, if currently shown in the graphics board.

```
int SetDeskpage.Close()
```

RETURNS

- 0** On success, no error occurred.
- 1** Otherwise

Show

Displays the diagram page in the graphics board.

```
int SetDeskpage.Show()
```

RETURNS

- 0** On success, no error occurred.
- 1** Otherwise

5.5.5 SetDesktop

Overview

[AddPage](#)
[Close](#)
[DoAutoScaleX](#)
[Freeze](#)
[GetActivePage](#)
[GetCanvasSize](#)
[GetPage](#)
[IsFrozen](#)
[IsOpened](#)
[RemovePage](#)
[SetAdaptX](#)
[SetAutoScaleX](#)
[SetResults](#)
[SetScaleX](#)
[SetXVar](#)
[Show](#)
[Unfreeze](#)
[WriteWMF](#)
[ZoomAll](#)

AddPage

Adds an existing page to a graphics and activates it

- Opens the graphics board if not already open.
- Adds the page if it is not already part of the graphics board.

```
DataObject SetDesktop.AddPage(DataObject page2add)
```

ARGUMENTS

page2add

The page to add to the desktop.

- Page is a plot page (GrpPage or SetVipage): A copy of the page is added.
- Page is an IntGrfnet (Single line graphic, block diagram): The graphic is added.

RETURNS

The page displayed or None if the desktop was not changed.

Close

Closes the graphics board, if it is currently shown.

```
int SetDesktop.Close()
```

RETURNS

0	on success
1	on error

DoAutoScaleX

Scales the x-axes of all plots in the graphics board which use the x-axis scale defined in the graphics board.

```
None SetDesktop.DoAutoScaleX()
```

Freeze

Enables the graphical freeze mode, preventing changes to open diagrams.

```
int SetDesktop.Freeze()
```

RETURNS

- 0** on success
- 1** on error

GetActivePage

Returns the page object of the currently shown page.

```
DataObject SetDesktop.GetActivePage()
```

RETURNS

Page object of the active page, or 0 if no page is active.

GetCanvasSize

Returns the pixel dimensions of the currently active (top-most) graphic page.

```
[int valid  
int width,  
int height] SetDesktop.GetCanvasSize()
```

ARGUMENTS

- width (out)*
Pixel width of the canvas. -1 if no graphics page is open.
- height (out)*
Pixel height of the canvas. -1 if no graphics page is open.

RETURNS

- 0** on success: canvas size could be determined
- 1** on error: no graphics page is open

GetPage

Searches, activates and returns a graphics page in the currently open graphics board. If “create” is true, then a new page will be created and added to the graphics board if no page with name was found.

```
DataObject SetDesktop.GetPage(str name,
                              [int create,]
                              [str class]
                              )
```

ARGUMENTS

name Name of the page.

create (optional)

Possible values:

- 0** do not create new plot page
- 1** create plot page if it does not exist already

class (optional)

Classname of the plot page object to create: either 'GrpPage' or 'SetVipage'. If not specified, a 'GrpPage' will be created if the new plot framework is enabled in the project settings, otherwise a 'SetVipage' will be created.

RETURNS

Plot page (GrpPage or SetVipage), or network graphic page (SetDeskpage)

IsFrozen

Returns whether the graphical freeze mode is currently enabled.

```
int SetDesktop.IsFrozen()
```

RETURNS

- 0** freeze mode is disabled, or the graphics board is not open
- 1** graphics board is open and freeze mode is enabled

IsOpened

Returns whether the graphics board is currently shown.

```
int SetDesktop.IsOpened()
```

RETURNS

- 1** if graphics board is shown
- 0** otherwise

RemovePage

Removes a graphic page. The page to be removed can be identified either by its name or by its page object.

```
int SetDesktop.RemovePage(str pageName)
int SetDesktop.RemovePage(DataObject pageObject)
```

ARGUMENTS

pageName
Name of graphics page.

pageObject
A graphics page object.

RETURNS

0 on success
1 on error

SetAdaptX

Sets the Adapt Scale option of the x-scale.

```
None SetDesktop.SetAdaptX(int mode,  
                           [float trigger]  
                           )
```

ARGUMENTS

mode Possible values:

0 off
1 on

trigger (optional)
Trigger value, unused if mode is off or empty

SetAutoScaleX

Sets automatic scaling mode of the x-scale. A warning is issued if an invalid mode is passed to the function.

```
None SetDesktop.SetAutoScaleX(int mode)
```

ARGUMENTS

mode Possible values:

0 never
1 after simulation
2 during simulation

SetResults

Sets default results object of graphics board.

```
None SetDesktop.SetResults(DataObject res)
```

ARGUMENTS

res Result object to set or None to reset. Valid result object is any of class ElmRes, IntComtrade and IntComtradeset.

SetScaleX

Sets x-axis scale. A function call without arguments sets the Auto Scale setting to On without changing the scale itself.

```
None SetDesktop.SetScaleX()
None SetDesktop.SetScaleX(float min,
                           float max,
                           [int log]
                           )
```

ARGUMENTS

min (optional)
Minimum of x-scale.

max (optional)
Maximum of x-scale.

log (optional)
Possible values:

0	linear
1	logarithmic

SetXVar

Sets x-axis variable. If The default x-axis variable (time) is set if no argument is passed.

```
None SetDesktop.SetXVar()
None SetDesktop.SetXVar(DataObject obj,]
                        str varname
                        )
```

ARGUMENTS

obj (optional)
x-axis object

varname (optional)
variable of obj

Show

Shows the virtual instrument panel with the same name as 'pageObject' or the page with name 'pageName' in the graphics board. The object 'pageObject' is typically a object of class 'SetVipage' (virtual instrument panel) but, as only its name is used, it may be any other type of object. Calling the function without an argument opens the graphics board.

```
int SetDesktop.Show()
int SetDesktop.Show(str pageName)
int SetDesktop.Show(DataObject pageObject)
```


ARGUMENTS

pageName (optional)
Name of graphics page.

pageObject (optional)
A graphics page object.

RETURNS

0 on success
1 on error

Unfreeze

Disables the graphical freeze mode, allowing changes to open diagrams.

```
int SetDesktop.Unfreeze()
```

RETURNS

0 on success
1 on error

WriteWMF

Writes the currently open graphic to a windows metafile file (*.wmf).

Please use the Save File command (ComWr) for writing to other formats like *.pdf, *.png, *.svg, *.emf or *.bmp.

```
int SetDesktop.WriteWMF(str filename)
```

ARGUMENTS

filename Filename without extension.

RETURNS

0 On error.
1 On success.

ZoomAll

Adjusts the zoom level of the currently active (top-most) graphic page such that the entire diagram is shown.

```
int SetDesktop.ZoomAll()
```

RETURNS

0 on success
1 on error

5.5.6 SetDistrstate

Overview

[CalcCluster](#)

CalcCluster

Calculates a load flow with a given load distribution state applied.

```
list(int, ...) SetDistrstate.CalcCluster(int clusterIndex,
                                         [int messageOn]
                                         )
```

ARGUMENTS

clusterIndex

The cluster index. Zero based value, the first cluster has index 0.

messageOn (optional)

Possible values:

- | | |
|----------|--|
| 0 | Do not emit a message in the output window. |
| 1 | Emit a message in the output window in case that the function does not execute properly. |

RETURNS

0 if ok. -1 if load flow of cluster did not converge.

5.5.7 SetFilt

Overview

[Get](#)

Get

Returns a container with the filtered objects.

```
list SetFilt.Get()
```

RETURNS

The set of filtered objects.

5.5.8 SetLevelvis

Overview

[AdaptWidth](#)

[Align](#)

[ChangeFont](#)

[ChangeFrameAndWidth](#)

[ChangeLayer](#)

[ChangeRefPoints](#)
[ChangeWidthVisibilityAndColour](#)
[Mark](#)
[Reset](#)

AdaptWidth

This function resizes the in the object specified group of text boxes regarding their text contents.

```
None SetLevelvis.AdaptWidth()
```

Align

This function aligns the text within a text box.

```
None SetLevelvis.Align(int iPos)
```

ARGUMENTS

<i>iPos</i>	Alignment position
0	left
1	middle
2	right

ChangeFont

This function sets the font number for the specified group of text boxes.

```
None SetLevelvis.ChangeFont(int iFont)
```

ARGUMENTS

<i>iFont</i>	Font number (default fonts range from 0 to 13)
--------------	--

ChangeFrameAndWidth

This method is not available anymore. Please use [SetLevelvis.ChangeWidthVisibilityAndColour\(\)](#) instead.

```
list(None, ...) SetLevelvis.ChangeFrameAndWidth([int iFrame,]
                                                  [int iWidth,]
                                                  [int iVisibility,]
                                                  [int iColour]
                                                  )
```

ChangeLayer

This function sets the specified group of text boxes to a given layer.

```
None SetLevelvis.ChangeLayer(str sLayer)
```

ARGUMENTS

sLayer Layer name (e.g. 'Object Names', 'Results', 'Invisible Objects',...)

ChangeRefPoints

This function sets the reference points between a text box (second parameter) and its parent object (first parameter), e.g. if the result box of a busbar shall be shown on top of a drawn bar instead of below the bar the values change from (6,4) to (4,6). The first number specifies the reference number of the text box. The integer values describe the position of the reference points within a rectangle (0=centre, 1=middle right, 2=top right,...):

4 3 2
5 0 1
6 7 8

```
None SetLevelvis.ChangeRefPoints(int iParRef,
                                   int iTBRef
                                   )
```

ARGUMENTS

iParRef Defines the reference point on the parent object (e.g. busbar)

iTBRef Defines the reference point on the text box

ChangeWidthVisibilityAndColour

This function sets the visibility of the frame, the width (in number of letters), the visibility and the colour of text boxes.

```
None SetLevelvis.ChangeWidthVisibilityAndColour([int iWidth,]
                                                  [int iVisibility,]
                                                  [int iColour]
                                                  )
```

ARGUMENTS

iWidth Sets the width in number of letters

0..n width

iVisibility Sets the visibility

0 not visible

1 visible

iColour Sets the colour

0..255 colour

Mark

Marks the specified group of text boxes in the currently shown diagram.

```
None SetLevelvis.Mark()
```

Reset

This function resets the individually modified text box settings.

```
None SetLevelvis.Reset(int iMode)
```

ARGUMENTS

iMode

- | | |
|----------|---|
| 0 | Reset to default (changed reference points are not reset) |
| 1 | Only font |
| 2 | Shift to original layer (result boxes to layer 'Results', object names to layer 'Object Names') |

5.5.9 SetParalman

Overview

[GetNumSlave](#)
[SetNumSlave](#)
[SetTransfType](#)

GetNumSlave

To get the number of slaves which is currently configured.

```
int SetParalman.GetNumSlave()
```

RETURNS

the number of slaves which is currently configured.

SetNumSlave

To configure the number of slaves to be used for parallel computing.

```
int SetParalman.SetNumSlave(int numSlaves)
```

ARGUMENTS

numSlaves

Number of slaves to be used for parallel computing

- | | |
|---------------|-----------------------------------|
| -1 | All cores available will be used. |
| > 0 | The number of slaves to be used. |

RETURNS

Always return 0.

SetTransfType

To change the data transfer type: via file or via socket communication.

```
int SetParalman.SetTransfType(int viaFile)
```

ARGUMENTS

viaFile

- | | |
|---|--|
| 0 | The data will be transferred via socket communication. |
| 1 | The data will be transferred via file. |

RETURNS

- | | |
|---|--|
| 0 | the data will be transferred via socket communication. |
| 1 | the data will be transferred via file. |

5.5.10 SetPath

Overview

[AllBreakers](#)
[AllClosedBreakers](#)
[AllOpenBreakers](#)
[AllProtectionDevices](#)
[Create](#)
[GetAll](#)
[GetBranches](#)
[GetBuses](#)
[GetPathFolder](#)

AllBreakers

Returns all breakers in the path definition.

```
list SetPath.AllBreakers()
```

RETURNS

The set of breakers.

AllClosedBreakers

Returns all closed breakers in the path definition.

```
list SetPath.AllClosedBreakers()
```

RETURNS

The set of closed breakers.

AllOpenBreakers

Returns all open breakers in the path definition.

```
list SetPath.AllOpenBreakers()
```

RETURNS

The set of open breakers.

AllProtectionDevices

Returns all protection devices in the path definition for a given direction.

```
list SetPath.AllProtectionDevices(int reverse)
```

ARGUMENTS

reverse

- | | |
|----------|--------------------------------------|
| 0 | Return devices in forward direction. |
| 1 | Return devices in reverse direction. |

RETURNS

The set of protection devices.

Create

Creates or extends the path with the elements provided.

```
DataObject SetPath.Create(list elements)
```

ARGUMENTS

elements Elements the path shall be created or extended with.

RETURNS

- DataObject** Modification was successful.
None Modification failed. (e.g. elements form an incomplete path)

GetAll

Returns all objects in the path definition.

```
list SetPath.GetAll()
```

RETURNS

The set of objects.

GetBranches

Returns all branches in the path definition.

```
list SetPath.GetBranches([int reverse])
```

ARGUMENTS

reverse (optional)

- | | |
|----------|---|
| 0 | Sort the branches in forward direction. |
| 1 | Sort the branches in reverse direction. |

RETURNS

The set of branches.

GetBuses

Returns all busbars and terminals in the path definition.

```
list SetPath.GetBuses()
```

RETURNS

The set of busbars and terminals.

GetPathFolder

Returns the default folder for storing path objects.

```
DataObject SetPath.GetPathFolder([int create])
```

ARGUMENTS

create (optional)

- | | |
|----------|---|
| 0 | Return only if the folder exists. |
| 1 | Create the folder if it does not exist. |

RETURNS

DataObject Default folder for storing path.

None Default folder does not exist or could not be created.

5.5.11 SetSelect

Overview

[AddRef](#)
[All](#)
[AllAsm](#)
[AllBars](#)
[AllBreakers](#)
[AllClosedBreakers](#)
[AllElm](#)
[AllLines](#)
[AllLoads](#)
[AllOpenBreakers](#)
[AllSym](#)
[AllTypLne](#)
[Clear](#)
[GetAll](#)

AddRef

Adds a reference to the objects to the existing selection.

```
None SetSelect.AddRef(DataObject O)
None SetSelect.AddRef(list S)
```

ARGUMENTS

- | | |
|----------|------------|
| O | An object. |
|----------|------------|

S A set of objects.

All

Returns all objects in the selection.

```
list SetSelect.All()
```

RETURNS

The set of objects

AllAsm

Returns all asynchronous machines in the selection.

```
list SetSelect.AllAsm()
```

RETURNS

The set of objects

AllBars

Returns all busbars and terminals in the selection.

```
list SetSelect.AllBars()
```

RETURNS

The set of objects

AllBreakers

Returns all breakers in the selection.

```
list SetSelect.AllBreakers()
```

RETURNS

The set of objects

AllClosedBreakers

Returns all closed breakers in the selection.

```
list SetSelect.AllClosedBreakers()
```

RETURNS

The set of objects

AllElm

Returns all elements (Elm*) in the selection.

```
list SetSelect.AllElm()
```

RETURNS

The set of containing objects

AllLines

Returns all lines and line routes in the selection.

```
list SetSelect.AllLines()
```

RETURNS

The set of objects

AllLoads

Returns all loads in the selection.

```
list SetSelect.AllLoads()
```

RETURNS

The set of objects

AllOpenBreakers

Returns all open breakers in the selection.

```
list SetSelect.AllOpenBreakers()
```

RETURNS

The set of objects

AllSym

Returns all synchronous machines in the selection.

```
list SetSelect.AllSym()
```

RETURNS

The set of objects

AllTypLne

Returns all line types in the selection.

```
list SetSelect.AllTypLne()
```

RETURNS

The set of objects

Clear

Clears (deletes) the selection.

```
None SetSelect.Clear()
```

GetAll

Returns all objects in the selection which are of the class 'ClassName'.

```
list SetSelect.GetAll(str ClassName)
```

ARGUMENTS

ClassName

The object class name.

RETURNS

The set of objects

5.5.12 SetTboxconfig

Overview

[Check](#)
[GetAvailableButtons](#)
[GetDisplayedButtons](#)
[Purge](#)
[SetDisplayedButtons](#)

Check

Checks buttons to be displayed for invalid or duplicate ids and prints error messages.

```
int SetTboxconfig.Check()
```

RETURNS

0	No errors found.
1	Errors found.

GetAvailableButtons

Gets buttons available for selected tool bar.

```
str SetTboxconfig.GetAvailableButtons()
```

RETURNS

String ids of all buttons available for selected tool bar; ids are separated by '\n'.

GetDisplayedButtons

Gets buttons configured to be displayed in selected tool bar.

```
str SetTboxconfig.GetDisplayedButtons()
```

RETURNS

String ids of all buttons configured to be displayed in selected tool bar; ids are separated by '\n'.

Purge

Purges buttons to be displayed from invalid or duplicate ids.

```
int SetTboxconfig.Purge()
```

RETURNS

- 0** No problems found.
- 1** Configuration was adapted.

SetDisplayedButtons

Sets buttons to be displayed in selected tool bar. Purges given buttons from invalid or duplicate buttons (duplicate separators or breaks are kept).

```
int SetTboxconfig.SetDisplayedButtons(str buttonIds)
```

ARGUMENTS

buttonIds String ids of all buttons to be set as displayed buttons; ids have to be separated by '\n'

RETURNS

- 0** Given buttons were stored without modification.
- 1** Given buttons were purged from invalid or duplicate ids.

5.5.13 SetTime

Overview

[Date](#)
[SetTime](#)
[SetTimeUTC](#)
[Time](#)

Date

Sets date component to current system date.

```
None SetTime.Date()
```

SEE ALSO

[SetTime.Time\(\)](#), [SetTime.SetTimeUTC\(\)](#)

SetTime

Sets the time in the current year. There is no restriction to the values for H, M and S, except for the fact that negative values are interpreted as zero. Values higher than 24 or 60 will be processed normally by adding the hours, minutes and seconds into an absolute time, from which a new hour-of-year, hour-of-day, minutes and seconds are calculated.

```
None SetTime.SetTime(float H,  
                     [float M,]  
                     [float S]  
                     )
```

ARGUMENTS

H The hours
M (optional)
 The minutes
S (optional)
 The seconds

SetTimeUTC

Sets date and time to given time. The time must be given in UTC format as seconds since 01.01.1970 00:00 GMT.

```
None SetTime.SetTimeUTC(int time)
```

ARGUMENTS

time UTC time in seconds since 01.01.1970 00:00 GMT

SEE ALSO

[SetTime.Date\(\)](#), [SetTime.Time\(\)](#)

Time

Sets time component to current system time.

```
None SetTime.Time()
```

SEE ALSO

[SetTime.Date\(\)](#), [SetTime.SetTimeUTC\(\)](#)

5.5.14 SetUser

Overview

[GetNumProcesses](#)

GetNumProcesses

This function returns the actual number of processes for parallel computation.

```
int SetUser.GetNumProcesses()
```

RETURNS

The actual number of processes for parallel computation.

5.5.15 SetVipage

Overview

[Close](#)
[DoAutoScaleX](#)
[DoAutoScaleY](#)
[GetOrInsertPlot](#)
[InsertPlot](#)
[MigratePage](#)
[SetAdaptX](#)
[SetAutoScaleX](#)
[SetResults](#)
[SetScaleX](#)
[SetStyle](#)
[SetTile](#)
[SetXVar](#)
[Show](#)

Close

Closes the graphic page, if currently shown, and deletes it from the database.

```
int SetVipage.Close()
```

RETURNS

- | | |
|---|--------------------------------|
| 0 | On success, no error occurred. |
| 1 | Otherwise |

DoAutoScaleX

Scales the x-axes of all plots on the virtual instrument panel automatically.

```
None SetVipage.DoAutoScaleX()
```

DoAutoScaleY

Scales the y-axes of all plots on the virtual instrument panel automatically.

```
None SetVipage.DoAutoScaleY()
```

GetOrInsertPlot

Get or create a virtual instrument of the virtual instrument panel.

```
DataObject SetVipage.GetOrInsertPlot (str name,
                                     [str class,]
                                     [int create]
                                     )
```

DEPRECATED NAMES

GetVI

ARGUMENTS

name Name of virtual instrument

class='VisPlot' (optional)
 classname of virtual instrument.

create (optional)
 Possible values:

- | | |
|----------|--|
| 0 | do not create new virtual instrument |
| 1 | create virtual instrument if it does not exist already |

RETURNS

Virtual instrument

InsertPlot

Creates a copy of the virtual instrument passed and displays the copy on this panel.

```
DataObject SetVipage.InsertPlot (DataObject vi)
```

DEPRECATED NAMES

CreateVI

ARGUMENTS

vi The virtual instrument which will be copied. Only virtual instruments are allowed (classname 'Vis*').

RETURNS

Returns the created virtual instrument.

MigratePage

Converts this SetVipage to the new plot framework introduced in PowerFactory 2021, creating a GrpPage:

- The original SetVipage will remain unchanged
- The created GrpPage will initially be hidden. Use GrpPage.Show() to make it visible.

Please note that currently (PowerFactory 2021) only plots of type VisPlot, VisPlot2, and VisXy-plot can be migrated. All other plot types will be missing in the migrated GrpPage.

```
None SetVipage.MigratePage ()
```

RETURNS

The migrated plot page (GrpPage)

SetAdaptX

Sets the Adapt Scale option of the x-scale.

```
None SetVipage.SetAdaptX(int mode,
                        [float trigger]
                        )
```

ARGUMENTS

mode Possible values:

0	off
1	on

trigger (optional)

Trigger value, unused if mode is off or empty

SetAutoScaleX

Sets automatic scaling mode of the x-scale. A warning is issued if an invalid mode is passed to the function.

```
None SetVipage.SetAutoScaleX(int mode)
```

ARGUMENTS

mode Possible values:

0	never
1	after simulation
2	during simulation

SetResults

Sets default results object of virtual instrument panel.

```
None SetVipage.SetResults(DataObject res)
```

ARGUMENTS

res Result object to set or None to reset. Valid result object is any of class ElmRes, IntComtrade and IntComtradeset.

SetScaleX

Sets x-axis scale. A function call without arguments sets the Auto Scale setting to On without changing the scale itself.


```

None SetVipage.SetScaleX()
None SetVipage.SetScaleX(float min,
                        float max,
                        [int log]
                        )

```

ARGUMENTS

min (optional)

Minimum of x-scale.

max (optional)

Maximum of x-scale.

log (optional)

Possible values:

0	linear
1	logarithmic

SetStyle

Sets style of virtual instrument panel. A warning message is issued in the case that a style with the given name does not exist.

```

None SetVipage.SetStyle(str name)

```

ARGUMENTS

name Style Name

SetTile

Rearranges the virtual instrument on the panel.

```

None SetVipage.SetTile([int tile])

```

ARGUMENTS

tile=1 (optional) **tile=0** arrange virtual instruments automatically (like tiles)
 tile=1 arrange them horizontally

SetXVar

Sets x-axis variable. If The default x-axis variable (time) is set if no argument is passed.

```

None SetVipage.SetXVar()
None SetVipage.SetXVar(DataObject obj,
                        str varname
                        )

```

ARGUMENTS

obj (optional)
x-axis object

varname (optional)
variable of obj

Show

Displays the plot page in the graphics board.

```
int SetVipage.Show()
```

RETURNS

0 On success, no error occurred.

1 Otherwise

5.6 Others

5.6.1 BlkDef

Overview

[Compile](#)
[Encrypt](#)
[GetCheckSum](#)
[Pack](#)
[PackAsMacro](#)
[ResetThirdPartyModule](#)
[SetThirdPartyModule](#)

Compile

Compiles the model to a DLL. Can be called on an already compiled model. A study case of a project has to be active.

```
None BlkDef.Compile([string modelPath])
```

ARGUMENTS

modelPath (optional)
Full path to a location where the model should be stored. Leave empty to use default.

Encrypt

Encrypts this block definition. It has to be packed as macro before.

```
int BlkDef.Encrypt([int removeObjectHistory])
```

ARGUMENTS

removeObjectHistory (optional)

Handling of unencrypted object history in database, e.g. used by project versions or by undo:

- 0** Do not remove.
- 1** Do remove (default).
- 2** Show dialog and ask.

RETURNS

- 0** On success.
- 1** On error.

SEE ALSO

[BlkDef.PackAsMacro\(\)](#)

GetChecksum

```
str BlkDef.GetChecksum()
```

DEPRECATED NAMES

CalculateChecksum

RETURNS

The checksum of the block definition (0000-0000-0000-0000 for frames).

Pack

Copies all used macros (i.e. referenced BlkDef) to this block.

```
int BlkDef.Pack()
```

RETURNS

- 0** On success.
- 1** On error.

PackAsMacro

Collects all equations, stores them to this model and deletes block diagram and all macro references.

```
int BlkDef.PackAsMacro()
```

RETURNS

- 0** On success.
- 1** On error.

SEE ALSO

[BlkDef.Encrypt\(\)](#)

ResetThirdPartyModule

Resets the third party licence. Only possible for non-encrypted, non-compiled blocks. Requires masterkey licence for third party module currently set.

```
int BlkDef.ResetThirdPartyModule()
```

RETURNS

- 0** On success.
- 1** On error.

SetThirdPartyModule

Sets the third party licence to a specific value. Only possible for non-encrypted, non-compiled blocks with no third party licence set so far. Requires masterkey licence for third party module to be set.

```
int BlkDef.SetThirdPartyModule(str companyCode,
                               str moduleCode
                               )
```

ARGUMENTS

companyCode

D isplay name or numeric value of company code.

moduleCode

D isplay name or numeric value of third party module.

RETURNS

- 0** On success.
- 1** On error.

5.6.2 BlkSig

Overview

[GetFromSigName](#)

[GetToSigName](#)

GetFromSigName

```
str BlkSig.GetFromSigName()
```

RETURNS

The name of the output from which the signal is connected. In cases of no connection, an empty string.

GetToSigName

```
str BlkSig.GetToSigName()
```

RETURNS

The name of the input to which the signal is connected. In cases of no connection, an empty string.

5.6.3 ChaVecfile

Overview

[Update](#)

Update

Reloads the file from disk. Same behaviour like button update.

```
int ChaVecfile.Update([int msgOn = 0])
```

ARGUMENTS

msgOn (optional)

Reporting of errors:

- | | |
|----------|--|
| 0 | No error message is shown in case that the file can not be loaded (default). |
| 1 | Emit an error message in case that the file can not be loaded. |

RETURNS

The number of samples (rows) read from the file.

5.6.4 CimModel

Overview

[DeleteParameterAtIndex](#)
[GetAttributeEnumerationType](#)
[GetModelsReferencingThis](#)
[GetParameterCount](#)
[GetParameterNamespace](#)
[GetParameterValue](#)
[HasParameter](#)
[RemoveParameter](#)
[SetAssociationValue](#)
[SetAssociationValue](#)
[SetAttributeEnumeration](#)
[SetAttributeEnumeration](#)
[SetAttributeValue](#)
[SetAttributeValue](#)

DeleteParameterAtIndex

Removes the parameter (attribute, or association) value at the given index.

```
None CimModel.DeleteParameterAtIndex(str parameter, int index)
```

ARGUMENTS

parameter Full-name specifier of the attribute, or association (e.g. "Model.profile")

index Index of the parameter

GetAttributeEnumerationType

Returns the enumeration type of the attribute.

```
str CimModel.GetAttributeEnumerationType(str attribute)
```

ARGUMENTS

attribute Full-name specifier of the attribute (e.g. "GeneratingUnit.genControlSource")

GetModelsReferencingThis

Returns all CIM models (CimModel) that reference the calling model.

```
list CimModel.GetModelsReferencingThis()
```

RETURNS

CIM models that reference the calling model. The order of the set is undefined.

GetParameterCount

Returns the number of parameters (attribute, or association) of given type.

```
int CimModel.GetParameterCount(str parameter)
```

ARGUMENTS

parameter Full-name specifier of the attribute, or association (e.g. "Model.profile")

GetParameterNamespace

Returns the namespace of the parameter (attribute, or association).

```
str CimModel.GetParameterNamespace(str parameter)
```

ARGUMENTS

parameter Full-name specifier of the attribute, or association (e.g. "Model.profile")

GetParameterValue

Returns the value of the parameter (attribute, or association) at the given index if available. If the parameter (attribute, or association) is not available, or the index is out of bounds the function returns an empty string.

```
str CimModel.GetParameterValue(str parameter, [int index])
```

ARGUMENTS

- parameter* Full-name specifier of the attribute, or association (e.g. "Model.modelingAuthoritySet")
- index* Index of the parameter:
- 0** Default index

HasParameter

Checks whether the CimModel has the parameter (attribute, or association) specified.

```
int CimModel.HasParameter(str parameter)
```

ARGUMENTS

- parameter* Full-name specifier of the attribute, or association (e.g. "Model.modelingAuthoritySet")

RETURNS

- 1** if parameter is specified
- 0** if parameter is not specified

RemoveParameter

Removes all occurrences of the parameter (attribute, or association).

```
None CimModel.RemoveParameter(str parameter)
```

ARGUMENTS

- parameter* Full-name specifier of the attribute, or association (e.g. "Model.modelingAuthoritySet")

SetAssociationValue

Adds the association if not available yet, and sets its value at the given index. If the association is already added, the function sets a new value at the given index only.

```
None CimModel.SetAssociationValue(str association,
                                   str value,
                                   [int index])
```

ARGUMENTS

<i>association</i>	Full-name specifier of the association (e.g. "Model.DependentOn")
<i>value</i>	Value of the association
<i>index</i>	Index of the association:
0	Default index

SetAssociationValue

Adds the association if not available yet, and sets its namespace and value. If the association is already added, the function sets its namespace and value only.

```
None CimModel.SetAssociationValue(str association,
                                   str value,
                                   str nspace)
```

ARGUMENTS

<i>attribute</i>	Full-name specifier of the association (e.g. "Model.DependentOn")
<i>value</i>	Value of the association
<i>nspace</i>	Namespace of the association (e.g. "md")

SetAttributeEnumeration

Adds the attribute if not available yet, and sets its enumeration type and value. If the attribute is already added, the function sets its enumeration type and value only.

```
None CimModel.SetAttributeEnumeration(str attribute,
                                       str enumerationType,
                                       str value)
```

ARGUMENTS

<i>attribute</i>	Full-name specifier of the attribute (e.g. "GeneratingUnit.genControlSource")
<i>enumerationType</i>	Enumeration type of the attribute (e.g. "GeneratorControlSource")
<i>value</i>	Value of the enumeration (e.g. "offAGC")

SetAttributeEnumeration

Adds the attribute if not available yet, and sets its namespace, enumeration type and value. If the attribute is already added, the function sets its namespace, enumeration type and value only.

```
None CimModel.SetAttributeEnumeration(str attribute,
                                       str enumerationType,
                                       str value,
                                       str nspace)
```


ARGUMENTS

- attribute* Full-name specifier of the attribute (e.g. "GeneratingUnit.genControlSource")
- enumerationType* Enumeration type of the attribute (e.g. "GeneratorControlSource")
- value* Value of the attribute (e.g. "offAGC")
- nspace* Namespace of the attribute (e.g. "cim")

SetAttributeValue

Adds the attribute if not available yet, and sets its value at the given index. If the attribute is already added, the function sets a new value at the given index only.

```
None CimModel.SetAttributeValue(str attribute,
                                str value,
                                [int index])
```

ARGUMENTS

- attribute* Full-name specifier of the attribute (e.g. "Model.modelingAuthoritySet")
- value* Value of the attribute
- index* Index of the attribute:
 - 0** Default index

SetAttributeValue

Adds the attribute if not available yet, and sets its namespace and value. If the attribute is already added, the function sets its namespace and value only.

```
None CimModel.SetAttributeValue(str attribute,
                                str value,
                                str nspace)
```

ARGUMENTS

- attribute* Full-name specifier of the attribute (e.g. "Model.modelingAuthoritySet")
- value* Value of the attribute
- nspace* Namespace of the attribute (e.g. "md")

5.6.5 CimObject

Overview

[DeleteParameterAtIndex](#)
[GetAttributeEnumerationType](#)
[GetObjectsReferencingThis](#)
[GetObjectsWithSameId](#)
[GetParameterCount](#)
[GetParameterNamespace](#)
[GetParameterValue](#)
[GetPfObjects](#)
[HasParameter](#)
[RemoveParameter](#)
[SetAssociationValue](#)
[SetAssociationValue](#)
[SetAttributeEnumeration](#)
[SetAttributeEnumeration](#)
[SetAttributeValue](#)
[SetAttributeValue](#)

DeleteParameterAtIndex

Removes the parameter (attribute, or association) value at the given index.

```
None CimObject.DeleteParameterAtIndex(str parameter, int index)
```

ARGUMENTS

parameter Full-name specifier of the attribute, or association (e.g. "Model.profile")
index Index of the parameter

GetAttributeEnumerationType

Returns the enumeration type of the attribute.

```
str CimObject.GetAttributeEnumerationType(str attribute)
```

ARGUMENTS

attribute Full-name specifier of the attribute (e.g. "GeneratingUnit.genControlSource")

GetObjectsReferencingThis

Returns all CIM objects (CimObject) that reference the calling object. The set of objects returned is also determined by the DependentOn and Supersedes references set in parent CIM model objects. In order for a CIM object to reference another CIM object, the parent CIM model of the former object has to hold a reference to the parent CIM model of the later object.

```
list CimObject.GetObjectsReferencingThis()
```

g

RETURNS

CIM objects that reference the calling object. The order of the set is undefined.

GetObjectsWithSameId

Returns all CIM objects (CimObject) that have the same Resource ID as this object.

```
list CimObject.GetObjectsWithSameId()
```

RETURNS

CIM objects that have the same Resource ID as this object.

GetParameterCount

Returns the number of parameters (attribute, or association) of given type.

```
int CimObject.GetParameterCount(str parameter)
```

ARGUMENTS

parameter

Full-name specifier of the attribute, or association (e.g. "Model.profile")

GetParameterNamespace

Returns the namespace of the parameter (attribute, or association).

```
str CimObject.GetParameterNamespace(str parameter)
```

ARGUMENTS

parameter

Full-name specifier of the attribute, or association (e.g. "IdentifiedObject.name")

GetParameterValue

Returns the value of the parameter (attribute, or association) at the given index if available. If the parameter (attribute, or association) is not available, or the index is out of bounds the function returns an empty string.

```
str CimObject.GetParameterValue(str parameter, [int index])
```

ARGUMENTS

parameter

Full-name specifier of the attribute, or association (e.g. "IdentifiedObject.name")

index

Index of the parameter:

0 Default index

GetPfObjects

Returns all PF objects that have the same Resource ID as this CIM object.

```
list CimObject.GetPfObjects()
```

RETURNS

PF objects that have the same Resource ID as this CIM object.

HasParameter

Checks whether the CimObject has the parameter (attribute, or association) specified.

```
int CimObject.HasParameter(str parameter)
```

ARGUMENTS

parameter

Full-name specifier of the attribute, or association (e.g. "IdentifiedObject.name")

RETURNS

- 1** if parameter is specified
- 0** if parameter is not specified

RemoveParameter

Removes all occurrences of the parameter (attribute, or association).

```
None CimObject.RemoveParameter(str parameter)
```

ARGUMENTS

parameter

Full-name specifier of the attribute, or association (e.g. "IdentifiedObject.name")

SetAssociationValue

Adds the association if not available yet, and sets its value at the given index. If the association is already added, the function sets a new value at the given index only.

```
None CimObject.SetAssociationValue(str association,
                                   str value,
                                   [int index])
```

ARGUMENTS

association

Full-name specifier of the association (e.g. "Equipment.EquipmentContainer")

value

Value of the association

index

Index of the association:

- 0** Default index

SetAssociationValue

Adds the association if not available yet, and sets its namespace and value. If the association is already added, the function sets its namespace and value only.

```
None CimObject.SetAssociationValue(str association,  
                                   str value,  
                                   str nspace)
```

ARGUMENTS

- attribute* Full-name specifier of the association (e.g. "Equipment.EquipmentContainer")
- value* Value of the association
- nspace* Namespace of the association (e.g. "cim")

SetAttributeEnumeration

Adds the attribute if not available yet, and sets its enumeration type and value. If the attribute is already added, the function sets its enumeration type and value only.

```
None CimObject.SetAttributeEnumeration(str attribute,  
                                       str enumerationType,  
                                       str value)
```

ARGUMENTS

- attribute* Full-name specifier of the attribute (e.g. "GeneratingUnit.genControlSource")
- enumerationType*
Enumeration type of the attribute (e.g. "GeneratorControlSource")
- value* Value of the enumeration (e.g. "offAGC")

SetAttributeEnumeration

Adds the attribute if not available yet, and sets its namespace, enumeration type and value. If the attribute is already added, the function sets its namespace, enumeration type and value only.

```
None CimObject.SetAttributeEnumeration(str attribute,  
                                       str enumerationType,  
                                       str value,  
                                       str nspace)
```

ARGUMENTS

- attribute* Full-name specifier of the attribute (e.g. "GeneratingUnit.genControlSource")
- enumerationType*
Enumeration type of the attribute (e.g. "GeneratorControlSource")
- value* Value of the attribute (e.g. "offAGC")
- nspace* Namespace of the attribute (e.g. "cim")

SetAttributeValue

Adds the attribute if not available yet, and sets its value at the given index. If the attribute is already added, the function sets a new value at the given index only.

```
None CimObject.SetAttributeValue(str attribute,
                                str value,
                                [int index])
```

ARGUMENTS

attribute Full-name specifier of the attribute (e.g. "IdentifiedObject.name")

value Value of the attribute

index Index of the attribute:

0 Default index

SetAttributeValue

Adds the attribute if not available yet, and sets its namespace and value. If the attribute is already added, the function sets its namespace and value only.

```
None CimObject.SetAttributeValue(str attribute,
                                str value,
                                str nspace)
```

ARGUMENTS

attribute Full-name specifier of the attribute (e.g. "IdentifiedObject.name")

value Value of the attribute

nspace Namespace of the attribute (e.g. "cim")

5.6.6 GrpPage

Overview

DoAutoScale
 DoAutoScaleX
 DoAutoScaleY
 GetOrInsertCurvePlot
 GetOrInsertDiscreteBarPlot
 GetOrInsertXYPlot
 GetPlot
 RemovePage
 SetAutoScaleModeX
 SetAutoScaleModeY
 SetLayoutMode
 SetResults
 SetScaleTypeX
 SetScaleTypeY
 SetScaleX
 SetScaleY
 Show

DoAutoScale

Adapts axis ranges of all plots on the page such that they show the entire data range.

```
None GrpPage.DoAutoScale([int axisDimension])
```

ARGUMENTS

axisDimension (optional)

Limits auto-scaling to one dimension. Possible values:

- | | |
|----------|-------------------|
| 0 | scale only x-axes |
| 1 | scale only y-axes |

DoAutoScaleX

Adapts x-axis ranges of all plots on the page such that they show the entire data range.

```
None GrpPage.DoAutoScaleX()
```

DoAutoScaleY

Adapts y-axis ranges of all plots on the page such that they show the entire data range.

```
None GrpPage.DoAutoScaleY()
```

GetOrInsertCurvePlot

Finds a curve plot by name, or creates it if not found.

```
DataObject GrpPage.GetOrInsertCurvePlot(str name,
                                         [int create]
                                         )
```

ARGUMENTS

name Name of plot

create=1 (optional)

Possible values:

- | | |
|----------|--|
| 0 | do not create new plot |
| 1 | create plot if it does not exist already |

RETURNS

PltLinebarplot object

GetOrInsertDiscreteBarPlot

Finds a discrete bar plot by name, or creates it if not found. A discrete bar plot is a PltLinebarplot whose x-axis mode is set to 'Discrete', i.e., it shows net elements on the x-axis.

```
DataObject GrpPage.GetOrInsertDiscreteBarPlot(str name,
                                                [int create]
                                                )
```

ARGUMENTS

name Name of plot

create=1 (optional)

Possible values:

- | | |
|----------|--|
| 0 | do not create new plot |
| 1 | create plot if it does not exist already |

RETURNS

PltLinebarplot object

GetOrInsertXYPlot

Finds a XY plot by name, or creates it if not found. A XY plot is a PltLinebarplot whose x-axis mode is set to 'XY'.

```
DataObject GrpPage.GetOrInsertXYPlot(str name,  
                                     [int create]  
                                     )
```

ARGUMENTS

name Name of plot

create=1 (optional)

Possible values:

- | | |
|----------|--|
| 0 | do not create new plot |
| 1 | create plot if it does not exist already |

RETURNS

PltLinebarplot object

GetPlot

Returns the plot on this page with the given name.

```
DataObject GrpPage.GetPlot(str name)
```

ARGUMENTS

name Name of the plot to look for

RETURNS

Plot object if found, or NULL otherwise

RemovePage

Closes the graphic page, if currently shown, and deletes it from the database.

```
int GrpPage.RemovePage()
```


SetAutoScaleModeX

Defines whether the x-axes on the page should automatically adapt their range on data changes.

```
None GrpPage.SetAutoScaleModeX(int mode)
```

ARGUMENTS

<i>mode</i>	Possible values:
0	off (do not react to data changes)
1	adapt scale after calculation has finished
2	adapt scale during live plotting

SetAutoScaleModeY

Defines whether the y-axes on the page should automatically adapt their range on data changes.

```
None GrpPage.SetAutoScaleModeY(int mode)
```

ARGUMENTS

<i>mode</i>	Possible values:
0	off (do not react to data changes)
1	adapt scale after calculation has finished
2	adapt scale during live plotting

SetLayoutMode

Defines the automatic arrangement of plots on the page.

```
None GrpPage.SetLayoutMode(int mode)
```

ARGUMENTS

<i>mode</i>	Possible values:
0	off (do not arrange plots automatically)
1	arrange plots vertically
2	arrange plots on grid

SetResults

Sets the default results object of page.

```
None GrpPage.SetResults(DataObject res)
```

ARGUMENTS

<i>res</i>	Result object to set or None to reset. Valid result object is any of class ElmRes, IntComtrade and IntComtradeset.
------------	--

SetScaleTypeX

Sets the scale type (linear, logarithmic) of all x-axes on the page.

```
None GrpPage.SetScaleTypeX(int scaleType)
```

ARGUMENTS

scaleType

Possible values:

0	linear
1	logarithmic

SetScaleTypeY

Set the scale type (linear, logarithmic, dB) of all y-axes on the page.

```
None GrpPage.SetScaleTypeY(int scaleType)
```

ARGUMENTS

scaleType

Possible values:

0	linear
1	logarithmic
2	dB

SetScaleX

Sets the scale of all x-axes on the page.

```
None GrpPage.SetScaleX(float min,  
                        float max  
                        )
```

ARGUMENTS

<i>min</i>	Minimum of x-scale.
<i>max</i>	Maximum of x-scale.

SetScaleY

Sets the scale of all y-axes on the page.

```
None GrpPage.SetScaleY(float min,  
                        float max  
                        )
```

ARGUMENTS

<i>min</i>	Minimum of y-scale.
<i>max</i>	Maximum of y-scale.

Show

Displays the diagram page in the graphics board.

```
int GrpPage.Show()
```

RETURNS

- 0** On success, no error occurred.
- 1** Otherwise

5.6.7 IntAddonvars**Overview**

[AddDouble](#)
[AddDoubleMatrix](#)
[AddDoubleVector](#)
[AddInteger](#)
[AddIntegerVector](#)
[AddObject](#)
[AddObjectVector](#)
[AddString](#)
[RemoveParameter](#)

AddDouble

Adds a new double parameter to the Date Extension configuration object.

```
int IntAddonvars.AddDouble(str parameterName,
                           str desc,
                           str unitText,
                           float initialValue
                           )
```

ARGUMENTS

- parameterName* The name of the new parameter
- desc* The description of the new parameter
- unitText* The unit of the new parameter
- initialValue* The initial value of the new parameter

RETURNS

Zero if attribute was successfully added. One based row number if there already is a parameter with the same name.

AddDoubleMatrix

Adds a new double vector parameter to the Date Extension configuration object.

```
int IntAddonvars.AddDoubleMatrix(str parameterName,
                                  str desc,
```

```
int initialRows,
int initialColumns,
str unitText,
float initialValue)
```

ARGUMENTS

parameterName
The name of the new parameter

desc The description of the new parameter

initialRows
The initial number of rows for the matrix

initialColumns
The initial number of columns for the matrix

unitText The unit of the new parameter

initialValue
The initial value for the elements

RETURNS

Zero if attribute was successfully added. One based row number if there already is a parameter with the same name.

AddDoubleVector

Adds a new double vector parameter to the Date Extension configuration object.

```
int IntAddonvars.AddDoubleVector(str parameterName,
                                str desc,
                                int initialSize,
                                str unitText,
                                float initialValue
                                )
```

ARGUMENTS

parameterName
The name of the new parameter

desc The description of the new parameter

initialSize The initial size of the vector

unitText The unit of the new parameter

initialValue
The initial value for the elements

RETURNS

Zero if attribute was successfully added. One based row number if there already is a parameter with the same name.

AddInteger

Adds a new integer parameter to the Date Extension configuration object.

```
int IntAddonvars.AddInteger(str parameterName,  
                           str desc,  
                           str unitText,  
                           int initialValue  
                           )
```

ARGUMENTS

parameterName The name of the new parameter

desc The description of the new parameter

unitText The unit of the new parameter

initialValue The initial value of the new parameter

RETURNS

Zero if attribute was successfully added. One based row number if there already is a parameter with the same name.

AddIntegerVector

Adds a new integer vector parameter to the Date Extension configuration object.

```
int IntAddonvars.AddIntegerVector(str parameterName,  
                                  str desc,  
                                  int initialSize,  
                                  str unitText,  
                                  int initialValue)
```

ARGUMENTS

parameterName The name of the new parameter

desc The description of the new parameter

initialSize The initial size of the vector

unitText The unit of the new parameter

initialValue The initial value for the elements

RETURNS

Zero if attribute was successfully added. One based row number if there already is a parameter with the same name.

AddObject

Adds a new string parameter to the Date Extension configuration object.

```
int IntAddonvars.AddObject(str parameterName,  
                           str desc,  
                           str classFilter  
                           )
```

ARGUMENTS

- parameterName* The name of the new parameter
- desc* The description of the new parameter
- classFilter* The filter for the objects which are allowed for selection

RETURNS

Zero if attribute was successfully added. One based row number if there already is a parameter with the same name.

AddObjectVector

Adds a new object vector parameter to the Date Extension configuration object.

```
int IntAddonvars.AddObjectVector(str parameterName,  
                                 str desc,  
                                 int initialSize,  
                                 str classFilter  
                                 )
```

ARGUMENTS

- parameterName* The name of the new parameter
- desc* The description of the new parameter
- initialSize* The initial size of the vector
- classFilter* The filter for the objects which are allowed for selection

RETURNS

Zero if attribute was successfully added. One based row number if there already is a parameter with the same name.

AddString

Adds a new string parameter to the Date Extension configuration object.

```
int IntAddonvars.AddString(str parameterName,  
                           str desc,  
                           str unitText,  
                           str initialValue  
                           )
```

ARGUMENTS

<i>parameterName</i>	The name of the new parameter
<i>desc</i>	The description of the new parameter
<i>unitText</i>	The unit of the new parameter
<i>initialValue</i>	The initial value of the new parameter

RETURNS

Zero if attribute was successfully added. One based row number if there already is a parameter with the same name.

RemoveParameter

Removes the given parameter from the Data Extension configuration.

```
None IntAddonvars.RemoveParameter(str parameterName)
```

ARGUMENTS

<i>parameterName</i>	The name of the parameter to be removed
----------------------	---

5.6.8 IntCase**Overview**

[Activate](#)
[ApplyNetworkState](#)
[ApplyStudyTime](#)
[Consolidate](#)
[Deactivate](#)
[SetStudyTime](#)

Activate

Activates the study case. Deactivates other study cases first.

```
int IntCase.Activate()
```

RETURNS

0	on success
1	on error

ApplyNetworkState

For a study case in a combined project, copy the network state from another case.

Copies the active grids, scenarios and network variations configuration to the current case. The data will be added to any already existing configuration.

```
int IntCase.ApplyNetworkState(DataObject other)
```

ARGUMENTS

other The source Study Case to copy data from

RETURNS

- 0** On success
- 1** Source object is not an IntCase object
- 2** Case where function is called on is not the active case
- 3** Source case is not from active project
- 4** Source Study Case is not from a source project in a combined project
- 5** Other error. Details are given in an error message

ApplyStudyTime

For a study case in a combined project, apply the study time from another study case.

```
int IntCase.ApplyStudyTime(DataObject other)
```

ARGUMENTS

other The source study case to copy study time from

RETURNS

- 0** On success
- 1** Source object is not an IntCase object
- 2** Study case where function is called on is not the active case
- 3** Source case is not from active project
- 4** Source case is not from a project part of a combined project

Consolidate

Changes that are recorded in a project's active Variations are permanently applied to the Network Data folder (like right mouse button Consolidate Network Variation)

Note: Modified scenarios are not saved!

Works only:

- For active study cases
- If a network variation is active

```
int IntCase.Consolidate()
```

RETURNS

- 0** On success
- 1** If an error has occurred

SEE ALSO

[IntScheme.Consolidate\(\)](#)

Deactivate

De-activates the study case.

```
int IntCase.Deactivate()
```

RETURNS

0	on success
1	on error

SetStudyTime

Sets the current Study Case time to seconds since 01.01.1970 00:00:00. Use IntCase:iStudyTime for getting current Study Case time.

```
None IntCase.SetStudyTime(int dateTime)
```

ARGUMENTS

dateTime Seconds since 01.01.1970 00:00:00.

5.6.9 IntComtrade

Overview

[ConvertToASCIIFormat](#)
[ConvertToBinaryFormat](#)
[FindColumn](#)
[FindMaxInColumn](#)
[FindMinInColumn](#)
[GetAnalogueDescriptions](#)
[GetDescription](#)
[GetDigitalDescriptions](#)
[GetNumberOfAnalogueSignalDescriptions](#)
[GetNumberOfColumns](#)
[GetNumberOfDigitalSignalDescriptions](#)
[GetNumberOfRows](#)
[GetObjectValue](#)
[GetSignalHeader](#)
[GetUnit](#)
[GetValue](#)
[GetVariable](#)
[Load](#)
[Release](#)
[SortAccordingToColumn](#)

ConvertToASCIIFormat

Creates new comtrade configuration and data files in ASCII format in the file system directory of the original files. The new configuration file is linked automatically to a new IntComtrade object created in the same **PowerFactory** folder like this object. An existing IntComtrade object is already in ASCII format when its parameter 'Binary' is set to 0.

```
int IntComtrade.ConvertToASCIIFormat()
```

RETURNS

- 0** File successfully converted.
- 1** Error occurred, e.g. file is already in ASCII format.

ConvertToBinaryFormat

Creates new comtrade configuration and data files in binary format in the file system directory of the original files. The new configuration file is linked automatically to a new IntComtrade object created in the same **PowerFactory** folder like this object. An existing IntComtrade object is already in binary format when its parameter 'Binary' is set to 1.

```
int IntComtrade.ConvertToBinaryFormat()
```

RETURNS

- 0** File successfully converted.
- 1** Error occurred, e.g. file is already in binary format.

FindColumn

Returns the first column matching the variable name.

```
int IntComtrade.FindColumn(str variable,
                           [int startCol]
                           )
```

ARGUMENTS

- variable* The variable name to look for.
- startCol (optional)*
The index of the column at which to start the search.

RETURNS

- ≥ 0 The column index found.
- < 0 The column with name variable was not found.

FindMaxInColumn

Find the maximum value of the variable in the given column.

```
[int row,
float value] IntComtrade.FindMaxInColumn(int column)
```

ARGUMENTS

- column* The column index.
- value (optional, out)*
The maximum value found. The value is 0. in case that the maximum value was not found.

RETURNS

- < 0 The maximum value of column was not found.
- ≥ 0 The row with the maximum value of the column.

FindMinInColumn

Find the minimum value of the variable in the given column.

```
[int row,
float value] IntComtrade.FindMinInColumn(int column)
```

ARGUMENTS

- column* The column index.
- value (optional, out)*
The minimum value found. The value is 0. in case that the minimum value was not found.

RETURNS

- < 0 The minimum value of column was not found.
- ≥ 0 The row with the minimum value of the column.

GetAnalogueDescriptions

Get the descriptions of the analogue channel Please note that the comtrade info file contains proprietary data from PFM. Info files not written by DigSILENT PFM are not supported.

```
str IntComtrade.GetAnalogueDescriptions(int index)
```

ARGUMENTS

- index*) Digital channel index

RETURNS

- Descriptions for analogue channel in the comtrade info file.

GetDescription

Get the description of a column.

```
str IntComtrade.GetDescription([int column],
                               [int short]
                               )
```

ARGUMENTS

- column (optional)*
The column index. The description name of the default variable is returned if the parameter is not passed to the function.
- short (optional)*
 - 0** long desc. (default)
 - 1** short description

RETURNS

Returns the description which is empty in case that the column index is not part of the data.

GetDigitalDescriptions

Get the descriptions of the digital channel Please note that the comtrade info file contains proprietary data from PFM. Info files not written by DlgSILENT PFM are not supported.

```
str IntComtrade.GetDigitalDescriptions(int index)
```

ARGUMENTS

index Digital channel index

RETURNS

Descriptions for digital channel in the comtrade info file.

GetNumberOfAnalogueSignalDescriptions

Gets the number of descriptions for analogue channels in the comtrade info file. Please note that the comtrade info file contains proprietary data from PFM. Info files not written by DlgSILENT PFM are not supported.

```
int IntComtrade.GetNumberOfAnalogueSignalDescriptions()
```

RETURNS

Number of descriptions for analogue channels in the comtrade info file.

GetNumberOfColumns

Returns the number of variables (columns) in result file excluding the default variable (e.g. time for time domain simulation).

```
int IntComtrade.GetNumberOfColumns()
```

RETURNS

Number of variables (columns) in result file.

GetNumberOfDigitalSignalDescriptions

Get the number of descriptions for digital channels in the comtrade info file. Please note that the comtrade info file contains proprietary data from PFM. Info files not written by DlgSILENT PFM are not supported.

```
int IntComtrade.GetNumberOfDigitalSignalDescriptions()
```

RETURNS

Number of descriptions for digital channels in the comtrade info file.

GetNumberOfRows

Returns the number of values per column (rows) stored in result object.

```
int IntComtrade.GetNumberOfRows()
```

RETURNS

Returns the number of values per column stored in result object.

GetObjectValue

Returns a value from a result object for row iX of curve col.

```
[int error,
DataObject o ] IntComtrade.GetObjectValue(int iX,
                                           [int col])
```

ARGUMENTS

o (out) The object retrieved from the data.

iX The row.

col (optional)

The curve number, which equals the variable or column number, first column value (time,index, etc.) is returned when omitted.

RETURNS

- 0** when ok
- 1** when iX out of bound
- 2** when col out of bound
- 3** when invalid value is returned from a sparse file. Sparse files are written e.g. by the contingency analysis, the value is invalid in case that it was not written, because it was below the recording limit. Result files created using DPL/Python are always full and will not return invalid values.

GetSignalHeader

Get the headline of the channel section in the comtrade info file. Please note that the comtrade info file contains proprietary data from PFM. Info files not written by DlgSILENT PFM are not supported.

```
str IntComtrade.GetSignalHeader()
```

RETURNS

Headline of signal descriptions

GetUnit

Get the unit of a column.

```
str IntComtrade.GetUnit([int column])
```

ARGUMENTS

column (optional)

The column index. The unit of the default variable is returned if the parameter is nor passed to the function.

RETURNS

Returns the unit which is empty in case that the column index is not part of the data.

GetValue

Returns a value from a result object for row iX of curve col.

```
[int error,
float d    ] IntComtrade.GetValue(int iX,
                                   [int col])
```

ARGUMENTS

d (out) The value retrieved from the data.

iX The row.

col (optional)

The curve number, which equals the variable or column number, first column value (time,index, etc.) is returned when omitted.

RETURNS

- 0** when ok
- 1** when iX out of bound
- 2** when col out of bound
- 3** when invalid value is returned from a sparse file. Sparse files are written e.g. by the contingency analysis, the value is invalid in case that it was not written, because it was below the recording limit. Result files created using DPL/Python are always full and will not return invalid values.

GetVariable

Get variable name of column

```
str IntComtrade.GetVariable([int column])
```

ARGUMENTS

column (optional)

The column index. The variable name of the default variable is returned if the parameter is nor passed to the function.

RETURNS

Returns the variable name which is empty in case that the column index is not part of the data.

Load

Loads the data of a result object (**IntComtrade**) in memory for reading.

```
None IntComtrade.Load()
```

Release

Releases the data loaded to memory. This function should be used whenever several result objects are processed in a loop. Data is always released from memory automatically after execution of the current script.

```
None IntComtrade.Release()
```

SortAccordingToColumn

Sorts all rows in the data loaded according to the given column. The IntComtrade itself remains unchanged.

```
int IntComtrade.SortAccordingToColumn(int column)
```

ARGUMENTS

col The column number.

RETURNS

- 0** The function executed correctly, the data was sorted correctly according to the given column.
- 1** The column with index column does not exist.

5.6.10 IntComtradeset

Overview

[FindColumn](#)
[FindMaxInColumn](#)
[FindMinInColumn](#)
[GetAnalogueDescriptions](#)
[GetDescription](#)
[GetDigitalDescriptions](#)
[GetNumberOfAnalogueSignalDescriptions](#)
[GetNumberOfColumns](#)
[GetNumberOfDigitalSignalDescriptions](#)
[GetNumberOfRows](#)
[GetObjectValue](#)
[GetSignalHeader](#)
[GetUnit](#)
[GetValue](#)
[GetVariable](#)
[Load](#)
[Release](#)
[SortAccordingToColumn](#)

FindColumn

Returns the first column matching the variable name.

```
int IntComtradeset.FindColumn(str variable,
                             [int startCol]
                             )
```

ARGUMENTS

variable The variable name to look for.

startCol (optional)
The index of the column at which to start the search.

RETURNS

≥ 0 The column index found.

< 0 The column with name variable was not found.

FindMaxInColumn

Find the maximum value of the variable in the given column.

```
[int row,
float value] IntComtradeset.FindMaxInColumn(int column)
```

ARGUMENTS

column The column index.

value (optional, out)
The maximum value found. The value is 0. in case that the maximum value was not found.

RETURNS

< 0 The maximum value of column was not found.

≥ 0 The row with the maximum value of the column.

FindMinInColumn

Find the minimum value of the variable in the given column.

```
[int row,
float value] IntComtradeset.FindMinInColumn(int column)
```

ARGUMENTS

column The column index.

value (optional, out)
The minimum value found. The value is 0. in case that the minimum value was not found.

RETURNS

< 0 The minimum value of column was not found.

≥ 0 The row with the minimum value of the column.

GetAnalogueDescriptions

Get the descriptions of the analogue channel Please note that the comtrade info file contains proprietary data from PFM. Info files not written by DlgSILENT PFM are not supported.

```
str IntComtradeset.GetAnalogueDescriptions(int index)
```

ARGUMENTS

index) Digital channel index

RETURNS

Descriptions for analogue channel in the comtrade info file.

GetDescription

Get the description of a column.

```
str IntComtradeset.GetDescription([int column],
                                   [int short]
                                   )
```

ARGUMENTS

column (optional)

The column index. The description name of the default variable is returned if the parameter is not passed to the function.

short (optional)

0	long desc. (default)
1	short description

RETURNS

Returns the description which is empty in case that the column index is not part of the data.

GetDigitalDescriptions

Get the descriptions of the digital channel Please note that the comtrade info file contains proprietary data from PFM. Info files not written by DlgSILENT PFM are not supported.

```
str IntComtradeset.GetDigitalDescriptions(int index)
```

ARGUMENTS

index Digital channel index

RETURNS

Descriptions for digital channel in the comtrade info file.

GetNumberOfAnalogueSignalDescriptions

Gets the number of descriptions for analogue channels in the comtrade info file. Please note that the comtrade info file contains proprietary data from PFM. Info files not written by DlgSILENT PFM are not supported.

```
int IntComtradeset.GetNumberOfAnalogueSignalDescriptions()
```

RETURNS

Number of descriptions for analogue channels in the comtrade info file.

GetNumberOfColumns

Returns the number of variables (columns) in result file excluding the default variable (e.g. time for time domain simulation).

```
int IntComtradeset.GetNumberOfColumns()
```

RETURNS

Number of variables (columns) in result file.

GetNumberOfDigitalSignalDescriptions

Get the number of descriptions for digital channels in the comtrade info file. Please note that the comtrade info file contains proprietary data from PFM. Info files not written by DlgSILENT PFM are not supported.

```
int IntComtradeset.GetNumberOfDigitalSignalDescriptions()
```

RETURNS

Number of descriptions for digital channels in the comtrade info file.

GetNumberOfRows

Returns the number of values per column (rows) stored in result object.

```
int IntComtradeset.GetNumberOfRows()
```

RETURNS

Returns the number of values per column stored in result object.

GetObjectValue

Returns a value from a result object for row iX of curve col.

```
[int error,  
DataObject o ] IntComtradeset.GetObjectValue(int iX,  
                                              [int col])
```

ARGUMENTS

o (out) The object retrieved from the data.

iX The row.

col (optional)
The curve number, which equals the variable or column number, first column value (time,index, etc.) is returned when omitted.

RETURNS

- 0** when ok
- 1** when iX out of bound
- 2** when col out of bound
- 3** when invalid value is returned from a sparse file. Sparse files are written e.g. by the contingency analysis, the value is invalid in case that it was not written, because it was below the recording limit. Result files created using DPL/Python are always full and will not return invalid values.

GetSignalHeader

Get the headline of the channel section in the comtrade info file. Please note that the comtrade info file contains proprietary data from PFM. Info files not written by DlgSILENT PFM are not supported.

```
str IntComtradeset.GetSignalHeader()
```

RETURNS

Headline of signal descriptions

GetUnit

Get the unit of a column.

```
str IntComtradeset.GetUnit([int column])
```

ARGUMENTS

column (optional)

The column index. The unit of the default variable is returned if the parameter is not passed to the function.

RETURNS

Returns the unit which is empty in case that the column index is not part of the data.

GetValue

Returns a value from a result object for row iX of curve col.

```
[int error,
float d ] IntComtradeset.GetValue(int iX,
                                   [int col])
```

ARGUMENTS

d (out) The value retrieved from the data.

iX The row.

col (optional)

The curve number, which equals the variable or column number, first column value (time, index, etc.) is returned when omitted.

RETURNS

- 0** when ok
- 1** when iX out of bound
- 2** when col out of bound
- 3** when invalid value is returned from a sparse file. Sparse files are written e.g. by the contingency analysis, the value is invalid in case that it was not written, because it was below the recording limit. Result files created using DPL/Python are always full and will not return invalid values.

GetVariable

Get variable name of column

```
str IntComtradeset.GetVariable([int column])
```

ARGUMENTS

column (optional)

The column index. The variable name of the default variable is returned if the parameter is not passed to the function.

RETURNS

Returns the variable name which is empty in case that the column index is not part of the data.

Load

Loads the data of a result object (**IntComtradeset**) in memory for reading.

```
None IntComtradeset.Load()
```

Release

Releases the data loaded to memory. This function should be used whenever several result objects are processed in a loop. Data is always released from memory automatically after execution of the current script.

```
None IntComtradeset.Release()
```

SortAccordingToColumn

Sorts all rows in the data loaded according to the given column. The **IntComtradeset** itself remains unchanged.

```
int IntComtradeset.SortAccordingToColumn(int column)
```

ARGUMENTS

col The column number.

RETURNS

- 0** The function executed correctly, the data was sorted correctly according to the given column.
- 1** The column with index column does not exist.

5.6.11 IntDataset

Overview

[AddRef](#)
[All](#)
[Clear](#)
[GetAll](#)

AddRef

Adds new reference(s) for passed object(s) as children to the dataset object. Nothing happens if there exists already a reference for the passed object.

```
None IntDataset.AddRef(DataObject object)
None IntDataset.AddRef(list objects)
```

ARGUMENTS

obj/objects

Object(s) for which references should be created and added to the dataset object

All

Returns all children of the dataset object.

```
list IntDataset.All()
```

RETURNS

All objects contained in dataset object.

Clear

Deletes all children of the dataset object.

```
None IntDataset.Clear()
```

GetAll

Returns all children of the dataset filtered according to given class name.

```
list IntDataset.GetAll(str className)
```

ARGUMENTS

className

class name filter, e.g. ElmTerm

RETURNS

All objects of given class stored in dataset object.

5.6.12 IntDocument

Overview

[Export](#)
[Import](#)
[Reset](#)
[View](#)

Export

Exports the embedded data as new file on disk. The embedded data remains unmodified. If desired it can be removed by calling the `Reset()` function afterwards

```
int IntDocument.Export(str filename)
```

ARGUMENTS

filename Name of export file on disk

RETURNS

0 On success.
1 On error.

SEE ALSO

[IntDocument.Import\(\)](#)

Import

Imports the content of selected file into the PowerFactory object. The data is afterwards embedded in the PowerFactory database.

```
int IntDocument.Import()
```

RETURNS

0 On success.
1 On error.

SEE ALSO

[IntDocument.Export\(\)](#)

Reset

Resets embedded data and reference to an external file.

```
int IntDocument.Reset()
```

View

Views the file in external application. If the file is embedded, it's extracted into a temporary file that is opened afterwards. Please note, the action is only executed if access to given file (type) is enabled in the 'External Access' configuration of PowerFactory (IntExtaccess).

```
int IntDocument.View()
```

RETURNS

- 0** Success, file was opened
- 1** Error, file not opened (because of invalid address or security reasons)

SEE ALSO

[IntUrl.View\(\)](#)

5.6.13 IntDplmap

Overview

[Clear](#)
[Contains](#)
[First](#)
[GetValue](#)
[Insert](#)
[Next](#)
[Remove](#)
[Size](#)
[Update](#)

Clear

Removes all key/value pairs from the container and resets type information.

```
None IntDplmap.Clear()
```

Contains

Checks if a key/value pair with given key is contained in the container.

```
int IntDplmap.Contains(int | float | str | DataObject | list key)
```

ARGUMENTS

- key* Key of the associated pair in the container

RETURNS

- 1 if an entry of same key is contained, otherwise 0.

First

Outputs the first key/value pair stored in the container.

Note:

- The sequence of the returned pairs is determined by internal criteria and cannot be changed.
- It is not allowed to modify a container while iterating over it. If doing so, the next call of the `Next` command will return a value of 1.
Exception: `Update()` does not invalidate current position.

```
[int end,
int | float | str | DataObject | list key
int | float | str | DataObject | list value] IntDplmap.First()
```

ARGUMENTS

key (out) Key of the associated pair in the container

value (out)
Value of the associated pair in the container

RETURNS

1 if no next entry is available in the container (e.g. end is reached), otherwise 0.

GetValue

Returns the associated value for given key.

```
[int|float|str|DataObject|list value,
int error] IntDplmap.GetValue(int|float|str|DataObject|list key)
```

ARGUMENTS

key Key of the associated pair in the container to find.

error (optional, out)

1 Key was not found in container.
0 Key was found in the container.

RETURNS

The value which is associated to the given key or an undefined value if key is not associated with any value.

Insert

Inserts given key and value as an associated pair into the container.

On the first insertion, the container is (automatically) typed by given data types of key and value. From now on, only keys and values of that types are accepted. (This type information is removed when `IntDplmap.Clear()` is called.)

If given key already exists in the container, its associated value will be overwritten. (Each key can only be contained once in a map (no multi-map support).)

Note:

- Type of key and value can be different, of course.
- Sets are always inserted by value, not by reference!

```
None IntDplmap.Insert (int | float | str | DataObject | list key,
                      int | float | str | DataObject | list value)
```


ARGUMENTS

- key* Key of the associated pair in the container.
- value* Value of the associated pair in the container.

Next

Outputs the next key/value pair relative to the last key/value pair in the container.

Note:

- The sequence of the returned pairs is determined by internal criteria and cannot be changed.
- It is not allowed to modify a container while iterating over it. If doing so, the next call of the Next command will return a value of 1.

Exception: Update() does not invalidate current position.

```
[int end,
int | float | str | DataObject | list key
int | float | str | DataObject | list value] IntDplmap.Next()
```

ARGUMENTS

- key (out)* Key of the associated pair in the container.
- value (out)* Value of the associated pair in the container.

RETURNS

- 1 if no next entry is available in the container (e.g. end is reached), otherwise 0.

Remove

Removes the key/value pair for given key from the container. No error will occur, if the key is not contained in the container.

```
None IntDplmap.Remove (int | float | str | DataObject | list key)
```

ARGUMENTS

- key* Key of the associated pair in the container

Size

Returns the number of key/value pairs stored in the container.

```
int IntDplmap.Size()
```

RETURNS

- Number of key-value pairs stored in the container.

Update

Is a special insert function that can be used for updating key/value pairs in the map. It can only be used if the key is already contained in the map.

```
None IntDplmap.Update(int | float | str | DataObject | list key,
                      int | float | str | DataObject | list value)
```

ARGUMENTS

key Key of the associated pair in the container

value Value of the associated pair in the container

5.6.14 IntDplvec

Overview

[Clear](#)
[Get](#)
[IndexOf](#)
[Insert](#)
[Remove](#)
[Size](#)
[Sort](#)

Clear

Removes all elements from the container and resets the typ information.

```
None IntDplvec.Clear()
```

Get

Returns the element stored at given position in the container.

```
int|float|str|DataObject|list IntDplvec.Get(int position)
```

ARGUMENTS

position Position in the container. It is zero-based and must always be lesser than the container's size.

RETURNS

Element stored at given position in the container.

IndexOf

Returns the position where the given element is stored in the container.

```
int IntDplvec.IndexOf(int|float|str|DataObject|list element,
                     [int startPosition]
                     )
```

ARGUMENTS

element Element for which the position will be searched.

startPosition

Start position from which the next occurrence greater or equal to this position is searched.

RETURNS

Position of the the given element in the container. The returned position is zero-based. If no occurrence was found, -1 is returned.

Insert

Inserts an element at given position into the container. If no position is given then the element is appended to the back. Inserting an element to an empty container fixes the type of elements which can be hold by itself. Clearing the container resets this type information.

```
None IntDplvec.Insert(int|float|str|DataObject|list element)
None IntDplvec.Insert(int position,
                      int|float|str|DataObject|list element
                      )
```

ARGUMENTS

element Element to be inserted.

position Position (zero-based) to insert the element at. Any old entry at that position will be overwritten. Note: The size of the vector is automatically increased if given position is greater than current size.

Remove

Removes the element stored at given position from the container.

```
None IntDplvec.Remove(int position)
```

ARGUMENTS

position Given position (zero-based) at which the element is to be removed.

Size

Returns the number of elements stored in the container.

```
int IntDplvec.Size()
```

RETURNS

Number of elements stored in the container.

Sort

Sorts the elements of the vector depending on the type of elements stored inside the vector:

string lexically

double/int
 according to value

object according to full name (path + name) or given attribute

```
None IntDplvec.Sort([int descending,]
                    [str attribute]
                    )
```

ARGUMENTS

descending (optional)

- 1** Descending sorting order
- 0** Ascending sorting order (default)

attribute (optional)

For objects only: Attribute according to which the sorting is done (default is full name)

5.6.15 IntEvt

Overview

[CreateCBEvents](#)
[RemoveSwitchEvents](#)

CreateCBEvents

Create boundary breaker events for all shc locations which occur simultaneously in this fault case.

```
None IntEvt.CreateCBEvents([int iRemoveExisting])
```

ARGUMENTS

iRemoveExisting (optional)

- 1** Query user if circuit breaker events exist.
- 0** Do not create circuit breaker events if circuit breaker events are already defined events exist (default)
- 1** Remove existing circuit breaker events.

RemoveSwitchEvents

Remove all switch events of this fault case.

```
None IntEvt.RemoveSwitchEvents([int onlyContingency])
```

ARGUMENTS

onlyContingency (optional)

Condition to remove.

- 0** Remove all switch events regardless of the calculation type.

- 1 Remove all switch events only when this fault case is used for contingency analysis.

5.6.16 IntExtaccess

Overview

[CheckUrl](#)

CheckUrl

Checks whether access to given url will be granted or not according to the security settings. See also [IntUrl.View\(\)](#) for accessing that url.

```
int IntExtaccess.CheckUrl(str url)
```

ARGUMENTS

url url to check

RETURNS

0	access granted
1	access denied

5.6.17 IntGate

Overview

[AddTrigger](#)

AddTrigger

Adds either a condition or a gate to the gate.

```
list IntGate.AddTrigger(DataObject newTrigger)
```

ARGUMENTS

newTrigger
The condition or gate that shall be added.

5.6.18 IntGrf

Overview

[MoveToLayer](#)

MoveToLayer

Moves an annotation element stored as (obsolete) *IntGrf* object to an annotation layer (*IntGrflayer*) or group (*IntGrfgroup*).

```
None IntGrf.MoveToLayer(DataObject layer)
```

ARGUMENTS

layer Target *IntGrflayer* or *IntGrfgroup* object.

5.6.19 IntGrfgroup

Overview

[ClearData](#)
[Export](#)
[Import](#)

ClearData

Removes all annotation elements from this group.

```
None IntGrfgroup.ClearData()
```

Export

Exports all objects of a group into svg-file.

```
None IntGrfgroup.Export(str path,  
                        [int OpenDialog])
```

ARGUMENTS

path Full export file path

OpenDialog (optional)

Prompt for export path in dialog

0 Export directly and do not show any dialog (default)

1 Show dialog with path before exporting

Import

Imports svg-file into group object.

```
None IntGrfgroup.Import(str path)
```

ARGUMENTS

path Path of file to be imported.

5.6.20 IntGrflayer

Overview

[ClearData](#)
[Export](#)
[ExportToVec](#)
[Import](#)
[ImportFromVec](#)

ClearData

Removes all annotation elements on this layer (keeps contained groups and annotation elements).

```
None IntGrflayer.ClearData()
```

Export

Exports all objects of a layer into svg-file, inclusive annotation objects of contained group objects.

```
None IntGrflayer.Export(str path,
                        [int OpenDialog])
```

ARGUMENTS

path Full export file path
OpenDialog (optional)
 Prompt for export path in dialog
 0 Export directly and do not show any dialog (default)
 1 Show dialog with path before exporting

ExportToVec

Fills string description of annotation elements of this layer into an *IntDplvec*. Clears *IntDplvec* before filling it.

ARGUMENTS

intDplVec *IntDplvec* object to be filled.

Import

Imports svg file into layer.

```
None IntGrflayer.Import(str path)
```

ARGUMENTS

path Path of file to be imported.

ImportFromVec

Fills this layer with the string description of annotation elements from an *IntDplvec*. Clears layer before filling it.

ARGUMENTS

intDplVec *IntDplvec* containing description of annotation elements.

5.6.21 IntGrfnet

Overview

[Close](#)
[SetLayerVisibility](#)
[SetSymbolComponentVisibility](#)
[Show](#)

Close

Closes the graphic page that displays this diagram.

```
int IntGrfnet.Close()
```

RETURNS

- 0** On success, no error occurred.
- 1** Otherwise

SetLayerVisibility

Sets a layer visible or invisible.

```
None IntGrfnet.SetLayerVisibility(str sLayer,  
                                  int iVis)
```

ARGUMENTS

- sLayer* Layer to be modified.
- iVis* Visibility
 - 0** Make layer invisible.
 - 1** Make layer visible.

SetSymbolComponentVisibility

Determines which parts of net element symbols are shown in the diagram.

```
None IntGrfnet.SetSymbolComponentVisibility(int componentID,  
                                             int visible)
```


ARGUMENTS

componentID

Component to be modified.

5	Connection points
7	Tap positions
8	Vector groups
9	Load flow arrows
11	Phases
13	Line sections and loads
14	Connection arrows
21	Connection numbers (block diagrams only)
22	Connection names (block diagrams only)
33	Remotely controlled substation markers
38	Tie open point markers
39	Open standby switch markers
40	Normally open switch markers

visible

Visibility

0	Make component invisible.
1	Make component visible.

Show

Opens a diagram.

```
int IntGrfnet.Show()
```

RETURNS

0	On success, no error occurred.
1	Otherwise

5.6.22 Intlcon**Overview**

[Export](#)
[Import](#)

Export

Exports current icon as a bitmap file.

```
int Intlcon.Export(str filename)
```

ARGUMENTS

filename Name of export image on disk. Extension needs to be '.bmp'

RETURNS

- 0** On success.
- 1** On error.

SEE ALSO

[IntIcon.Import\(\)](#)

Import

Imports icon from a bitmap file.

```
int IntIcon.Import(str filename)
```

ARGUMENTS

filename Name of bitmap file on disk. Extension and format needs to be '.bmp'

RETURNS

- 0** On success.
- 1** On error.

SEE ALSO

[IntIcon.Export\(\)](#)

5.6.23 IntLibrary

Overview

[Activate](#)
[Deactivate](#)

Activate

Activates this library. If another library is already activated it will be deactivated first.

```
int IntLibrary.Activate()
```

RETURNS

- 1 if successful**
- 0 otherwise**

Deactivate

Deactivates this library.

```
int IntLibrary.Deactivate()
```

RETURNS

- 1 if successful**
- 0 otherwise**

5.6.24 IntMat

Overview

[ColLbl](#)
[Get](#)
[GetColumnLabel](#)
[GetColumnLabelIndex](#)
[GetNumberOfColumns](#)
[GetNumberOfRows](#)
[GetRowLabel](#)
[GetRowLabelIndex](#)
[Init](#)
[Invert](#)
[Multiply](#)
[Resize](#)
[RowLbl](#)
[Save](#)
[Set](#)
[SetColumnLabel](#)
[SetRowLabel](#)
[SortToColumn](#)

ColLbl

Deprecated function to get or set the label of the given column. Please use [IntMat.GetColumnLabel\(\)](#) or [IntMat.SetColumnLabel\(\)](#) instead.

```
str IntMat.ColLbl(int column)
str IntMat.ColLbl(str label,
                    int column
                    )
```

Get

Returns the value at the position (row, column) of the matrix. A run-time error will occur when 'row' or 'column' is out of range.

```
float IntMat.Get(int row,
                 int column
                 )
```

ARGUMENTS

row Row in matrix: 1 ... GetNumberOfRows().
column column in matrix: 1 ... GetNumberOfColumn()

RETURNS

Value in matrix.

SEE ALSO

[IntMat.Set\(\)](#)

GetColumnLabel

Returns the label of a column.

```
str IntMat.GetColumnLabel(int column)
```

ARGUMENTS

column Column index (first column has index 1).

RETURNS

Column label of given column.

DEPRECATED NAMES

ColLbl

SEE ALSO

[IntMat.SetColumnLabel\(\)](#), [IntMat.GetColumnLabelIndex\(\)](#), [IntMat.GetRowLabel\(\)](#)

GetColumnLabelIndex

Gets the index of a label in all column labels.

```
int IntMat.GetColumnLabelIndex(str label)
```

ARGUMENTS

label Label to search.

RETURNS

≥ 1 The index in the column labels, if label was found.
0 Otherwise

SEE ALSO

[IntMat.GetColumnLabel\(\)](#)

GetNumberOfColumns

Returns the number of columns in the matrix.

```
int IntMat.GetNumberOfColumns()
```

RETURNS

The number of columns of the matrix.

DEPRECATED NAMES

NCol, SizeY

SEE ALSO

[IntMat.GetNumberOfRows\(\)](#)

GetNumberOfRows

Returns the number of rows in the matrix.

```
int IntMat.GetNumberOfRows()
```

RETURNS

The number of rows.

DEPRECATED NAMES

NRow, SizeX

SEE ALSO

[IntMat.GetNumberOfColumns\(\)](#)

GetRowLabel

Returns the label of a row.

```
str IntMat.GetRowLabel(int row)
```

ARGUMENTS

row Row index (first row has index 1).

RETURNS

Row label of given row.

DEPRECATED NAMES

RowLbl

SEE ALSO

[IntMat.SetRowLabel\(\)](#), [IntMat.GetRowLabelIndex\(\)](#), [IntMat.GetColumnLabel\(\)](#)

GetRowLabelIndex

Gets the index of a label in all row labels.

```
int IntMat.GetRowLabelIndex(str label)
```

ARGUMENTS

label Label to search.

RETURNS

≥ 1 The index in the row labels, if it was found.
0 Otherwise

SEE ALSO

[IntMat.GetRowLabel\(\)](#)

Init

Initializes the matrix with given size and values, regardless of the previous size and data.

This operation is performed in memory only. Use [IntMat.Save\(\)](#) to save the modified matrix to database.

```
int IntMat.Init(int numberOfRows,
               int numberOfColumns,
               [float initialValue = 0.0]
               )
```

ARGUMENTS

numberOfRows

The number of rows.

numberOfColumns

The number of columns.

initialValue (optional)

Initial values: All matrix entries are initialised with this value. Matrix is initialized with 0 if omitted.

RETURNS

Always 1 and can be ignored.

SEE ALSO

[IntMat.Resize\(\)](#)

Invert

Inverts the matrix.

This operation is performed in memory only. Use [IntMat.Save\(\)](#) to save the modified matrix to database.

```
int IntMat.Invert()
```

RETURNS

- 0** Success, the matrix is replaced by its inversion.
- 1** Error, inversion not possible. Original matrix was not changed.

Multiply

Multiplies 2 matrixes and stores the result in this matrix.

This operation is performed in memory only. Use [IntMat.Save\(\)](#) to save the modified matrix to database.

```
int IntMat.Multiply(DataObject M1,
                   DataObject M2
                   )
```

ARGUMENTS

object M1

Matrix 1 to be multiplied.

object M2

Matrix 2 to be multiplied.

RETURNS

Always 0 and can be ignored.

Resize

Resizes the matrix to a given size. Existing values will not be changed. Added values will be set to the optional value, otherwise to 0.

This operation is performed in memory only. Use [IntMat.Save\(\)](#) to save the modified matrix to database.

```
int IntMat.Resize(int numberOfRows,
                  int numberOfColumns,
                  [float initialValue = 0.0]
                  )
```

ARGUMENTS

numberOfRows

The number of rows.

numberOfColumns

The number of columns.

initialValue (optional)

Initial values: Additional matrix entries are initialised with this value. Additional values are initialized with 0. if omitted.

RETURNS

Always 1 and can be ignored.

SEE ALSO

[IntMat.Init\(\)](#)

RowLbl

Deprecated function to get or set the label of the given row. Please use [IntMat.GetRowLabel\(\)](#) or [IntMat.SetRowLabel\(\)](#) instead.

```
str IntMat.RowLbl(int row)
str IntMat.RowLbl(str label,
                  int row
                  )
```

Save

Saves the current state of this matrix to database.

```
None IntMat.Save()
```

Set

Sets a value at the position (row, column) of the matrix. The matrix is resized automatically if the given coordinates exceed the size.

This operation is performed in memory only. Use [IntMat.Save\(\)](#) to save the modified matrix to database.

```
int IntMat.Set(int row,
               int column,
               float value
              )
```

ARGUMENTS

- row* Row index, 1 based. The first row has index 1. Invalid index ($leq 0$) leads to scripting error.
- column* Column index, 1 based. The first column has index 1. Invalid index ($leq 0$) leads to scripting error.
- value* Value to assign.

RETURNS

Always 1 and can be ignored.

SEE ALSO

[IntMat.Get\(\)](#)

SetColumnLabel

Sets the label of a column.

This operation is performed in memory only. Use [IntMat.Save\(\)](#) to save the modified matrix to database.

```
None IntMat.SetColumnLabel(int column,
                           str label
                          )
```

ARGUMENTS

- column* Column index (first column has index 1).
- label* Label to set.

SEE ALSO

[IntMat.GetColumnLabel\(\)](#), [IntMat.GetColumnLabelIndex\(\)](#), [IntMat.SetRowLabel\(\)](#)

SetRowLabel

Sets the label of a row.

This operation is performed in memory only. Use [IntMat.Save\(\)](#) to save the modified matrix to database.

```
str IntMat.SetRowLabel(int row,
                      str label
                     )
```


ARGUMENTS

row Row index (first row has index 1).
label Label to set.

SEE ALSO

[IntMat.GetRowLabel\(\)](#), [IntMat.GetRowLabelIndex\(\)](#), [IntMat.SetColumnLabel\(\)](#)

SortToColumn

Sorts the matrix alphanumerically according to a column, which is specified by the input parameter. The row labels are sorted accordingly (if input parameter `storeInDB` is 1).

```
int IntMat.SortToColumn(int columnIndex,
                        [float epsilon = 0.0],
                        [int storeInDB = 1])
```

DEPRECATED NAMES

`SortToColumn`

ARGUMENTS

columnIndex
 The column index, 1 based. The first column has index 1.

epsilon (optional)
 Accuracy for comparing equal values. Values which differ less than epsilon are treated as being equal. Default value is 0.

storeInDb (optional)
 Possible Values:

0	Non-persistent change. Values are not stored in database.
1	Values are stored in database. (default)

RETURNS

0	On success.
1	Error. Original matrix was not changed.

5.6.25 IntMon**Overview**

[AddVar](#)
[AddVars](#)
[ClearVars](#)
[GetVar](#)
[NVars](#)
[PrintAllVal](#)
[PrintVal](#)
[RemoveVar](#)

AddVar

Appends the variable “name” to the list of selected variable names.

```
None IntMon.AddVar(str name)
```

ARGUMENTS

name The variable name to add.

AddVars

Appends the filtered variables to the list of selected variables.

```
None IntMon.AddVars(str varFilter)
```

ARGUMENTS

varFilter The filter for variables to add. For example: 'e:.*' to add all parameters of element to variable selection.

ClearVars

Clears the list of selected variable names.

```
None IntMon.ClearVars()
```

GetVar

Returns the variable name on the given row of the variable selection text on the second page of the IntMon dialogue, which should contain one variable name per line.

```
str IntMon.GetVar(int row)
```

ARGUMENTS

row Given row

RETURNS

The variable name in line row.

NVars

Returns the number of selected variables or, more exact, the number of lines in the variable selection text on the second page of the IntMon dialogue, which usually contains one variable name per line.

```
int IntMon.NVars()
```

RETURNS

The number of variables selected.

PrintAllVal

Writes all calculation results of the object assigned in `obj_id` to the output window. The output includes the variable name followed by the value, its unit and the description. It should be noted that the variable set itself is modified by this method.

```
None IntMon.PrintAllVal()
```

PrintVal

Prints the values of the selected variables to the output window.

```
None IntMon.PrintVal()
```

RemoveVar

Removes the variable “name” from the list of selected variable names.

```
int IntMon.RemoveVar(str name)
```

ARGUMENTS

name The variable name.

RETURNS

- 0** If variable with name was found and removed.
- 1** If the variable name was not found.

5.6.26 IntOutage

Overview

[Apply](#)
[ApplyAll](#)
[Check](#)
[CheckAll](#)
[IsInStudyTime](#)
[ResetAll](#)

Apply

```
None IntOutage.Apply([int reportSwitches])
```

Applies the outage object. The functionality corresponds to pressing the 'Apply' button in edit dialog with the difference that the scripting function can also be used without an active scenario.

ARGUMENTS

reportSwitches (optional)
 Flag to enable the reporting of changed switches to the output window.

- 0** No output (default)
- 1** Print switches to output window

ApplyAll

```
None IntOutage.ApplyAll([int reportSwitches])
```

Applies all currently relevant (=in study time and not out-of-service) outage objects of current project. The functionality corresponds to pressing the 'ApplyAll' button in edit dialog with the difference that the scripting function can also be used without an active scenario. It applies all relevant outages independent of the one it was called on.

ARGUMENTS

reportSwitches (optional)

Flag to enable the reporting of changed switches to the output window.

- 0** No output (default)
- 1** Print switches to output window

Check

```
int IntOutage.Check([int outputMessage])
```

This function checks if the outage is correctly reflected by the network elements.

ARGUMENTS

outputMessage (optional)

Flag to enable detailed output to the output window.

- 0** No output (default)
- 1** Detailed report of mismatch to output window

RETURNS

- 0** Ok, outage is correctly reflected
- 1** Not ok, status of network elements does not reflect outage

CheckAll

This function checks if all outages are correctly reflected by the network components for current study time. It checks all outages independent of the one it was called on.

```
[list notOutaged,  
list wronglyOutaged] IntOutage.CheckAll([int emitMsg,  
[DataObject gridfilter,])
```

ARGUMENTS

int emitMsg (optional)

whether to report inconsistencies to the output window

- 1** No output
- 0** (Default) print inconsistencies but without start / end message
- 1** Full output, including start / end message

gridfilter (optional)

Possibility to restrict checking for accidentally outaged elements to given object (e.g. grid) and its children (by default, all elements for all active grids are checked).

notOutaged (optional, out)(optional)

If given, all network components that should be outaged but are not are filled into this set.

wronglyOutaged (optional, out)(optional)

If given, all network components that should be outaged but are not are filled into this set.

IsInStudyTime

```
int IntOutage.IsInStudyTime()
```

Checks if outage is relevant for current study time, i.e. the study time lies within the outage's validity period.

RETURNS

- 0** Outage is not relevant for current study time (outside validity period)
- 1** Outage is relevant for current study time (inside validity period)

DEPRECATED NAMES

IsInStudytime

ResetAll

```
None IntOutage.ResetAll([int reportSwitches])
```

Resets all currently relevant (=in study time and not out-of-service) outage objects of current project. The functionality corresponds to pressing the 'Reset' button in all outage objects with difference that the scripting function can also be used without an active scenario. It resets all relevant outages independent of the one it was called on.

ARGUMENTS

reportSwitches (optional)

Flag to enable the reporting of changed switches to the output window.

- 0** No output (default)
- 1** Print switches to output window

5.6.27 IntPlannedout

Overview

[SetRecurrence](#)

SetRecurrence

Copies the settings of a Recurrence Pattern object to the planned outage object and enables the flag Recurrent.

```
int IntPlannedout.SetRecurrence(DataObject recurrencePattern)
```

ARGUMENTS

recurrencePattern

Recurrence pattern object, classname IntRecurrence

RETURNS

- 0** Ok, settings copied successful
- 1** Failed, passed object is NULL or not of class IntRecurrence

5.6.28 IntPlot

Overview

[SetAdaptY](#)
[SetAutoScaleY](#)
[SetScaleY](#)

SetAdaptY

Sets the Adapt Scale option of the x-scale.

```
None IntPlot.SetAdaptY(int mode,
                        [float offset]
                        )
```

ARGUMENTS

mode Possible values:

- 0** off
- 1** on

offset (optional)

Offset, unused if mode is off or empty

SetAutoScaleY

Sets automatic scaling mode of the y-scale. A warning is issued if an invalid mode is passed to the function.

```
None IntPlot.SetAutoScaleY(int mode)
```

ARGUMENTS

mode Possible values:

- 0** never
- 1** after simulation
- 2** during simulation

SetScaleY

Sets y-axis scale limits. A function call without arguments sets the Auto Scale setting to On without changing the scale itself.

```
None IntPlot.SetScaleY()
None IntPlot.SetScaleY(float min,
                        float max,
                        [int log]
                        )
```

ARGUMENTS

min (optional)

Minimum of y-scale.

max (optional)

Maximum of y-scale.

log (optional)

Possible values:

0	linear
1	logarithmic

5.6.29 IntPrj

Overview

- Activate
- AddProjectToCombined
- AddProjectToRemoteDatabase
- Archive
- BeginDataExtensionModification
- CanAddProjectToRemoteDatabase
- CanSubscribeProjectReadOnly
- CanSubscribeProjectReadWrite
- ClearInvalidReferences
- CopyDataExtensionFrom
- CreateVersion
- Deactivate
- EndDataExtensionModification
- GetDerivedProjects
- GetExternalReferences
- GetGeoCoordinateSystem
- GetLatestVersion
- GetVersions
- HasExternalReferences
- LoadData
- MergeToBaseProject
- Migrate
- NormaliseCombined
- PackExternalReferences
- Purge
- RemoveProjectFromCombined
- Restore
- SetGeoCoordinateSystem

[SubscribeProjectReadOnly](#)
[SubscribeProjectReadWrite](#)
[TransformGeoCoordinates](#)
[UnsubscribeProject](#)
[UpdateStatistics](#)
[UpdateToDefaultStructure](#)
[UpdateToMostRecentBaseVersion](#)

Activate

Activates the project. If another project is already activated it will be deactivated first.

```
int IntPrj.Activate()
```

RETURNS

0	on success
1	on error

AddProjectToCombined

Adds a project to this using the Project Combination logic. The passed object must be an `IntVersion`. The receiving project must be activated but not have a Study Case active, otherwise this method will fail.

```
int IntPrj.AddProjectToCombined(object projectVersion)
```

ARGUMENTS

projectVersion
The version of a project to add

RETURNS

0	operation was successful
1	an error occurred

AddProjectToRemoteDatabase

Adds a project to the online database if possible.

Can only be used if the database driver is set to Offline Mode.

```
int IntPrj.AddProjectToRemoteDatabase()
```

Archive

Archives the project if the functionality is configured and activated. Does nothing otherwise.

```
int IntPrj.Archive()
```

RETURNS

0	project has been archived
1	project has not been archived

BeginDataExtensionModification

Signals the start of a Data Extension modification.

Must be terminated with `EndDataExtensionModification`, otherwise all changes will be discarded. Data Extensions can only be modified when the project is active.

```
DataObject IntPrj.BeginDataExtensionModification()
```

RETURNS

The `SetDataext` configuration object in the settings folder for further processing.

SEE ALSO

[IntPrj.EndDataExtensionModification\(\)](#), [SetDataext.AddConfiguration\(\)](#), [SetDataext.GetConfiguration\(\)](#), [SetDataext.GetConfigurations\(\)](#), [SetDataext.RemoveAllConfigurations\(\)](#), [SetDataext.RemoveConfiguration\(\)](#)

CanAddProjectToRemoteDatabase

Checks if the project can be pushed to the remote database.

The project must be subscribable as read and write and it must be unsubscribed. Can only be used if the database driver is set to Offline Mode.

```
int IntPrj.CanAddProjectToRemoteDatabase()
```

RETURNS

- 0** project cannot be added to the remote database
- 1** project can be added to the remote database

CanSubscribeProjectReadOnly

Checks if a project can be subscribed read-only by the user executing the script.

```
int IntPrj.CanSubscribeProjectReadOnly()
```

RETURNS

- 0** no permission to subscribe project
- 1** project can be subscribed

CanSubscribeProjectReadWrite

Checks if a project can be subscribed read-write by the user executing the script.

```
int IntPrj.CanSubscribeProjectReadWrite()
```

RETURNS

- 0** no permission to subscribe project
- 1** project can be subscribed

ClearInvalidReferences

Removes all references from the project which can no longer be resolved due to target objects being deleted or projects no longer being available in the database, i.e. project stubs and non-migrated projects.

```
int IntPrj.ClearInvalidReferences(int type)
```

ARGUMENTS

type

- | | |
|----------|--|
| 0 | Remove references to deleted objects |
| 1 | Remove references to project stubs and non-migrated projects |
| 2 | Remove both |

RETURNS

Number of references cleared.

CopyDataExtensionFrom

Copies the Data Extension configuration from another project into this project. The configuration is applied immediately. The target project must be active.

```
int IntPrj.CopyDataExtensionFrom(object other)
```

ARGUMENTS

other Project to copy the Data Extension configuration from.

RETURNS

- | | |
|----------|---|
| 0 | Copied successfully |
| 1 | Target project is not active |
| 2 | Source object is not a project |
| 3 | Error during copying, see Output Window |

CreateVersion

Creates a new version of project it was called on.

Optionally allows to pass the version name, the timestamp, a bool for user notification, a bool to enforce project approval and a description.

```
DataObject IntPrj.CreateVersion([str name = "",]
                                [int timestamp = 0,]
                                [int notifyUsers = 0,]
                                [int approvalRequired = 0,]
                                [str description = ""])
DataObject IntPrj.CreateVersion(int notifyUsersAndApprovalRequired,
                                [str name = "",]
                                [float timestamp = 0,]
                                [str description = ""])
```

ARGUMENTS

name Version name. The default version name e.g. 'Project Version' is used on an empty string. (default: empty string)

timestamp Seconds since 01.01.1970 00:00:00. On zero the current date and time is used. (default: 0)

notifyUsers
User notifications activated:

- 0** Create version and do not notify users (default).
- 1** Notify users.

approvalRequired
Project approval required:

- 0** Create version without approval (default).
- 1** Require approval.

notifyUsersAndApprovalRequired
Project approval required and user notifications activated:

- 0** Create version without approval and do not notify users (default).
- 1** Require approval and notify users.

description
Version description. (default: empty string)

RETURNS

- DataObject** Newly created *IntVersion* object.
- None** On failure e.g. missing permission rights.

Deactivate

De-activates the project if it is active. Does nothing otherwise.

```
int IntPrj.Deactivate()
```

RETURNS

- 0** on success
- 1** on error

EndDataExtensionModification

Terminates a Data Extension modification previously initiated with *BeginDataExtensionModification*.

This will deactivate the Study Case if the new Data Extension configuration can be applied. In case of errors the project will be deactivated and rolled back. Omitting the call to *EndDataExtensionModification* and exiting from the script will discard all changes to the Data Extension configuration.

```
None IntPrj.EndDataExtensionModification()
```

SEE ALSO

[IntPrj.BeginDataExtensionModification\(\)](#)

GetDerivedProjects

Return a set holding all versions created in the project.

```
list IntPrj.GetDerivedProjects()
```

RETURNS

Set holding all versions of a project.

GetExternalReferences

Fills the given map with objects from this project mapping to its external references.

```
int IntPrj.GetExternalReferences(DataObject resultMap,  
                                [DataObject externalReferencesSettings])
```

ARGUMENTS

resultMap

DPL map (IntDplmap) which will contain objects mapping to its external references. Objects without external references are not mapped.

externalReferencesSettings (optional)

External References settings object (SetExtref). Defines the properties for objects being external references. The External References settings object from current user is used if omitted.

RETURNS

- 0** Getting external references succeeded.
- 1** Getting external references cancelled in dialogue or failed e.g project inactive or not migrated.

SEE ALSO

[IntPrj.HasExternalReferences\(\)](#), [IntPrj.PackExternalReferences\(\)](#)

GetGeoCoordinateSystem

Returns the EPSG code of the geographic coordinate system in which the geo coordinates of the project's net elements are stored.

```
int IntPrj.GetGeoCoordinateSystem()
```

RETURNS

EPSG code of the project's geographic coordinate system, or 0 if the coordinate system is not known.

GetLatestVersion

Returns the most recent version available in the project which has the notify users option set. Optionally allows to consider all versions, regardless of notify users option.

```
DataObject IntPrj.GetLatestVersion([int onlyregular])
```

ARGUMENTS

onlyregular (optional)

- | | |
|----------|---|
| 1 | consider only regular version (default) |
| 0 | consider all versions |

RETURNS

Latest version of the project

GetVersions

Returns a set containing all versions of the project.

```
list IntPrj.GetVersions()
```

RETURNS

Set that contains all versions of the project

HasExternalReferences

Checks if any object inside the project references external non-system objects and prints a report to the Output Window.

```
int IntPrj.HasExternalReferences([int considerGlobal = 1,
                                [int considerRemoteVariants = 0]
                                ])
int IntPrj.HasExternalReferences(DataObject externalReferencesSettings)
```

ARGUMENTS

considerGlobal (optional)

- | | |
|----------|--|
| 0 | References to global (non-system) objects are not considered as external references. |
| 1 | References to global (non-system) objects are considered as external references (default). |

considerRemoteVariants (optional)

- | | |
|----------|--|
| 0 | References to remote variants are not considered as external references (default). |
| 1 | References to remote variants are considered as external references. |

externalReferencesSettings (optional)

External References settings object (SetExtref). Defines the properties for objects being external references.

RETURNS

- | | |
|----------|----------------------------------|
| 0 | No external reference was found. |
| 1 | An external reference was found. |

SEE ALSO

[IntPrj.PackExternalReferences\(\)](#), [IntPrj.GetExternalReferences\(\)](#)

LoadData

Loads all objects of the project from the data base. Does nothing when called on an active project.

This function is useful to optimise searches which would traverse deep into an inactive project.

```
None IntPrj.LoadData()
```

MergeToBaseProject

Merges the modifications of a derived project to the base project.

```
int IntPrj.MergeToBaseProject(int conflictMode)
```

ARGUMENTS

conflictMode

Assignment in case of modification conflict:

- | | |
|----------|--|
| 0 | Favour none. Diff browser is shown in case of conflicts. |
| 1 | Favour base version. |
| 2 | Favour derived project. |
| 3 | Favour base project. |

RETURNS

- | | |
|----------|---|
| 0 | Merge finished successfully. |
| 1 | Merge failed (no further information). |
| 2 | Merge failed due to failing assignment check. |
| 3 | Merge failed due to invalid projects (e.g. no derived project). |
| 4 | Merge was canceled by user. |
| 5 | Merge completed with errors (see output window). |

Migrate

Migrates a project from version V13 to V14. Migration is only executed if project has been created in build 400 or earlier (and is not yet migrated).

```
None IntPrj.Migrate([int createCopy])
```

ARGUMENTS

createCopy (optional)

- | | |
|----------|---|
| 1 | Creates a copy of current project (original copy is maintained) (default) |
| 0 | Does an "in-place" migration of the project (original is overwritten) |

NormaliseCombined

Normalises a combined project so it appears to be a regular project. This will remove all intermediate folders which were added when creating the combined project. This might lead to naming duplications which will be resolved by the normal logic of numbering duplicates.

```
int IntPrj.NormaliseCombined()
```

RETURNS

- 0** operation was successful
- 1** operation was cancelled because the project is active

PackExternalReferences

Packs external references of this project and prints a report to the Output Window.

```
int IntPrj.PackExternalReferences([DataObject externalReferencesSettings])
```

ARGUMENTS

externalReferencesSettings (optional)

External References settings object (SetExtref). Defines the properties for objects being external references. The External References settings object from current user is used if omitted.

RETURNS

- 0** Packing external references succeeded.
- 1** Packing external references cancelled in dialogue or failed e.g project inactive or not migrated.

SEE ALSO

[IntPrj.HasExternalReferences\(\)](#), [IntPrj.GetExternalReferences\(\)](#)

Purge

Purges project storage and updates storage statistics.

Requires write access to the project; the functions does nothing when the project is locked by another user.

```
None IntPrj.Purge()
```

RemoveProjectFromCombined

Removes a project from a combined project. For the removal the mapping key must be specified. Mapping keys are stored in the project, parameter `project.mapped`. The project this method is called on must be activated but not have a Study Case active, otherwise this method will fail.

```
int IntPrj.RemoveProjectFromCombined(str mappingKey)
```

ARGUMENTS

mappingKey

The mapping key for the project that should be removed

RETURNS

- 0** operation was successful
- 1** an unknown error occurred
- 2** an error occurred and is documented in the output window

Restore

Restores an archived project so it can be used again. Does nothing if the project is not an archived one.

```
int IntPrj.Restore()
```

RETURNS

- 0** project has not been restored
- 1** project has been restored

SetGeoCoordinateSystem

Defines the the geographic coordinate system in which the geo coordinates of the project's net elements are stored.

```
None IntPrj.SetGeoCoordinateSystem(int epsgCode)
```

ARGUMENTS

epsgCode

EPSG code of the geographic coordinate system to be used for this project. A value of 0 denotes an unknown coordinate system.

SubscribeProjectReadOnly

Subscribes a project read only if the permission is granted.

Can only be used if the database driver is set to Offline Mode.

```
None IntPrj.SubscribeProjectReadOnly()
```

SubscribeProjectReadWrite

Subscribes a project read/write if the permission is granted.

Can only be used if the database driver is set to Offline Mode.

```
None IntPrj.SubscribeProjectReadWrite()
```

TransformGeoCoordinates

Transforms geographic coordinates from a source coordinate system to a destination coordinate system. Geodetic coordinates (longitude/latitude) are specified in decimal degrees. Projected coordinates are specified in meters.


```
[int error,
float xOut,
float yOut] IntPrj.TransformGeoCoordinates(float xIn, float yIn,
                                           int sourceEpsg, int destEpsg
                                           )
```

ARGUMENTS

- xIn* horizontal component of the input location: x position or longitude, resp.
- yIn* vertical component of the input location: y position or latitude, resp.
- xOut (out)* horizontal component of the output location: x position or longitude, resp.
- yOut (out)* vertical component of the output location: y position or latitude, resp.
- sourceEpsg* EPSG code of the source coordinate system
- destEpsg* EPSG code of the destination coordinate system

RETURNS

- 0** operation was successful
- 1** an error occurred

UnsubscribeProject

Unsubscribes a project.

Can only be used if the database driver is set to Offline Mode.

```
None IntPrj.UnsubscribeProject()
```

UpdateStatistics

Updates the storage statistics for a project. The statistics are displayed on the page Storage of a project.

Note: This function requires write access to the project otherwise the update is not executed and an error message is printed to the output window.

```
None IntPrj.UpdateStatistics()
```

UpdateToDefaultStructure

Updates folder structure of currently active project to that of the default project (used for creation of new projects). Existing folders will be moved to a new location within the project. Folders that have no correspondence in the default project will remain untouched.

NB: Folders might get moved or additional ones might be created, but no folder is deleted by this routine.

```
int IntPrj.UpdateToDefaultStructure(int createMissingFolders, int updateForeignKeys)
```

ARGUMENTS

createMissingFolders (optional)

- 0** Missing folders will not be created. Only existing ones will be moved to new locations, if required.
- 1** Missing folders will be created (default).

updateForeignKeys (optional)

- 0** Foreign-keys of folders will not be updated.
- 1** Foreign-keys of folders will be updated (default).

RETURNS

- 0** operation was successful
- 1** operation was not successful, e.g. project not active

UpdateToMostRecentBaseVersion

Updates a derived project to the most recent version of the base project. This is done by creating a new derived project from the new version and (optionally) merging the modifications from the existing derived project into it.

```
int IntPrj.UpdateToMostRecentBaseVersion(DataObject newDerivedProject,
                                         int versionWithNotification,
                                         int discardOwnModifications,
                                         [int conflictMode = 0]
                                         )
```

ARGUMENTS

newDerivedProject (out)

New derived project if no error occurred.

versionWithNotification

- 0** Update to the most recent version independent of the "Notify users" setting.
- 1** Update to the most recent version with "Notify users" enabled.

discardOwnModifications

- 0** Merge modifications of new version and derived project.
- 1** Discard modifications of derived project. The Compare and Merge Tool is not used at all.

conflictMode (optional)

Assignment in case of modification conflict (only used if not discarding derived modifications):

- 0** Favour none. Diff browser is shown in case of conflicts. (default)
- 1** Favour new base version.
- 2** Favour derived project.

RETURNS

- | | |
|----------|---|
| 0 | Merge finished successfully. |
| 1 | Merge failed (no further information). |
| 2 | Merge failed due to failing assignment check. |
| 3 | Merge failed due to invalid projects (e.g. no derived project). |
| 4 | Merge was canceled by user. |
| 5 | Merge completed with errors (see output window). |

5.6.30 IntPrjfolder

Overview

[GetProjectFolderType](#)
[IsProjectFolderType](#)

GetProjectFolderType

Returns the type of the project folder stored in attribute "iopt_type".
The following types are currently available (language independent):

- blk - User Defined Models
- cbrat - CB Ratings
- chars - Characteristics
- cstgen - Generator Cost Curves
- effgen - Generator Efficiency Curves
- rnd - Probabilistic Assessment
- cim - CIM Model
- common - Common Mode Failures
- demand - Demand Transfers
- dia - Diagrams
- equip - Equipment Type Library
- fault - Faults
- ras - Remedial Action Schemes
- gen - Generic
- lib - Library
- mvar - Mvar Limit Curves
- netdat - Network Data
- netmod - Network Model
- oplib - Operational Library
- outage - Outages

- qpc - QP-Curves
- ra - Running Arrangements
- report - Table Reports
- scen - Operation Scenarios
- scheme - Variations
- script - Scripts
- study - Study Cases
- sw - StationWare
- tariff - Tariffs
- templ - Templates
- therm - Thermal Ratings
- ucc - V-Control-Curves

```
str IntPrjfolder.GetProjectFolderType()
```

RETURNS

The type of the project folder as string. For possible return values see list above.

SEE ALSO

[Application.GetProjectFolder\(\)](#)

IsProjectFolderType

This function checks if a project folder is of given type.

```
int IntPrjfolder.IsProjectFolderType(str type)
```

ARGUMENTS

type Folder type; for possible type values see [IntPrjfolder.GetProjectFolderType\(\)](#)

RETURNS

- 1** true, is of given type
- 0** false, is not of given type

SEE ALSO

[Application.GetProjectFolder\(\)](#), [IntPrjfolder.GetProjectFolderType\(\)](#)

5.6.31 IntQlim

Overview

[GetQlim](#)

GetQlim

Returns either the current maximum or the minimum reactive power limit, given the specified active power and voltage.

The active power must be given in the same units as the input mode definition of the capability curve object (parameter "inputmod" is 0 for MW/Mvar and 1 for p.u.).

```
float IntQlim.GetQlim(float p,
                     float v,
                     [float minmax]
                     )
```

ARGUMENTS

p the current value of active power in MW or p.u.

v the current value of voltage in p.u.

minmax (optional)

Returns either the maximum or minimum value. Possible values are:

-1	minimum value
1	maximum value. This is the default value

RETURNS

Returns the minimum/maximum limit. The units might be Mvar or p.u., depending on the input mode of the capability curve. Also, the limits are calculated for a single machine.

5.6.32 IntRas

Overview

[AddEvent](#)
[AddTrigger](#)
[IsValid](#)

AddEvent

Adds an event to the Remedial Action Scheme.

```
list IntRas.AddEvent(DataObject newEvent)
```

ARGUMENTS

newEvent
 The event that shall be added.

AddTrigger

Adds either a condition or a gate to the Remedial Action Scheme.

```
list IntRas.AddTrigger(DataObject newTrigger)
```

ARGUMENTS

newTrigger
 The condition or gate that shall be added.

IsValid

Checks if the Remedial Action Scheme is valid.

```
int IntRas.IsValid(int emitMessage = 0)
```

ARGUMENTS

emitMessage
Emit messages if not equal to zero.

RETURNS

0 If invalid.
1 If valid.

5.6.33 IntRunarrange**Overview**

[GetSwitchStatus](#)

GetSwitchStatus

Determines the status of the given switch in the running arrangement, without assigning or applying the running arrangement.

```
int IntRunarrange.GetSwitchStatus(DataObject switch)
```

ARGUMENTS

switch *ElmCoupe* or *StaSwitch* from which to get the status stored in running arrangement

RETURNS

Status of the switch in the running arrangement. Possible values are

-1 Switch is not part of the running arrangement
0 Switch is open
1 Switch is closed

5.6.34 IntScenario**Overview**

[Activate](#)
[Apply](#)
[ApplySelective](#)
[Deactivate](#)
[DiscardChanges](#)
[GetObjects](#)
[GetOperationValue](#)
[ReleaseMemory](#)
[Save](#)
[SetOperationValue](#)

Activate

Activates a scenario. If there is currently another scenario active that one will be deactivated automatically.

```
int IntScenario.Activate()
```

RETURNS

- 0** successfully activated
- 1** error, e.g. already activate, no project and study case active

Apply

Copies the values stored in a scenario to the corresponding network elements. The value transfer is identical to scenario activation, however, the scenario will not be activated. In case of having an active variation or another scenario, the values will be recorded there.

```
int IntScenario.Apply([int requestUserConfirmation])
int IntScenario.Apply(int requestUserConfirmation,
                      DataObject parentfilter
                      )
```

ARGUMENTS

requestUserConfirmation(optional)

- 0** silent, just apply the data without further confirmation requests
- 1** request a user confirmation first (default)

parentfilter (optional)

If given, scenario data is only applied for given object and all of its children (hierarchical filter)

RETURNS

- 0 on success

ApplySelective

Similar to function Apply() but copies only the set of attributes listed in the given apply configuration. An apply configuration is a folder consisting of variable selection objects (IntMon), one per class. For each class the attributes to be copied can be selected.

```
int IntScenario.ApplySelective(DataObject configuration)
int IntScenario.ApplySelective(int requestUserConfirmation,
                              DataObject applyConfiguration
                              )
```

ARGUMENTS

applyConfiguration

folder containing variable selection objects

requestUserConfirmation(optional)

- 0** silent, just apply the data without further confirmation requests
- 1** request a user confirmation first (default)

RETURNS

0 on succes

Deactivate

Deactivates the currently active scenario.

```
int IntScenario.Deactivate([int saveOrUndo])
```

ARGUMENTS

saveOrUndo(optional)

Determines whether changes in active scenario will be saved or discarded before the scenario is deactivated. If this argument is omitted, the user will be asked.

0 discard changes

1 save changes

RETURNS

0 on success

DiscardChanges

Discards all unsaved changes made to a scenario.

```
int IntScenario.DiscardChanges()
```

RETURNS

0 on success

1 error, scenario was not modified or not active

GetObjects

Returns a set of all objects for which operational data are stored in scenario.

```
list IntScenario.GetObjects()
```

RETURNS

Set of all objects for which operational data are stored in scenario

GetOperationValue

This function offers read access to the operation data values stored in the scenario.

```
[int error,
int|float|str|DataObject value] IntScenario.GetOperationValue(DataObject obj,
                                                                str attribute,
                                                                [int fromObject])
```


ARGUMENTS

- value (out)* variable that holds the value after call
- obj* object for which the operation to be retrieved
- attribute* name of the operation data attribute
- fromObject* only if current scenario is active:
- 0** value is taken from scenario (as stored on db)
 - 1** (default), value is taken from object (reflects un-saved modifications)

RETURNS

0 on success

ReleaseMemory

Releases the memory used by a scenario. Any further access to the scenario will reload the data from database. The function can be called on inactive scenarios only. Use this function with care!

```
int IntScenario.ReleaseMemory()
```

RETURNS

- 0** on success
- 1** error, scenario is active

Save

Saves the current active value of all operational attributes for all active network elements to database.

```
int IntScenario.Save()
```

RETURNS

- 0** successfully saved
- 1** error, scenario was not modified or not active

SetOperationValue

Offers write access to operational data stored in a scenario.

```
int IntScenario.SetOperationData(int|float|str|DataObject newvalue,
                                DataObject obj,
                                str attribute,
                                [int toObject = 1]
                                )
```

ARGUMENTS

- newvalue* New value to store in the scenario.
- obj* Object for which the operation data to store.
- attribute* Name of the operation data attribute.
- toObject* Only if current scenario is active:
- 0** Value is only stored to scenario on db.
 - 1** As 0 but value is also updated on object in memory. (default)

RETURNS

0 on success, otherwise 1.

5.6.35 IntScensched

Overview

[Activate](#)
[Deactivate](#)
[DeleteRow](#)
[GetScenario](#)
[GetStartEndTime](#)
[SearchScenario](#)

Activate

Activates a scenario scheduler.

```
int IntScensched.Activate()
```

RETURNS

- 0** successfully activated
- 1** error, e.g. already activate, no project and study case active

Deactivate

Deactivates a scenario scheduler.

```
int IntScensched.Deactivate()
```

RETURNS

- 0** successfully deactivated
- 1** error, e.g. already deactivates, no project and study case active

DeleteRow

Delete row(s) of the scenario scheduler.

```
None IntScensched.DeleteRow(int row, [int numberOfRows])
```

ARGUMENTS

row row number (begin with 0)

numberOfRows (optional)
number of rows to delete (default = 1)

GetScenario

Get the scenario for corresponding time 'iTime'.

```
DataObject IntScensched.GetScenario(int iTime)
```

ARGUMENTS

iTime Time (UCTE) to get the corresponding scenario.

RETURNS

None No scenario at time 'iTime' defined

IntScenario Scenario will be activated at time 'iTime'

GetStartEndTime

Get start and end time of the corresponding scenario.

```
[int error  
int startTime,  
int endTime ] IntScensched.GetStartEndTime(DataObject scenario)
```

ARGUMENTS

scenario A scenario (*IntScenario*).

startTime (out)
Start time (time when the scenario is activated)).

endTime (out)
End time (time until the scenario is still activated).

RETURNS

-1 Scenario not found (not part of scenario scheduler)

≥ 0 Vector index (index of scenario)

SearchScenario

Search at which table index (row) the corresponding scenario is defined in the scheduler.

```
int IntScensched.SearchScenario(DataObject scenarioObject)
```

ARGUMENTS

scenarioObject
scenario object

RETURNS

- −1 Scenario not found (not part of scenario scheduler).
- ≥ 0 Vector index (row, index of scenario).

5.6.36 IntScheme

Overview

[Activate](#)
[Consolidate](#)
[Deactivate](#)
[GetActiveScheduler](#)
[NewStage](#)

Activate

Activates a variation and inserts a variation reference in a 'Variation Configuration Folder'stored in the study case.

```
int IntScheme.Activate()
```

RETURNS

- 0** successfully activated
- 1** error, e.g. already activate, no project and study case active

Consolidate

Changes that are recorded in this variation will be permanently applied to the original location.
Note: Modified scenarios are not saved.

Works only:

- for non network variation e.g. used for Mvar Limit Curves, Thermal Ratings ...
- and the variation must be activated.

```
int IntScheme.Consolidate()
```

RETURNS

- 0** On success.
- 1** If an error has occurred.

Deactivate

Deactivates a variation and removes the variation reference in the 'Variation Configuration Folder'stored in the study case.

```
int IntScheme.Deactivate()
```

RETURNS

- 0** successfully deactivated
- 1** error, e.g. already deactivated, no project and study case active

GetActiveScheduler

Returns the corresponding active variation scheduler or None if no scheduler is active for this variation (IntScheme).

```
DataObject IntScheme.GetActiveScheduler()
```

NewStage

Adds a new expansion stage into the variation (name = sname).

```
int IntScheme.NewStage(str name,
                       int activationTime,
                       int activate
                       )
```

ARGUMENTS

- name* Name of the new expansion stage.
- activationTime*
Activation time of the new expansion stage in seconds since 01.01.1970 00:00:00.
- activate*
 - 1** The actual study time is changed to the parameter iUTCtime and the variation will be activated. If the variation is a network variation, the new created expansion stage is used as 'recording' expansion stage. If the variation (this) is not active, the variation will be automatically activated.
 - 0** Expansion stage and/or variation will not be activated.

5.6.37 IntScheduler**Overview**

[Activate](#)
[Deactivate](#)
[Update](#)

Activate

Activates a variation scheduler. An already activated scheduler for same variation will be deactivated automatically.

```
int IntScheduler.Activate()
```

RETURNS

- = 0** On success

$\neq 0$ If an error has occurred

Deactivate

Deactivates a variation scheduler.

```
int IntSscheduler.Deactivate()
```

RETURNS

$= 0$ on success
 $\neq 0$ If an error has occurred especially if scheduler was not active (to be consistent with scenario scheduler deactivate()).

Update

Update variation scheduler (updates internal reference stages).

```
int IntSscheduler.Update()
```

RETURNS

$= 0$ On success
 $\neq 0$ If an error has occurred

5.6.38 IntSstage

Overview

[Activate](#)
[CreateStageObject](#)
[EnableDiffMode](#)
[GetVariation](#)
[IsExcluded](#)
[PrintModifications](#)
[ReadValue](#)
[WriteValue](#)

Activate

Activates the expansion stage and sets the 'recording' expansion stage. The study time will be automatically set to the correponsing time of the stage.

```
int IntSstage.Activate([int iQueryOption])
```

ARGUMENTS

iQueryOption

0 (default) The user must confirm the query.
1 The “Yes” button is automatically applied.
2 The “No” button is automatically applied.

RETURNS

0 Successfully activated.

- 1 Error, e.g. scheme is not active.

CreateStageObject

Creates a stage object (delta or delete object) inside corresponding *IntSstage*.

```
DataObject IntSstage.CreateStageObject(int type,
                                         DataObject rootObject
                                         )
```

ARGUMENTS

type Kind of object to create

1	Delete object
2	Delta object

rootObject
(Original) object for which the stage object should be created.

RETURNS

Stage object on success.

EnableDiffMode

Enables the comparison mode for the variation management system. If the mode is enabled a DELTA object is only created when the object is different.

```
None IntSstage.EnableDiffMode(int enable)
```

ARGUMENTS

enable

0	disables the difference/comparison mode
1	enables the difference/comparison mode

GetVariation

Returns variation of expansion stage.

```
DataObject IntSstage.GetVariation()
```

RETURNS

Variation object corresponding to stage.

DEPRECATED NAMES

GetScheme

IsExcluded

Returns if expansion stage flag 'Exclude from Activation' is switched on (return value = 1) or not (return value = 0). The function checks also if the stage is excluded regarding the restricted validity period of the corresponding variation and considers also the settings of an variation scheduler when defined.

```
float IntSstage.IsExcluded()
```

RETURNS

- 1** if stage is excluded
- 0** if stage is considered

PrintModifications

Reports in the the output window the modification of the corresponding expansion stage. Works only if the expansion stage is the active 'recording' expansion stage.

```
int IntSstage.PrintModifications([int onlyNetworkData,]
                                [str ignoredParameter]
                                )
```

ARGUMENTS

onlyNetworkData (optional)

- 1** (default) Show only network data modifications. Graphical modifications are not report when the diagrams folder are recored.
- 0** Show all modifications

ignoredParameter (optional)

Comma separated list of parameters which are ignored for reporting.

RETURNS

- 0** on success
- 1** if the actual expansion stage is not the 'recording' expansion stage.

ReadValue

Get the value for an attribute of an ADD or DELTA object which modifies "rootObj" (root object).

```
[int error,
float|str|DataObject value] IntSstage.ReadValue(DataObject rootObj,
                                                str attributeName)
```

RETURNS

- = 0** On success.
- ≠ 0** Error e.g. wrong data type.

WriteValue

Writes a value for an attribute to an ADD or DELTA object which modifies rootObj (root object).

```
int IntSstage.WriteValue(float|str|DataObject value,
                          DataObject rootObj,
                          str attributeName)
```


RETURNS

- = 0 On success.
- ≠ 0 Error e.g. wrong data type.

5.6.39 IntSubset

Overview

[Apply](#)
[ApplySelective](#)
[Clear](#)
[GetConfiguration](#)
[GetObjects](#)

Apply

Copies the values stored in a scenario to the corresponding network elements. The value transfer is identical to scenario activation, however, the scenario will not be activated. In case of having an active variation or another scenario, the values will be recorded there.

```
int IntSubset.Apply([int requestUserConfirmation])
```

ARGUMENTS

requestUserConfirmation(optional)

- 0** silent, just apply the data without further confirmation requests
- 1** request a user confirmation first (default)

RETURNS

0 on success

ApplySelective

Similar to function Apply() but copies only the set of attributes listed in the given apply configuration. An apply configuration is a folder consisting of variable selection objects (IntMon), one per class. For each class the attributes to be copied can be selected.

```
int IntSubset.ApplySelective(DataObject applyConfiguration)
int IntSubset.ApplySelective(int requestUserConfirmation,
                             DataObject applyConfiguration
                             )
```

ARGUMENTS

applyConfiguration

folder containing variable selection objects

requestUserConfirmation(optional)

- 0** silent, just apply the data without further confirmation requests
- 1** request a user confirmation first (default)

RETURNS

0 on succes

Clear

Clears all values stored in the subset.

Please note that this function can only be called on subsets of currently in-active scenarios.

```
int IntSubset.Clear()
```

RETURNS

- 0** On success.
- 1** On error, e.g. subset belongs to a currently active scenario.

GetConfiguration

Returns class and attribute configuration for the subset.

```
str IntSubset.GetConfiguration(str subset)
```

ARGUMENTS

subset

RETURNS

Returns a list of classes and attributes for which operational data are stored in subset.

GetObjects

Returns a set of all objects for which operational data are stored in the subset.

```
list IntSubset.GetObjects()
```

RETURNS

Set of all objects for which operational data are stored in the subset

5.6.40 IntThrating**Overview**

[GetCriticalTimePhase](#)
[GetRating](#)

GetCriticalTimePhase

This function returns the smallest duration (time-phase) for which the power flow is beyond the rating.

```
float IntThrating.GetCriticalTimePhase(float Flow,
                                       float Loading
                                       )
```

ARGUMENTS

- Flow* Power from the load flow calculation, in MVA.
- Loading* Element loading, in %.

RETURNS

- >0** Smallest time-phase for which the flow is beyond the rating.
- 1** In case that no rating is violated.

GetRating

This function returns the rating in MVA according to the thermal rating table, considering element overloading and its duration (time-phase).

```
float IntThrating.GetRating(float Loading,
                           float Duration
                           )
```

ARGUMENTS

- Loading* Element loading, in %.
- Duration* Duration or time phase for which the loading is considered, in minutes

RETURNS

Rating in MVA or 0 if not found.

5.6.41 IntUrl**Overview**

[View](#)

View

Requests the operating system to open given URL for viewing. The performed action depends on the default action configured in the system. For example, by default 'http://www.google.com' would be opened in standard browser.

Please note, the action is only executed if access to given URL is enabled in the 'External Access' configuration of PowerFactory (IntExtaccess).

```
int IntUrl.View()
```

RETURNS

The returned value reports the success of the operation:

- 0** Success, URL was opened
- 1** Error, URL was not opened (because of invalid address or security reasons)

5.6.42 IntUser**Overview**

[Purge](#)
[SetPassword](#)
[TerminateSession](#)

Purge

Purges project storage and updates storage statistics for all projects of the user.

Requires write access to the project; the functions does nothing when the project is locked by another user.

```
None IntUser.Purge()
```

SetPassword

Sets the password for the user the function is called on.

Note: Only the administrator user is allowed to use this function. He can (re-)set the password for every user.

```
int IntUser.SetPassword(str newpassword)
```

ARGUMENTS

newpassword
Case sensitive user password to set

RETURNS

Returns whether or not the password has successfully been set. Possible values are:

0	error
1	password set successfully

TerminateSession

Allows the Administrator to log out another user. Prints an error if the current user is not the Administrator.

```
None IntUser.TerminateSession()
```

5.6.43 IntUserman

Overview

[CreateGroup](#)
[CreateUser](#)
[GetGroups](#)
[GetUsers](#)
[UpdateGroups](#)

CreateGroup

Creates a new user group of given name. If a group with given name already exists the existing one is returned instead.

Note: Only Administrator user is allowed to call this function.

```
DataObject IntUserman.CreateGroup(str name)
```

ARGUMENTS

name Given name of the user group

RETURNS

Created user group (IntGroup)

CreateUser

Creates a new user with given name. If the user already exists the existing one is returned instead.

Note: Only Administrator user is allowed to call this function.

```
DataObject IntUserman.CreateUser(str name)
```

ARGUMENTS

name Given name of the user

RETURNS

Created user (IntUser)

GetGroups

Returns a container with all user groups.

Note: Only the administrator user is allowed to call this function.

```
list IntUserman.GetGroups()
```

RETURNS

Set of all available users

GetUsers

Returns a container with all users as they are currently visible in the Data Manager tree.

Note: Only the administrator user is allowed to call this function.

```
list IntUserman.GetUsers()
```

RETURNS

Set of all available users

UpdateGroups

Updates the Everybody group so it contains all currently existing users and cleans it of removed users.

```
None IntUserman.UpdateGroups()
```

5.6.44 IntVec

Overview

[Get](#)
[Init](#)
[Max](#)
[Mean](#)
[Min](#)
[Resize](#)
[Save](#)
[Set](#)
[Size](#)
[Sort](#)

Get

Get the value in row index. Index is one based, therefore the index of the first entry is 1.

```
float IntVec.Get(int index)
```

ARGUMENTS

index Index in vector, one based.

SEE ALSO

[IntVec.Set\(\)](#)

Init

Initializes the vector. Resizes the vector and initializes all values to 0.

This operation is performed in memory only. Use [IntVec.Save\(\)](#) to save the modified vector to database.

```
None IntVec.Init(int size)
```

ARGUMENTS

size The new size of the vector.

Max

Gets the maximum value stored in the vector.

```
float IntVec.Max()
```

RETURNS

The maximum value stored in the vector. Empty vectors return 0 as maximum value.

Mean

Calculates the average value of the vector.

```
float IntVec.Mean()
```

RETURNS

The average value of the vector. A value of 0. is returned for empty vectors.

Min

Gets the minimum value stored in the vector.

```
float IntVec.Min()
```

RETURNS

The minimum value stored in the vector. Empty vectors return 0 as minimum value.

Resize

Resizes the vector. Inserted values are initialized to 0.

This operation is performed in memory only. Use [IntVec.Save\(\)](#) to save the modified vector to database.

```
None IntVec.Resize(int size)
```

ARGUMENTS

size The new size.

Save

Saves the current state of this vector to database.

```
None IntVec.Save()
```

Set

Set the value in row index. Index is one based, therefore the index of the first entry is 1.

The vector is resized automatically to size index in case that the index exceeds the current vector size. Values inserted are automatically initialized to a value of 0.

This operation is performed in memory only. Use [IntVec.Save\(\)](#) to save the modified vector to database.

```
None IntVec.Set(int index,  
                float value)
```

ARGUMENTS

index Index in vector.

value Value to assign in row index.

SEE ALSO

[IntVec.Get\(\)](#)

Size

Returns the size of the vector.

```
int IntVec.Size()
```

RETURNS

The size of the vector.

Sort

Sorts the vector.

This operation is performed in memory only. Use [IntVec.Save\(\)](#) to save the modified vector to database.

```
None IntVec.Sort([int ascending = 0])
```

ARGUMENTS

ascending

Sort order:

- 0** Highest value first (descending, default).
- 1** Smallest value first (ascending).

5.6.45 IntVecobj

Overview

[Get](#)
[Resize](#)
[Save](#)
[Search](#)
[Set](#)
[Size](#)

Get

Get the object in row index. Index is one based, therefore the index of the first entry is 1.

```
DataObject IntVecobj.Get(int index)
```

ARGUMENTS

index Index in vector, one based.

SEE ALSO

[IntVecobj.Set\(\)](#)

Resize

Resizes the vector. Inserted new entries are initialized to None.

This operation is performed in memory only. Use [IntVecobj.Save\(\)](#) to save the modified vector to database.


```
None IntVecobj.Resize(int size)
```

ARGUMENTS

size The new size.

Save

Saves the current state of this vector to database.

```
None IntVecobj.Save()
```

Search

Search if the object (obj) is part of the vecor and returns the corresponding index of the vector (one based).

```
int IntVecobj.Search(DataObject obj)
```

RETURNS

- 1 ... size** object found, located at index
- 0** object not part of vector

Set

Set the object (obj) in row index. Index is one based, therefore the index of the first entry is 1. The vector is resized automatically to size index in case that the index exceeds the current vector size. Object inserted are automatically initialized to a value of None.

This operation is performed in memory only. Use [IntVecobj.Save\(\)](#) to save the modified vector to database.

```
None IntVecobj.Set(int index,  
                   DataObject obj)
```

ARGUMENTS

- index* Index in vector.
- obj* Object to assign in row index.

SEE ALSO

[IntVecobj.Get\(\)](#)

Size

Returns the size of the vector.

```
int IntVecobj.Size()
```

RETURNS

The size of the vector.

5.6.46 IntVersion

Overview

[CreateDerivedProject](#)
[GetDerivedProjects](#)
[GetHistoricalProject](#)
[Rollback](#)

CreateDerivedProject

Creates a derived project from the version.

```
DataObject IntVersion.CreateDerivedProject(str name,
                                           [DataObject parent]
                                           )
```

ARGUMENTS

name The name of the project which will be created.

parent(optional)

The parent of the project which will be created. Default is the current user.

RETURNS

Returns the created project.

GetDerivedProjects

list of projects derived from this version

```
list IntVersion.GetDerivedProjects()
```

RETURNS

list of derived projects

GetHistoricalProject

Returns historic project within version

```
DataObject IntVersion.GetHistoricalProject()
```

RETURNS

Returns the historic project object

Rollback

Roll backs the project to this version. No project have to be active. Furthermore no script from the project of the version have to be running.

```
int IntVersion.Rollback()
```

RETURNS

0	on success
1	otherwise

5.6.47 IntViewbookmark

Overview

[JumpTo](#)
[UpdateFromCurrentView](#)

JumpTo

Opens the referenced diagram (if not already open) and sets the viewing area.

```
None IntViewbookmark.JumpTo()
```

UpdateFromCurrentView

Updates the bookmark's diagram and view area from the current drawing window.

```
None IntViewbookmark.UpdateFromCurrentView()
```

5.6.48 PltDataseries

Overview

[AddCurve](#)
[AddXYCurve](#)
[ClearCurves](#)
[GetDataSource](#)
[GetIntCalcres](#)

AddCurve

Appends a curve to the plot.

```
None PltDataseries.AddCurve(DataObject element,  
                             str varname,  
                             [DataObject datasource]  
                             )
```

ARGUMENTS

element Element to display

varname Name of the element variable to display

datasource (optional)

Data source to assign to the curve (ElmRes or IntComtrade). If not specified, the plot's default result file will be used for the curve.

AddXYCurve

Appends a curve to a XY plot.

```
None PltDataserie.AddXYCurve(DataObject elementX,
                              str varnameX,
                              DataObject elementY,
                              str varnameY,
                              [DataObject datasource]
                              )
```

ARGUMENTS

elementX Element to display on x-axis

varnameX
Name of the element variable to display on x-axis

elementY Element to display on y-axis

varnameY
Name of the element variable to display on y-axis

datasource (optional)
Data source to assign to the curve (ElmRes or IntComtrade). If not specified, the plot's default result file will be used for the curve.

ClearCurves

Removes all curves from the plot.

```
None PltDataserie.ClearCurves()
```

GetDataSource

Returns the data source that is used for the curve with given index.

```
DataObject PltDataserie.GetDataSource(int curveIndex)
```

ARGUMENTS

int curveIndex
curve index in table, must be ≥ 0

RETURNS

DataObject Data source

None No data source found

GetIntCalcres

Gets all user Calculated Result objects (IntCalcres) stored inside the data series.

```
list PltDataserie.GetIntCalcres()
```

RETURNS

All Calculated Result objects (IntCalces) stored inside the data series.

5.6.49 PltLinebarplot

Overview

[DoAutoScale](#)
[GetAxisX](#)
[GetAxisY](#)
[GetDataSet](#)
[GetLegend](#)
[GetTitleObject](#)
[SetAutoScaleModeX](#)
[SetAutoScaleModeY](#)
[SetAxisSharingLevelX](#)
[SetAxisSharingLevelY](#)
[SetScaleTypeX](#)
[SetScaleTypeY](#)
[SetScaleX](#)
[SetScaleY](#)

DoAutoScale

Adapts the plot's axes ranges such that they show the entire data range.

```
None PltLinebarplot.DoAutoScale([int axisDimension])
```

ARGUMENTS

axisDimension (optional)

Limits auto-scaling to one dimension. Possible values:

- | | |
|----------|-------------------|
| 0 | scale only x-axes |
| 1 | scale only y-axes |

GetAxisX

Returns one of the plot's x-axes.

```
DataObject PltLinebarplot.GetAxisX([int axisIndex])
```

ARGUMENTS

axisIndex=0 (optional)

Determines which x-axis should be returned:

- | | |
|----------|--|
| 0 | main x-axis |
| 1 | second x-axis (will be created if not yet present) |

RETURNS

PltAxis object

GetAxisY

Returns one of the plot's y-axes.

```
DataObject PltLinebarplot.GetAxisY([int axisIndex])
```

ARGUMENTS

axisIndex=0 (optional)

Determines which y-axis should be returned:

- | | |
|----------|--|
| 0 | main y-axis |
| 1 | second y-axis (will be created if not yet present) |

RETURNS

PltAxis object

GetDataSeries

Returns the plot's data series object (PltDataseries).

```
DataObject PltLinebarplot.GetDataSeries()
```

RETURNS

Data series object (PltDataseries)

GetLegend

Returns the plot's legend object (PltLegend).

```
DataObject PltLinebarplot.GetLegend()
```

RETURNS

Legend object (PltLegend)

GetTitleObject

Returns the plot's title object (PltTitle).

```
DataObject PltLinebarplot.GetTitleObject()
```

RETURNS

Title object (PltTitle)

SetAutoScaleModeX

Defines whether the plot should automatically adapt its main x-axis range on data changes.

```
None PltLinebarplot.SetAutoScaleModeX(int mode)
```

ARGUMENTS

<i>mode</i>	Possible values:
0	off (do not react to data changes)
1	adapt scale after calculation has finished
2	adapt scale during live plotting

SetAutoScaleModeY

Defines whether the plot should automatically adapt its main y-axis range on data changes.

```
None PltLinebarplot.SetAutoScaleModeY(int mode)
```

ARGUMENTS

<i>mode</i>	Possible values:
0	off (do not react to data changes)
1	adapt scale after calculation has finished
2	adapt scale during live plotting

SetAxisSharingLevelX

Defines whether the plot has its own x-axis or should share its x-axis across the page or the entire graphics board.

```
None PltLinebarplot.SetAxisSharingLevelX(int level)
```

ARGUMENTS

<i>level</i>	Possible values:
0	local (do not share x-axis)
1	use x-axis from page
2	use x-axis from graphics board

SetAxisSharingLevelY

Defines whether the plot has its own y-axis or should share its y-axis across the page or the entire graphics board.

```
None PltLinebarplot.SetAxisSharingLevelY(int level)
```

ARGUMENTS

<i>level</i>	Possible values:
0	local (do not share y-axis)
1	use y-axis from page
2	use y-axis from graphics board

SetScaleTypeX

Sets the scale type of the plot's main x-axis.

```
None PltLinebarplot.SetScaleTypeX(int scaleType)
```

ARGUMENTS

scaleType

Possible values:

0	linear
1	logarithmic

SetScaleTypeY

Set the scale type of the plot's main y-axis.

```
None PltLinebarplot.SetScaleTypeY(int scaleType)
```

ARGUMENTS

scaleType

Possible values:

0	linear
1	logarithmic
2	dB

SetScaleX

Sets the scale of the plot's main x-axis.

```
None PltLinebarplot.SetScaleX(float min,  
                                float max  
                                )
```

ARGUMENTS

min Minimum of x-scale.

max Maximum of x-scale.

SetScaleY

Sets the scale of the plot's main y-axis.

```
None PltLinebarplot.SetScaleY(float min,  
                                float max  
                                )
```


ARGUMENTS

<i>min</i>	Minimum of y-scale.
<i>max</i>	Maximum of y-scale.

5.6.50 RelZpol

Overview

[AssumeCompensationFactor](#)
[AssumeReRl](#)
[AssumeXeXl](#)

AssumeCompensationFactor

Triggers a calculation of the complex compensation factor and stores the result.

```
int RelZpol.AssumeCompensationFactor()
```

RETURNS

- 0** The compensation factor was successfully calculated.
- 1** An error occurred (e.g. conencted branch was not found).

AssumeReRl

Triggers a calculation of the real part of the decoupled compensation factor and stores the result.

```
int RelZpol.AssumeReRl()
```

RETURNS

- 0** The compensation factor was successfully calculated.
- 1** An error occurred (e.g. conencted branch was not found).

AssumeXeXl

Triggers a calculation of the imaginary part of the decoupled compensation factor and stores the result.

```
int RelZpol.AssumeXeXl()
```

RETURNS

- 0** The compensation factor was successfully calculated.
- 1** An error occurred (e.g. conencted branch was not found).

5.6.51 ScnFreq

Overview

[GetLimit](#)
[GetNumberOfViolations](#)
[GetValue](#)
[GetVariable](#)
[GetViolatedElement](#)
[GetViolationTime](#)

GetLimit

Returns the limit value (in p.u.) of the i^{th} violation, given by vldx.

```
float ScnFreq.GetLimit(int vIdx)
```

ARGUMENTS

vldx $vldx > 0$

RETURNS

The limit value (in p.u.) of the i^{th} violation, given by vldx. If vldx is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetNumberOfViolations

Returns the number of violations.

```
int ScnFreq.GetNumberOfViolations()
```

RETURNS

The number of violations.

GetValue

Returns the i^{th} value (in p.u.), given by vldx, which causes the violation.

```
float ScnFreq.GetValue(int vIdx)
```

ARGUMENTS

vldx $vldx > 0$

RETURNS

The i^{th} value (in p.u.), given by vldx, which causes the violation. If vldx is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetVariable

Returns the name of the variable of the i^{th} violation, given by vldx.

```
str ScnFreq.GetVariable(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The name of the variable of the i^{th} violation, given by *vIdx*. If *vIdx* is negative, zero or greater than the number of violations, then the function returns "NoVariable".

GetViolatedElement

Returns the element of the i^{th} violation, given by *vIdx*.

```
DataObject ScnFreq.GetViolatedElement(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The element of the i^{th} violation, given by *vIdx*. If *vIdx* is negative, zero or greater than the number of violations, then the function returns None.

GetViolationTime

Returns the time (in seconds) of the i^{th} violation, given by *vIdx*.

```
float ScnFreq.GetViolationTime(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The time (in seconds) of the i^{th} violation, given by *vIdx*. If *vIdx* is negative, zero or greater than the number of violations, then the function returns 'nan'.

5.6.52 ScnFrt**Overview**

[GetLimit](#)
[GetNumberOfViolations](#)
[GetValue](#)
[GetVariable](#)
[GetViolatedElement](#)
[GetViolationTime](#)

GetLimit

Returns the limit value of the i^{th} violation, given by *vIdx*.

```
float ScnFrt.GetLimit(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The limit value of the i^{th} violation, given by *vIdx*. If *vIdx* is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetNumberOfViolations

Returns the number of violations.

```
int ScnFrt.GetNumberOfViolations()
```

RETURNS

The number of violations.

GetValue

Returns the i^{th} value, given by *vIdx*, which causes the violation.

```
float ScnFrt.GetValue(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The i^{th} value, given by *vIdx*, which causes the violation. If *vIdx* is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetVariable

Returns the name of the variable of the i^{th} violation, given by *vIdx*.

```
str ScnFrt.GetVariable(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The name of the variable of the i^{th} violation, given by *vIdx*. If *vIdx* is negative, zero or greater than the number of violations, then the function returns "NoVariable".

GetViolatedElement

Returns the element of the i^{th} violation, given by *vIdx*.

```
DataObject ScnFrt.GetViolatedElement(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The element of the i^{th} violation, given by vldx. If vldx is negative, zero or greater than the number of violations, then the function returns None.

GetViolationTime

Returns the time (in seconds) of the i^{th} violation, given by vldx.

```
float ScnFrt.GetViolationTime(int vIdx)
```

ARGUMENTS

vldx $vldx > 0$

RETURNS

The time (in seconds) of the i^{th} violation, given by vldx. If vldx is negative, zero or greater than the number of violations, then the function returns 'nan'.

5.6.53 ScnSpeed**Overview**

[GetLimit](#)
[GetNumberOfViolations](#)
[GetValue](#)
[GetVariable](#)
[GetViolatedElement](#)
[GetViolationTime](#)

GetLimit

Returns the limit value (in p.u.) of the i^{th} violation, given by vldx.

```
float ScnSpeed.GetLimit(int vIdx)
```

ARGUMENTS

vldx $vldx > 0$

RETURNS

The limit value (in p.u.) of the i^{th} violation, given by vldx. If vldx is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetNumberOfViolations

Returns the number of violations.

```
int ScnSpeed.GetNumberOfViolations()
```

RETURNS

The number of violations.

GetValue

Returns the i^{th} value (in p.u.), given by vIdx, which causes the violation.

```
float ScnSpeed.GetValue(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The i^{th} value (in p.u.), given by vIdx, which causes the violation. If vIdx is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetVariable

Returns the name of the variable of the i^{th} violation, given by vIdx.

```
str ScnSpeed.GetVariable(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The name of the variable of the i^{th} violation, given by vIdx. If vIdx is negative, zero or greater than the number of violations, then the function returns "NoVariable".

GetViolatedElement

Returns the element of the i^{th} violation, given by vIdx.

```
DataObject ScnSpeed.GetViolatedElement(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The element of the i^{th} violation, given by vIdx. If vIdx is negative, zero or greater than the number of violations, then the function returns None.

GetViolationTime

Returns the time (in seconds) of the i^{th} violation, given by vIdx.

```
float ScnSpeed.GetViolationTime(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The time (in seconds) of the i^{th} violation, given by vIdx. If vIdx is negative, zero or greater than the number of violations, then the function returns 'nan'.

5.6.54 ScnSync

Overview

[GetLimit](#)
[GetNumberOfViolations](#)
[GetValue](#)
[GetVariable](#)
[GetViolatedElement](#)
[GetViolationTime](#)

GetLimit

Returns the limit value of the i^{th} violation, given by vIdx.

```
float ScnSync.GetLimit(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The limit value of the i^{th} violation, given by vIdx. If vIdx is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetNumberOfViolations

Returns the number of violations.

```
int ScnSync.GetNumberOfViolations()
```

RETURNS

The number of violations.

GetValue

Returns the i^{th} value, given by vIdx, which causes the violation.

```
float ScnSync.GetValue(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The i^{th} value, given by vIdx, which causes the violation. If vIdx is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetVariable

Returns the name of the variable of the i^{th} violation, given by vIdx.

```
str ScnSync.GetVariable(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The name of the variable of the i^{th} violation, given by *vIdx*. If *vIdx* is negative, zero or greater than the number of violations, then the function returns "NoVariable".

GetViolatedElement

Returns the element of the i^{th} violation, given by *vIdx*.

```
DataObject ScnSync.GetViolatedElement(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The element of the i^{th} violation, given by *vIdx*. If *vIdx* is negative, zero or greater than the number of violations, then the function returns None.

GetViolationTime

Returns the time (in seconds) of the i^{th} violation, given by *vIdx*.

```
float ScnSync.GetViolationTime(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The time (in seconds) of the i^{th} violation, given by *vIdx*. If *vIdx* is negative, zero or greater than the number of violations, then the function returns 'nan'.

5.6.55 ScnVar**Overview**

[GetLimit](#)
[GetNumberOfViolations](#)
[GetValue](#)
[GetVariable](#)
[GetViolatedElement](#)
[GetViolationTime](#)

GetLimit

Returns the limit value of the i^{th} violation, given by *vIdx*.

```
float ScnVar.GetLimit(int vIdx)
```


ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The limit value of the i^{th} violation, given by *vIdx*. If *vIdx* is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetNumberOfViolations

Returns the number of violations.

```
int ScnVar.GetNumberOfViolations()
```

RETURNS

The number of violations.

GetValue

Returns the i^{th} value, given by *vIdx*, which causes the violation.

```
float ScnVar.GetValue(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The i^{th} value, given by *vIdx*, which causes the violation. If *vIdx* is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetVariable

Returns the name of the variable of the i^{th} violation, given by *vIdx*.

```
str ScnVar.GetVariable(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The name of the variable of the i^{th} violation, given by *vIdx*. If *vIdx* is negative, zero or greater than the number of violations, then the function returns "NoVariable".

GetViolatedElement

Returns the element of the i^{th} violation, given by *vIdx*.

```
DataObject ScnVar.GetViolatedElement(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The element of the i^{th} violation, given by vldx. If vldx is negative, zero or greater than the number of violations, then the function returns None.

GetViolationTime

Returns the time (in seconds) of the i^{th} violation, given by vldx.

```
float ScnVar.GetViolationTime(int vIdx)
```

ARGUMENTS

vldx $vldx > 0$

RETURNS

The time (in seconds) of the i^{th} violation, given by vldx. If vldx is negative, zero or greater than the number of violations, then the function returns 'nan'.

5.6.56 ScnVolt**Overview**

[GetLimit](#)
[GetNumberOfViolations](#)
[GetValue](#)
[GetVariable](#)
[GetViolatedElement](#)
[GetViolationTime](#)

GetLimit

Returns the limit value (in p.u.) of the i^{th} violation, given by vldx.

```
float ScnVolt.GetLimit(int vIdx)
```

ARGUMENTS

vldx $vldx > 0$

RETURNS

The limit value (in p.u.) of the i^{th} violation, given by vldx. If vldx is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetNumberOfViolations

Returns the number of violations.

```
int ScnVolt.GetNumberOfViolations()
```

RETURNS

The number of violations.

GetValue

Returns the i^{th} value (in p.u.), given by vIdx, which causes the violation.

```
float ScnVolt.GetValue(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The i^{th} value (in p.u.), given by vIdx, which causes the violation. If vIdx is negative, zero or greater than the number of violations, then the function returns 'nan'.

GetVariable

Returns the name of the variable of the i^{th} violation, given by vIdx.

```
str ScnVolt.GetVariable(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The name of the variable of the i^{th} violation, given by vIdx. If vIdx is negative, zero or greater than the number of violations, then the function returns "NoVariable".

GetViolatedElement

Returns the element of the i^{th} violation, given by vIdx.

```
DataObject ScnVolt.GetViolatedElement(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The element of the i^{th} violation, given by vIdx. If vIdx is negative, zero or greater than the number of violations, then the function returns None.

GetViolationTime

Returns the time (in seconds) of the i^{th} violation, given by vIdx.

```
float ScnVolt.GetViolationTime(int vIdx)
```

ARGUMENTS

vIdx $vIdx > 0$

RETURNS

The time (in seconds) of the i^{th} violation, given by vIdx. If vIdx is negative, zero or greater than the number of violations, then the function returns 'nan'.

5.6.57 StoMaint

Overview

[SetElms](#)

SetElms

Sets the maintenance elements.

```
None StoMaint.SetElms(DataObject singleElement)
None StoMaint.SetElms(list multipleElements)
```

ARGUMENTS

singleElement
single Element for Maintenance

multipleElements
multiple Elements for Maintenance

5.6.58 TypAsmo

Overview

[CalcElParams](#)

CalcElParams

Function calculates the electrical parameters from the input data. Behaves identically as the calculate button on the basic data page was pressed. Shall be applied only if the 'Slip-Torque/Current Characteristic' chosen.

```
int TypAsmo.CalcElParams()
```

RETURNS

0 Calculated successfully.

1 Error.

5.6.59 TypCtcore

Overview

[AddRatio](#)
[RemoveRatio](#)
[RemoveRatioByIndex](#)

AddRatio

Adds the given ratio to the core. The ratio is added so that the sorting remains in ascending order. The accuracy parameters will be copied from the previous row. If the new ratio is to be the first ratio, the accuracy parameters will be copied from the next row instead.

```
int TypCtcore.AddRatio(float ratio)
```

ARGUMENTS

ratio New ratio to add.

RETURNS

0 New ratio was added.
1 An error occurred.

RemoveRatio

Removes the given ratio from the core. The last remaining ratio can never be removed.

```
int TypCtcore.RemoveRatio(float ratio)
```

ARGUMENTS

ratio Ratio to remove.

RETURNS

0 Ratio was removed.
1 An error occurred.

RemoveRatioByIndex

Removes the ratio with the given index from the core. The index is zero based. The last remaining ratio can never be removed.

```
int TypCtcore.RemoveRatio(int index)
```

ARGUMENTS

index Index to remove.

RETURNS

0 Index was removed.
1 An error occurred.

5.6.60 TypLne**Overview**

[IsCable](#)

IsCable

Checks if the line type is a cable type.

```
int TypLne.IsCable()
```

RETURNS

1 Type is a cable
0 Type is not a cable

5.6.61 TypQdsl

Overview

[Encrypt](#)
[IsEncrypted](#)
[ResetThirdPartyModule](#)
[SetThirdPartyModule](#)

Encrypt

Encrypts a type. An encrypted type can be used without password but decrypted only with password. If no password is given a 'Choose Password' dialog appears.

```
int TypQdsl.Encrypt([str password = ""],
                   [int removeObjectHistory = 1],
                   [int masterCode = 0])
```

ARGUMENTS

password (optional)

Password for decryption. If no password is given a 'Choose Password' dialog appears.

removeObjectHistory (optional)

Handling of unencrypted object history in database, e.g. used by project versions or by undo:

- 0** Do not remove.
- 1** Do remove (default).
- 2** Show dialog and ask.

masterCode (optional)

Used for re-selling types. Third party licence codes already set in the type will be overwritten by this value (default = 0).

RETURNS

- 0** On success.
- 1** On error.

SEE ALSO

[TypQdsl.IsEncrypted\(\)](#)

IsEncrypted

Returns the encryption state of the type.

```
int TypQdsl.IsEncrypted()
```

RETURNS

- 1** Type is encrypted.
- 0** Type is not encrypted.

SEE ALSO

[TypQdsl.Encrypt\(\)](#)

ResetThirdPartyModule

Resets the third party licence. Only possible for non-encrypted models. Requires masterkey licence for third party module currently set.

```
int TypQdsl.ResetThirdPartyModule()
```

RETURNS

0 On success.
1 On error.

SetThirdPartyModule

Sets the third party licence to a specific value. Only possible for non-encrypted models with no third party licence set so far. Requires masterkey licence for third party module to be set.

```
int TypQdsl.SetThirdPartyModule(str companyName,
                                str moduleCode
                                )
```

ARGUMENTS

companyCode

D isplay name or numeric value of company code.

moduleCode

D isplay name or numeric value of third party module.

RETURNS

0 On success.
1 On error.

5.6.62 TypTr2

Overview

[GetZeroSequenceHVLVT](#)

GetZeroSequenceHVLVT

Returns the calculated star equivalent of the zero sequence impedances.

```
[int error,
float hvReal,
float hvImag,
float lvReal,
float lvImag,
float tReal ,
float tImag ] TypTr2.GetZeroSequenceHVLVT()
```

ARGUMENTS

hvReal (out)

Real part of the HV impedance in %.

hvlmag (out)

Imaginary part of the HV impedance in %.

lvReal (out)

Real part of the LV impedance in %.

lvlmag (out)

Imaginary part of the LV impedance in %.

tReal (out)

Real part of the tertiary (delta) impedance in %.

tlmag (out)

Imaginary part of the tertiary (delta) impedance in %.

RETURNS

0 No error occurred.**1** An error occurred; the values are invalid.

5.6.63 VisBdia

Overview

[AddObjs](#)
[AddResObjs](#)
[Clear](#)
[SetScaleY](#)
[SetXVariable](#)
[SetYVariable](#)

AddObjs

Adds objects to elements column in table 'Bars'.

```
None VisBdia.AddObjs(list elements)
```

ARGUMENTS

elements Elements to add in table.

AddResObjs

Adds objects to elements column in table 'Bars' (similar to AddObjs). Additionally a result file is assigned to all rows added in the 'Result File' column.

```
None VisBdia.AddResObjs(DataObject resultFileObj,
                        list elements
                        )
```


ARGUMENTS

resultFileObj

The result file to assign. Must be an object of class ElmRes.

elements Elements to add in table.**Clear**

Removes all elements from plot by erasing all rows from the table named 'Bars'.

`None VisBdia.Clear()`**SetScaleY**

Sets y-axis scale limits.

```

None VisBdia.SetScaleY(float min,
                        float max,
                        [int log]
                        )

```

ARGUMENTS

min (optional)

Minimum of y-scale.

max (optional)

Maximum of y-scale.

log (optional)

Possible values:

0	linear
1	logarithmic

SetXVariable

Set the x-axis Variable of the Distortion Analysis Diagram

`int VisBdia.SetXVariable(str variable)`

ARGUMENTS

variable x-axis variable to set.Length of variable must not exceed 37 characters.

RETURNS

0 if ok, 1 if variable length exceeds 37 characters,

SetYVariable

Set the y-axis variable of the Distortion Analysis Diagram

`int VisBdia.SetYVariable(str variable)`

ARGUMENTS

variable y-axis variable to set.Length of variable must not exceed 37 characters.

RETURNS

0 if ok, 1 if variable length exceeds 37 characters,

5.6.64 VisDraw

Overview

[AddRelay](#)
[AddRelays](#)
[CentreOrigin](#)
[Clear](#)
[DoAutoScaleOnAll](#)
[DoAutoScaleOnCharacteristics](#)
[DoAutoScaleOnImpedances](#)
[DoAutoScaleX](#)
[DoAutoScaleY](#)

AddRelay

Adds a relay to the plot and optionally sets the drawing style.

```
None VisDraw.AddRelay(DataObject relay,  
                        [float colour,]  
                        [float style,]  
                        [float width])
```

ARGUMENTS

relay The protection device to be added.

colour (optional)
The colour to be used.

style (optional)
The line style to be used.

width (optional)
The line width to be used.

AddRelays

Adds relays to the plot.

```
None VisDraw.AddRelays(list relays)
```

ARGUMENTS

relays The protection devices (ElmRelay or RelFuse) to be added.

CentreOrigin

Centre the origin of the plot

```
None VisDraw.CentreOrigin()
```

Clear

Removes all protection devices from the plot.

```
None VisDraw.Clear()
```

DoAutoScaleOnAll

Scales the plot automatically under consideration of relay characteristics, simulation curves and short circuit arrows. The function works for local scales only.

```
int VisDraw.DoAutoScaleOnAll()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

DoAutoScaleOnCharacteristics

Scales the plot automatically under consideration of relay characteristics. Same as button named Characteristics. The function works for local scales only.

```
int VisDraw.DoAutoScaleOnCharacteristics()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

DoAutoScaleOnImpedances

Scales the plot automatically under consideration of branch impedances. Same as button named Impedances. The function works for local scales only.

```
int VisDraw.DoAutoScaleOnImpedances()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

DoAutoScaleX

Scales the x-axis of the plot automatically. The function works for local x-scales only. If the x-scale is not local a warning is shown in the output window and 1 is returned by the function.

```
int VisDraw.DoAutoScaleX()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

DoAutoScaleY

Scales the y-axis of the plot automatically. The function works for local y-scales only. If the x-scale is not local a warning is shown in the output window and 1 is returned by the function.

```
int VisDraw.DoAutoScaleY()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

5.6.65 VisHrm**Overview**

- [Clear](#)
- [DoAutoScaleX](#)
- [DoAutoScaleY](#)
- [GetDataSource](#)
- [GetScaleObjX](#)
- [GetScaleObjY](#)
- [SetAutoScaleX](#)
- [SetAutoScaleY](#)
- [SetCrvDesc](#)
- [SetDefScaleX](#)
- [SetDefScaleY](#)

Clear

Removes all curves by clearing table named 'Curves'.

```
None VisHrm.Clear()
```

DoAutoScaleX

Scales x-axis automatically.

```
int VisHrm.DoAutoScaleX()
```

RETURNS

- 0** Ok, call to DoAutoScaleX() was successfull
- 1** Failed, because the x-scale is not local

DoAutoScaleY

Scales y-axis automatically.

```
int VisHrm.DoAutoScaleY()
```

RETURNS

- 0** Ok, call to DoAutoScaleY() was successfull
- 1** Failed, because the y-scale is not local

GetDataSource

Get data source for curve with index

```
DataObject VisHrm.GetDataSource(int curveIndex)
```

ARGUMENTS

int curveIndex

Possible values:

- < 0 invalid
- ≥ 0 curve index in table

RETURNS

- DataObject** Data source
- None** No data source found

GetScaleObjX

Gets the object used for scaling the x-axis.

```
DataObject VisHrm.GetScaleObjX()
```

RETURNS

- this object** In case that 'Use local Axis' is set to 'Local'.
- the virtual instrument panel** In case that 'Use local axis' is set to 'Current Page'.
- the graphics board** In case that 'Use local axis' is set to 'Graphics Board'.

GetScaleObjY

Gets the object used for scaling the y-axis.

```
DataObject VisHrm.GetScaleObjY()
```

RETURNS

- this object** In case that 'Use local Axis' is enabled.
- the plot type** In case that 'Use local axis' is disabled.

SetAutoScaleX

Sets Auto Scale setting of the x-scale. The scale is automatic set to local, in case that the waveform plot is using the scale of the graphics board or the virtual instrument panel.

```
None VisHrm.SetAutoScaleX(int mode)
```

ARGUMENTS

<i>mode</i>	Possible values:
0	never
1	after simulation

SetAutoScaleY

Sets Auto Scale setting of the y-scale. The scale is automatic set to local, in case that the waveform plot is using the scale of the plot type.

```
None VisHrm.SetAutoScaleY(int mode)
```

ARGUMENTS

<i>mode</i>	Possible values:
0	never
1	after simulation

SetCrvDesc

Sets the user defined description of a curve.

```
None VisHrm.SetCrvDesc(int curveIndex, str curveDescription)
```

ARGUMENTS

<i>curveIndex</i>	Curve index; first curve in table is index 1.
<i>curveDescription</i>	Description to set

SetDefScaleX

Sets the x-scale to be used to the graphics board.

```
None VisHrm.SetDefScaleX()
```

SetDefScaleY

Sets the y-scale to be used to the plot type.

```
None VisHrm.SetDefScaleY()
```

5.6.66 VisMagndiffplt

Overview

[AddRelay](#)
[AddRelays](#)
[Clear](#)
[DoAutoScaleX](#)
[DoAutoScaleY](#)
[Refresh](#)

AddRelay

Adds a relay to the plot and optionally sets the drawing style.

```
None VisMagndiffplt.AddRelay(DataObject relay,  
                             [float colour,]  
                             [float style,]  
                             [float width])
```

ARGUMENTS

relay The protection device to be added.

colour (optional)
 The colour to be used.

style (optional)
 The line style to be used.

width (optional)
 The line width to be used.

AddRelays

Adds relays to the plot.

```
None VisMagndiffplt.AddRelays(list relays)
```

ARGUMENTS

relays The protection devices (ElmRelay or RelFuse) to be added.

Clear

Removes all protection devices from the plot.

```
None VisMagndiffplt.Clear()
```

DoAutoScaleX

Scales the x-axis of the plot automatically. The function works for local x-scales only. If the x-scale is not local a warning is shown in the output window and 1 is returned by the function.

```
int VisMagndiffplt.DoAutoScaleX()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

DoAutoScaleY

Scales the y-axis of the plot automatically. The function works for local y-scales only. If the x-scale is not local a warning is shown in the output window and 1 is returned by the function.

```
int VisMagndiffplt.DoAutoScaleY()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

Refresh

Refreshes the plot by attempting to automatically scale both axes.

```
None VisMagndiffplt.Refresh()
```

5.6.67 VisOcplot**Overview**

[AddRelay](#)
[AddRelays](#)
[Clear](#)
[DoAutoScaleX](#)
[DoAutoScaleY](#)
[Refresh](#)

AddRelay

Adds a relay to the plot and optionally sets the drawing style.

```
None VisOcplot.AddRelay(DataObject relay,
                        [float colour,]
                        [float style,]
                        [float width])
```

ARGUMENTS

- relay* The protection device to be added.
- colour (optional)*
The colour to be used.
- style (optional)*
The line style to be used.
- width (optional)*
The line width to be used.

AddRelays

Adds relays to the plot.

```
None VisOcplot.AddRelays(list relays)
```

ARGUMENTS

relays The protection devices (ElmRelay or RelFuse) to be added.

Clear

Removes all protection devices from the plot.

```
None VisOcplot.Clear()
```

DoAutoScaleX

Scales the x-axis of the plot automatically. The function works for local x-scales only. If the x-scale is not local a warning is shown in the output window and 1 is returned by the function.

```
int VisOcplot.DoAutoScaleX()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

DoAutoScaleY

Scales the y-axis of the plot automatically. The function works for local y-scales only. If the x-scale is not local a warning is shown in the output window and 1 is returned by the function.

```
int VisOcplot.DoAutoScaleY()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

Refresh

Refreshes the plot by attempting to automatically scale both axes.

```
None VisOcplot.Refresh()
```

5.6.68 VisPath

Overview

[Clear](#)
[DoAutoScaleX](#)
[DoAutoScaleY](#)
[SetAdaptX](#)
[SetAdaptY](#)
[SetScaleX](#)
[SetScaleY](#)

Clear

Removes all curves by clearing table named 'Variables' on page 'Curves'.

```
None VisPath.Clear()
```

DoAutoScaleX

Scales x-axis automatically.

```
int VisPath.DoAutoScaleX()
```

RETURNS

Always 0

DoAutoScaleY

Scales y-axis automatically.

```
int VisPath.DoAutoScaleY()
```

RETURNS

Always 0

SetAdaptX

Sets the Adapt Scale option of the x-scale.

```
None VisPath.SetAdaptX(int mode)
```

ARGUMENTS

<i>mode</i>	Possible values:
0	off
1	on

SetAdaptY

Sets the Adapt Scale option of the x-scale.

```
None VisPath.SetAdaptY(int mode)
```

ARGUMENTS

<i>mode</i>	Possible values:
0	off
1	on

SetScaleX

Sets x-axis scale.

```
None VisPath.SetScaleX(float min,
                        float max
                        )
```

ARGUMENTS

<i>min</i>	Minimum of x-scale.
<i>max</i>	Maximum of x-scale.

SetScaleY

Sets y-axis scale limits.

```
None VisPath.SetScaleY(float min,
                        float max,
                        [int log]
                        )
```

ARGUMENTS

<i>min</i>	Minimum of y-scale.
<i>max</i>	Maximum of y-scale.
<i>log (optional)</i>	Possible values:
0	linear
1	logarithmic

5.6.69 VisPcompdiffplt**Overview**

[AddRelay](#)
[AddRelays](#)
[CentreOrigin](#)
[Clear](#)
[DoAutoScaleX](#)
[DoAutoScaleY](#)

AddRelay

Adds a relay to the plot and optionally sets the drawing style.

```
None VisPcompdiffplt.AddRelay(DataObject relay,  
                               [float colour,]  
                               [float style,]  
                               [float width])
```

ARGUMENTS

relay The protection device to be added.

colour (optional)
 The colour to be used.

style (optional)
 The line style to be used.

width (optional)
 The line width to be used.

AddRelays

Adds relays to the plot.

```
None VisPcompdiffplt.AddRelays(list relays)
```

ARGUMENTS

relays The protection devices (ElmRelay or RelFuse) to be added.

CentreOrigin

Centre the origin of the plot

```
None VisPcompdiffplt.CentreOrigin()
```

Clear

Removes all protection devices from the plot.

```
None VisPcompdiffplt.Clear()
```

DoAutoScaleX

Scales the x-axis of the plot automatically. The function works for local x-scales only. If the x-scale is not local a warning is shown in the output window and 1 is returned by the function.

```
int VisPcompdiffplt.DoAutoScaleX()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

DoAutoScaleY

Scales the y-axis of the plot automatically. The function works for local y-scales only. If the x-scale is not local a warning is shown in the output window and 1 is returned by the function.

```
int VisPcompdiffplt.DoAutoScaleY()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

5.6.70 VisPlot

Overview

[AddResVars](#)
[AddVars](#)
[Clear](#)
[DoAutoScaleX](#)
[DoAutoScaleY](#)
[GetDataSource](#)
[GetIntCalcres](#)
[GetScaleObjX](#)
[GetScaleObjY](#)
[SetAdaptX](#)
[SetAdaptY](#)
[SetAutoScaleX](#)
[SetAutoScaleY](#)
[SetCrvDesc](#)
[SetDefScaleX](#)
[SetDefScaleY](#)
[SetScaleX](#)
[SetScaleY](#)
[SetXVar](#)

AddResVars

Appends variables to the plot. Variables which are already in the plot are not added.

```
None VisPlot.AddResVars(DataObject elmRes,  
                        DataObject element,  
                        str varname  
                        )
```

ARGUMENTS

- elmRes* Result object, classname ElmRes.
- element* Element to add.
- varname* Variable name.

AddVars

Appends variables to the plot. Variables which are already in the plot are not added.

```
None VisPlot.AddVars(DataObject element,  
                     str varname  
                     )
```

ARGUMENTS

element Element to add.

varname Variable name.

Clear

Removes all curves from plot.

```
None VisPlot.Clear()
```

DoAutoScaleX

Scales x-axis automatically.

```
int VisPlot.DoAutoScaleX()
```

RETURNS

- 0** Ok, call to DoAutoScaleX() was successfull
- 1** Failed, because the x-scale is not local

DoAutoScaleY

Scales y-axis automatically.

```
int VisPlot.DoAutoScaleY()
```

RETURNS

- 0** Ok, call to DoAutoScaleY() was successfull
- 1** Failed, because the y-scale is not local

GetDataSource

Get data source for curve with index

```
DataObject VisPlot.GetDataSource(int curveIndex)
```

ARGUMENTS

int curveIndex

Possible values:

- < 0 invalid
- ≥ 0 curve index in table

RETURNS

DataObject Data source
None No data source found

GetIntCalcres

Gets all user Calculated Result objects (IntCalcres) stored inside plot

```
list VisPlot.GetIntCalcres()
```

RETURNS

All Calculated Result objects (IntCalcres) stored inside plot.

GetScaleObjX

Gets the object used for scaling the x-axis.

```
DataObject VisPlot.GetScaleObjX()
```

RETURNS

this object In case that 'Use local Axis' is set to 'Local'.
the virtual instrument panel In case that 'Use local axis' is set to 'Current Page'.
the graphics board In case that 'Use local axis' is set to 'Graphics Board'.

GetScaleObjY

Gets the object used for scaling the y-axis.

```
DataObject VisPlot.GetScaleObjY()
```

RETURNS

this object In case that 'Use local Axis' is enabled.
the plot type In case that 'Use local axis' is disabled.

SetAdaptX

Sets the Adapt Scale option of the local x-scale.

```
None VisPlot.SetAdaptX(int mode,  
                        [float trigger]  
                        )
```

ARGUMENTS

mode Possible values:

0	off
1	on

trigger (optional)

Trigger value, unused if mode is off or empty

SetAdaptY

Sets the Adapt Scale option of the local y-scale.

```

None VisPlot.SetAdaptY(int mode,
                        [float offset]
                        )

```

ARGUMENTS

mode Possible values:

0	off
1	on

offset (optional)

Offset value, unused if mode is off or empty

SetAutoScaleX

Sets Auto Scale setting of the x-scale. The scale is automatic set to local, in case that the plot is using the scale of the graphics board or the virtual instrument panel.

```

None VisPlot.SetAutoScaleX(int mode)

```

ARGUMENTS

mode Possible values:

0	never
1	after simulation
2	during simulation

SetAutoScaleY

Sets Auto Scale setting of the y-scale. The scale is automatic set to local, in case that the plot is using the scale of the plot type.

```

None VisPlot.SetAutoScaleY(int mode)

```

ARGUMENTS

mode Possible values:

0	never
1	after simulation
2	during simulation

SetCrvDesc

Sets the user defined description of a curve.

```

None VisPlot.SetCrvDesc(int curveIndex,
                        str curveDescription
                        )

```


ARGUMENTS

curveIndex

Curve index; first curve in table is index 1.

curveDescription

Description to set.

SetDefScaleX

Sets the x-scale to be used to the graphics board.

`None VisPlot.SetDefScaleX()`**SetDefScaleY**

Sets the y-scale to be used to the plot type.

`None VisPlot.SetDefScaleY()`**SetScaleX**

Sets the local x-axis scale. A function call without arguments sets the Auto Scale setting to On without changing the scale itself.

```

None VisPlot.SetScaleX()
None VisPlot.SetScaleX(float min,
                        float max,
                        [int log]
                        )

```

ARGUMENTS

min (optional)

Minimum of x-scale.

max (optional)

Maximum of x-scale.

log (optional)

Possible values:

0	linear
1	logarithmic

SetScaleY

Sets the local y-axis scale. A function call without arguments sets the Auto Scale setting to On without changing the scale itself.

```

None VisPlot.SetScaleY()
None VisPlot.SetScaleY(float min,
                        float max,
                        [int log]
                        )

```

ARGUMENTS

min (optional)

Minimum of y-scale.

max (optional)

Maximum of y-scale.

log (optional)

Possible values:

0 linear**1** logarithmic**SetXVar**

Sets the local x-axis variable. If The default x-axis variable (time) is set if no argument is passed.

```
None VisPlot.SetXVar()
None VisPlot.SetXVar(DataObject obj, [
                        str varname
                        ])
```

ARGUMENTS

obj (optional)

x-axis object

varname (optional)

variable of obj

5.6.71 VisPlot2**Overview**

[AddResVars](#)
[AddVars](#)
[Clear](#)
[DoAutoScaleX](#)
[DoAutoScaleY](#)
[DoAutoScaleY2](#)
[GetDataSource](#)
[GetScaleObjX](#)
[GetScaleObjY](#)
[SetAdaptX](#)
[SetAdaptY](#)
[SetAutoScaleX](#)
[SetAutoScaleY](#)
[SetCrvDesc](#)
[SetDefScaleX](#)
[SetDefScaleY](#)
[SetScaleX](#)
[SetScaleY](#)
[SetXVar](#)
[ShowY2](#)

AddResVars

Appends variables to the plot. Variables which are already in the plot are not added.

```
None VisPlot2.AddResVars(DataObject elmRes
                        DataObject element,
                        str varname,
                        [int y2]
                        )
```

ARGUMENTS

elmRes Result object, classname ElmRes

element Element to add

varname Variable name

y2 (optional)

Possible values:

- 1** y1-axis, default value
- 2** y2 axis

AddVars

Appends variables to the plot. Variables which are already in the plot are not added.

```
None VisPlot2.AddVars(DataObject element,
                      str varname,
                      [int y2]
                      )
```

ARGUMENTS

element Element to add

varname Variable name

y2 (optional)

Possible values:

- 1** y1-axis, default value
- 2** y2 axis

Clear

Removes variables from plot

```
None VisPlot2.Clear([int y2])
```

ARGUMENTS

y2 (optional)

Possible values:

- 1** y1-axis, default value
- 2** y2 axis

DoAutoScaleX

Scales x-axis automatically.

```
int VisPlot2.DoAutoScaleX()
```

RETURNS

- 0** Ok, call to DoAutoScaleX() was successfull
- 1** Failed, because the x-scale is not local

DoAutoScaleY

Scales y1-axis automatically.

```
int VisPlot2.DoAutoScaleY()
```

RETURNS

- 0** Ok, call to DoAutoScaleY() was successfull
- 1** Failed, because the y-scale is not local

DoAutoScaleY2

Scales y2-axis automatically.

```
int VisPlot2.DoAutoScaleY2()
```

RETURNS

- 0** Ok, call to DoAutoScaleY() was successfull
- 1** Failed, because the y-scale is not local

GetDataSource

Get data source for curve with index

```
DataObject VisPlot2.GetDataSource(int curveIndex, int yaxis)
```

ARGUMENTS

int curveIndex

Possible values:

- < 0** invalid
- ≥ 0** curve index in table

int yaxis Possible values:

- 1** y1 axis
- 2** y2 axis

RETURNS

- DataObject** Data source
- None** No data source found

GetScaleObjX

Gets the object used for scaling the x-axis.

```
DataObject VisPlot2.GetScaleObjX()
```

RETURNS

this object In case that 'Use local Axis' is set to 'Local'.

the virtual instrument panel In case that 'Use local axis' is set to 'Current Page'.

the graphics board In case that 'Use local axis' is set to 'Graphics Board'.

GetScaleObjY

Returns used object defining y-scale. The returned object is either the plot itself or the plot type (IntPlot).

```
DataObject VisPlot2.GetScaleObjY ([int y2])
```

RETURNS

this object In case that 'Use local Axis' is enabled.

the plot type In case that 'Use local axis' is disabled.

SetAdaptX

Sets the Adapt Scale option of the local x-scale.

```
None VisPlot2.SetAdaptX(int mode,
                        [float trigger]
                        )
```

ARGUMENTS

mode Possible values:

0 off

1 on

trigger (optional)

Trigger value, unused if mode is off or empty

SetAdaptY

Sets the Adapt Scale option of the local y-scale.

```
None VisPlot2.SetAdaptY(int mode,
                        [float offset,]
                        [int y2]
                        )
```

ARGUMENTS

mode Possible values:

0 off

1 on

offset (optional)

Offset value, unused if mode is off or empty

y2 (optional)

Possible values:

- 1** y1-axis, default value
- 2** y2 axis

SetAutoScaleX

Sets Auto Scale setting of the x-scale. The scale is automatic set to local, in case that the plot is using the scale of the graphics board or the virtual instrument panel.

```
None VisPlot2.SetAutoScaleX(int mode)
```

ARGUMENTS

mode Possible values:

- 0** never
- 1** after simulation
- 2** during simulation

SetAutoScaleY

Sets automatic scaling mode of the y-scale. The axis given in the second argument is automatically set to local.

```
None VisPlot2.SetAutoScaleY (int mode,
                             [int y2]
                             )
```

ARGUMENTS

mode Possible values:

- 0** never
- 1** after simulation
- 2** during simulation

y2 (optional)

Possible values:

- 1** y1-axis, default value
- 2** y2 axis

SetCrvDesc

Sets the user defined description of a curve.

```
None VisPlot2.SetCrvDesc(int curveIndex, str curveDescription)
```

ARGUMENTS

curveIndex

Curve index; first curve in table is index 1.

curveDescription

Description to set

SetDefScaleX

Sets the x-scale to be used to the graphics board.

`None VisPlot2.SetDefScaleX()`**SetDefScaleY**

Sets the y-scale to be used to the plot type.

`None VisPlot2.SetDefScaleY([int y2])`

ARGUMENTS

y2 (optional)

Possible values:

- 1** y1-axis, default value
- 2** y2 axis

SetScaleX

Sets the local x-axis scale. A function call without arguments sets the Auto Scale setting to On without changing the scale itself.

```

None VisPlot.SetScaleX()
None VisPlot.SetScaleX(float min,
                        float max,
                        [int log]
                        )

```

ARGUMENTS

min (optional)

Minimum of x-scale.

max (optional)

Maximum of x-scale.

log (optional)

Possible values:

- 0** linear
- 1** logarithmic

SetScaleY

Sets scale of y-axis. Calling the function without any argument sets the Auto Scale option for the y axis (both share the same setting) to On.

```
None VisPlot2.SetScaleY()
None VisPlot2.SetScaleY(float min,
                        float max,
                        [int log,]
                        [int Y2]
                        )
```

ARGUMENTS

min (optional)

Minimum of y-scale.

max (optional)

Maximum of y-scale.

log (optional)

Possible values:

0	linear
1	logarithmic

y2 (optional)

Possible values:

1	y1-axis, default value
2	y2 axis

SetXVar

Sets the local x-axis variable. If The default x-axis variable (time) is set if no argument is passed.

```
None VisPlot.SetXVar()
None VisPlot.SetXVar(DataObject obj,]
                    str varname
                    )
```

ARGUMENTS

obj (optional)

x-axis object

varname (optional)

variable of obj

ShowY2

Enables or disables the y2 axis.

```
None VisPlot2.ShowY2([int show])
```


ARGUMENTS

show (optional)

Possible values:

- | | |
|----------|------------------------|
| 0 | hide y2 axis |
| 1 | show y2 axis (default) |

5.6.72 VisPlottz

Overview

[AddRelay](#)[AddRelays](#)[Clear](#)[DoAutoScaleX](#)[DoAutoScaleY](#)

AddRelay

Adds a relay to the plot and optionally sets the drawing style.

```
None VisPlottz.AddRelay(DataObject relay,
                        [float colour,]
                        [float style,]
                        [float width])
```

ARGUMENTS

relay The protection device to be added.*colour (optional)*

The colour to be used.

style (optional)

The line style to be used.

width (optional)

The line width to be used.

AddRelays

Adds relays to the plot.

```
None VisPlottz.AddRelays(list relays)
```

ARGUMENTS

relays The protection devices (ElmRelay or RelFuse) to be added.

Clear

Removes all protection devices from the plot.

```
None VisPlottz.Clear()
```

DoAutoScaleX

Scales the x-axis of the plot automatically. The function works for local x-scales only. If the x-scale is not local a warning is shown in the output window and 1 is returned by the function.

```
int VisPlottz.DoAutoScaleX()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

DoAutoScaleY

Scales the y-axis of the plot automatically. The function works for local y-scales only. If the x-scale is not local a warning is shown in the output window and 1 is returned by the function.

```
int VisPlottz.DoAutoScaleY()
```

RETURNS

- 0** Automatic scaling was executed.
- 1** An Error occurred.

5.6.73 VisVec**Overview**

[CentreOrigin](#)

CentreOrigin

Centre the origin of the plot

```
None VisVec.CentreOrigin()
```

5.6.74 VisVecres**Overview**

[CentreOrigin](#)

CentreOrigin

Centre the origin of the plot

```
None VisVecres.CentreOrigin()
```

5.6.75 VisXyplot

Overview

[Clear](#)
[DoAutoScaleX](#)
[DoAutoScaleY](#)
[GetDataSource](#)
[SetCrvDescX](#)
[SetCrvDescY](#)

Clear

Removes all curves from plot.

```
None VisXyplot.Clear()
```

DoAutoScaleX

Scales all used x-axes automatically.

```
int VisXyplot.DoAutoScaleX()
```

RETURNS

- 0** Ok, call to DoAutoScaleX() was successfull
- 1** Failed, because the x-scales are not local

DoAutoScaleY

Scales all used y-axes automatically.

```
int VisXyplot.DoAutoScaleY()
```

RETURNS

- 0** Ok, call to DoAutoScaleX() was successfull
- 1** Failed, because the x-scales are not local

GetDataSource

Get data source for curve with index

```
DataObject VisPlot.GetDataSource(int curveIndex)
```

ARGUMENTS

int curveIndex

Possible values:

- < 0** invalid
- ≥ 0** curve index in table

RETURNS

DataObject Data source

None No data source found

SetCrvDescX

Sets the user defined description of a curve for the x-variable.

```
None VisXyplot.SetCrvDescX(int curveIndex, str curveDescription)
```

ARGUMENTS

curveIndex

Curve index; first curve in table is index 1.

curveDescription

Description to set

SetCrvDescY

Sets the user defined description of a curve for the y-variable.

```
None VisXyplot.SetCrvDescY(int curveIndex, str curveDescription)
```

ARGUMENTS

curveIndex

Curve index; first curve in table is index 1.

curveDescription

Description to set

Index

__version__
Module Functions, 2

Activate

ElmNet, 105
IntCase, 383
IntLibrary, 410
IntPrj, 424
IntScenario, 439
IntScensched, 442
IntScheme, 444
IntSscheduler, 445
IntSstage, 446

ActivateProject

Application Methods, 5

AdaptWidth

SetLevelvis, 347

AddCntcy

ComSimoutage, 311

AddConfiguration

SetDataext, 337

AddContingencies

ComSimoutage, 312

AddCopy

General Object Methods, 49

AddCubicle

ElmBoundary, 84

AddCurve

PltDataseries, 459

AddDouble

IntAddonvars, 379

AddDoubleMatrix

IntAddonvars, 379

AddDoubleVector

IntAddonvars, 380

AddEvent

IntRas, 437

AddInteger

IntAddonvars, 381

AddIntegerVector

IntAddonvars, 381

AddObject

IntAddonvars, 381

AddObjectVector

IntAddonvars, 382

AddObjs

VisBdia, 480

AddPage

SetDesktop, 340

AddProjectToCombined

IntPrj, 424

AddProjectToRemoteDatabase

IntPrj, 424

AddRas

ComSimoutage, 312

AddRatio

TypCtcore, 476

AddRef

ComNmink, 290

IntDataset, 397

SetSelect, 352

AddRelay

VisDraw, 482

VisMagndiffplt, 487

VisOcplot, 488

VisPcompdiffplt, 492

VisPlottz, 505

AddRelays

VisDraw, 482

VisMagndiffplt, 487

VisOcplot, 489

VisPcompdiffplt, 492

VisPlottz, 505

AddResObjs

VisBdia, 480

AddResVars

VisPlot, 493

VisPlot2, 499

AddString

IntAddonvars, 382

AddToUpdatePages

ComProtgraphic, 296

AddTrigger

IntGate, 405

IntRas, 437

AddVar

IntMon, 418

AddVariable

ElmRes, 115

AddVars

- ElmRes, 115
 - IntMon, 418
 - VisPlot, 493
 - VisPlot2, 499
- AddXYCurve
 - PltDataseries, 460
- Align
 - SetLevelvis, 347
- All
 - IntDataset, 397
 - SetSelect, 353
- AllAsm
 - SetSelect, 353
- AllBars
 - SetSelect, 353
- AllBreakers
 - SetPath, 350
 - SetSelect, 353
- AllClosedBreakers
 - SetPath, 350
 - SetSelect, 353
- AllElm
 - SetSelect, 354
- AllLines
 - SetSelect, 354
- AllLoads
 - SetSelect, 354
- AllOpenBreakers
 - SetPath, 350
 - SetSelect, 354
- AllProtectionDevices
 - SetPath, 351
- AllSym
 - SetSelect, 354
- AllTypLne
 - SetSelect, 354
- AnalyseElmRes
 - ComRel3, 300
- AppendCommand
 - ComTasks, 318
- AppendStudyCase
 - ComTasks, 318
- Application Methods, 4
 - ActivateProject, 5
 - ClearRecycleBin, 6
 - CommitTransaction, 6
 - CreateFaultCase, 6
 - CreateProject, 7
 - DefineTransferAttributes, 7
 - DeleteUntouchedObjects, 7
 - ExecuteCmd, 8
 - GetActiveCalculationStr, 8
 - GetActiveNetworkVariations, 9
 - GetActiveProject, 9
 - GetActiveScenario, 9
 - GetActiveScenarioScheduler, 9
 - GetActiveStages, 9
 - GetActiveStudyCase, 10
 - GetAllUsers, 10
 - GetAttributeDescription, 10
 - GetAttributeUnit, 11
 - GetBorderCubicles, 11
 - GetBrowserSelection, 11
 - GetCalcRelevantObjects, 12
 - GetClassDescription, 12
 - GetClassId, 13
 - GetCurrentDiagram, 13
 - GetCurrentScript, 13
 - GetCurrentSelection, 13
 - GetCurrentUser, 14
 - GetCurrentZoomScaleLevel, 14
 - GetDataFolder, 14
 - GetDiagramSelection, 15
 - GetFlowOrientation, 15
 - GetFromStudyCase, 15
 - GetGlobalLibrary, 16
 - GetGraphicsBoard, 16
 - GetInterfaceVersion, 16
 - GetLanguage, 16
 - GetLocalLibrary, 17
 - GetMem, 17
 - GetProjectFolder, 17
 - GetRecordingStage, 18
 - GetSettings, 18
 - GetSummaryGrid, 18
 - GetUserManager, 18
 - Hide, 19
 - ImportDz, 19
 - ImportSnapshot, 19
 - IsAttributeModelInternal, 20
 - IsLdfValid, 20
 - IsRmsValid, 20
 - IsScenarioAttribute, 20
 - IsShcValid, 21
 - IsSimValid, 21
 - IsWriteCacheEnabled, 21
 - LoadProfile, 21
 - MarkInGraphics, 22
 - OutputFlexibleData, 22
 - PostCommand, 22
 - PrepForUntouchedDelete, 23
 - Rebuild, 23
 - ReloadProfile, 23
 - ResetCalculation, 23
 - ResGetData, 23
 - ResGetDescription, 24
 - ResGetFirstValidObject, 24
 - ResGetFirstValidObjectVariable, 24
 - ResGetFirstValidVariable, 24
 - ResGetIndex, 24
 - ResGetMax, 25
 - ResGetMin, 25

- ResGetNextValidObject, 25
- ResGetNextValidObjectVariable, 25
- ResGetNextValidVariable, 25
- ResGetObject, 25
- ResGetUnit, 26
- ResGetValueCount, 26
- ResGetVariable, 26
- ResGetVariableCount, 26
- ResLoadData, 26
- ResReleaseData, 26
- ResSortToVariable, 26
- SaveAsScenario, 27
- SearchObjectByForeignKey, 27
- SelectToolbox, 27
- SetAttributeModelInternal, 28
- SetInterfaceVersion, 28
- SetShowAllUsers, 28
- SetWriteCacheEnabled, 29
- Show, 29
- SplitLine, 29
- StatFileGetXrange, 30
- StatFileResetXrange, 30
- StatFileSetXrange, 30
- WriteChangesToDb, 30
- Apply
 - ElmBmu, 83
 - IntOutage, 419
 - IntScenario, 439
 - IntSubset, 449
- ApplyAll
 - IntOutage, 420
- ApplyAndResetRA
 - ElmSubstat, 130
- ApplyNetworkState
 - IntCase, 383
- ApplySelective
 - IntScenario, 439
 - IntSubset, 449
- ApplyStudyTime
 - IntCase, 384
- Archive
 - IntPrj, 424
- AreDistParamsPossible
 - ElmLne, 99
- AssumeCompensationFactor
 - RelZpol, 465
- AssumeReRI
 - RelZpol, 465
- AssumeXeXI
 - RelZpol, 465
- BeginDataExtensionModification
 - IntPrj, 425
- BlkDef, 362
 - CalculateChecksum, 363
 - Compile, 362
 - Encrypt, 362
 - GetChecksum, 363
 - Pack, 363
 - PackAsMacro, 363
 - ResetThirdPartyModule, 364
 - SetThirdPartyModule, 364
- BlkSig, 364
 - GetFromSigName, 364
 - GetToSigName, 365
- BlockSwitch
 - ComShctrace, 306
- BuildNodeNames
 - ComUcteexp, 326
- CalcAggrVarsInRadFeed
 - ElmFeeder, 90
- CalcCluster
 - SetCluster, 330
 - SetDistrstate, 346
- CalcContributions
 - ComRelpost, 302
- CalcEfficiency
 - ElmAsm, 76
 - ElmGenstat, 94
 - ElmPvsys, 107
 - ElmSym, 135
 - ElmXnet, 161
- CalcElParams
 - TypAsmo, 476
- CalcMaxHostedPower
 - ComHostcap, 278
- CalcShiftedReversedBoundary
 - ElmBoundary, 84
- CalculateChecksum
 - BlkDef, 363
- CalculateInterchangeTo
 - ElmArea, 75
 - ElmNet, 106
 - ElmZone, 162
- CanAddProjectToRemoteDatabase
 - IntPrj, 425
- CanSubscribeProjectReadOnly
 - IntPrj, 425
- CanSubscribeProjectReadWrite
 - IntPrj, 425
- CentreOrigin
 - VisDraw, 483
 - VisPcompdiffplt, 492
 - VisVec, 506
 - VisVecres, 506
- ChangeFont
 - SetLevelvis, 347
- ChangeFrameAndWidth
 - SetLevelvis, 347
- ChangeLayer
 - SetLevelvis, 347

- ChangeRefPoints
 - SetLevelvis, 348
- ChangeWidthVisibilityAndColour
 - SetLevelvis, 348
- ChaVecfile, 365
 - Update, 365
- Check
 - ComAuditlog, 246
 - IntOutage, 420
 - SetTboxconfig, 355
- CheckAll
 - IntOutage, 420
- CheckAssignments
 - ComMerge, 284
- CheckBbPath
 - ElmBbone, 81
- CheckControllers
 - ComLdf, 280
- CheckRanges
 - ElmRelay, 110
- CheckSyntax
 - ComDpl, 270
- CheckUrl
 - IntExtaccess, 405
- CimModel, 365
 - DeleteParameterAtIndex, 366
 - GetAttributeEnumerationType, 366
 - GetModelsReferencingThis, 366
 - GetParameterCount, 366
 - GetParameterNamespace, 366
 - GetParameterValue, 367
 - HasParameter, 367
 - RemoveParameter, 367
 - SetAssociationValue, 367, 368
 - SetAttributeEnumeration, 368
 - SetAttributeValue, 369
- CimObject, 370
 - DeleteParameterAtIndex, 370
 - GetAttributeEnumerationType, 370
 - GetObjectsReferencingThis, 370
 - GetObjectsWithSameld, 371
 - GetParameterCount, 371
 - GetParameterNamespace, 371
 - GetParameterValue, 371
 - GetPfObjects, 372
 - HasParameter, 372
 - RemoveParameter, 372
 - SetAssociationValue, 372, 373
 - SetAttributeEnumeration, 373
 - SetAttributeValue, 374
- Clear
 - ComNmink, 290
 - ElmBoundary, 85
 - ElmRes, 116
 - IntDataset, 397
 - IntDplmap, 399
 - IntDplvec, 402
 - IntSubset, 450
 - Output Window Methods, 46
 - SetSelect, 355
 - VisBdia, 481
 - VisDraw, 483
 - VisHrm, 484
 - VisMagndiffplt, 487
 - VisOcplot, 489
 - VisPath, 490
 - VisPcompdiffplt, 492
 - VisPlot, 494
 - VisPlot2, 499
 - VisPlottz, 505
 - VisXyplot, 507
- ClearCont
 - ComSimoutage, 312
- ClearCurves
 - PltDataseries, 460
- ClearData
 - IntGrfgroup, 406
 - IntGrflayer, 407
- ClearInvalidReferences
 - IntPrj, 426
- ClearOutputWindow
 - Output Window Functions, 44
- ClearRecycleBin
 - Application Methods, 6
- ClearUpdatePages
 - ComProtgraphic, 296
- ClearVars
 - IntMon, 418
- Close
 - ElmCoup, 87
 - ElmGndswt, 97
 - ElmRes, 118
 - IntGrfnet, 408
 - SetDeskpage, 339
 - SetDesktop, 340
 - SetVipage, 358
 - StaSwitch, 233
- CloseTableReports
 - Dialogue Boxes Functions, 32
- ColLbl
 - IntMat, 411, 412
- ComAddlabel, 234
 - Execute, 234
- ComAddon, 235
 - CreateModule, 235
 - DefineDouble, 235
 - DefineDoubleMatrix, 236
 - DefineDoublePerConnection, 237
 - DefineDoubleVector, 237
 - DefineDoubleVectorPerConnection, 238
 - DefineInteger, 238
 - DefineIntegerPerConnection, 239

- DefineIntegerVector, 240
- DefineIntegerVectorPerConnection, 240
- DefineObject, 241
- DefineObjectPerConnection, 242
- DefineObjectVector, 242
- DefineObjectVectorPerConnection, 243
- DefineString, 243
- DefineStringPerConnection, 244
- DeleteModule, 245
- FinaliseModule, 245
- GetActiveModule, 245
- ModuleExists, 245
- SetActiveModule, 246
- ComAmpacity, 246
 - ExecuteAmpacityCalc, 246
- ComAuditlog, 246
 - Check, 246
- ComBoundary, 247
 - GetCreatedBoundaries, 247
- ComCapo, 247
 - ConnectShuntToBus, 247
 - LossCostAtBusTech, 248
 - TotalLossCost, 248
- ComCimdbexp, 249
 - Execute, 249
- ComCimdbimp, 249
 - Execute, 249
 - ImportAndConvert, 250
- ComCimvalidate, 250
 - Execute, 250
 - GetClassType, 251
 - GetDescriptionText, 251
 - GetInputObject, 251
 - GetModel, 251
 - GetModelId, 252
 - GetNumberOfValidationMessages, 252
 - GetObject, 252
 - GetObjectid, 252
 - GetProfile, 253
 - GetSeverity, 253
 - GetType, 253
- ComConreq, 253
 - Execute, 254
- ComContingency, 254
 - ContinueTrace, 254
 - CreateRecoveryInformation, 254
 - GetGeneratorEvent, 255
 - GetInterruptedPowerAndCustomersForStage, 255
 - GetInterruptedPowerAndCustomersForTimeStep, 256
 - GetLoadEvent, 256
 - GetNumberOfGeneratorEventsForTimeStep, 257
 - GetNumberOfLoadEventsForTimeStep, 257
 - GetNumberOfSwitchEventsForTimeStep, 257
- GetNumberOfTimeSteps, 258
- GetObj, 258
- GetSwitchEvent, 258
- GetTimeOfStepInSeconds, 259
- GetTotalInterruptedPower, 259
- JumpToLastStep, 259
- RemoveEvents, 259
- StartTrace, 260
- StopTrace, 260
- ComCoordreport, 260
 - DevicesToReport, 261
 - HasResultsForDirectionalBackup, 261
 - HasResultsForNonDirectionalBackup, 261
 - HasResultsForOverreach, 261
 - HasResultsForZone, 262
 - MaxZoneNumberFor, 262
 - ResultForDirectionalBackupVariable, 262
 - ResultForNonDirectionalBackupVariable, 263
 - ResultForOverreachVariable, 264
 - ResultForZoneVariable, 264
 - TopologyForDirectionalBackupVariable, 265
 - TopologyForNonDirectionalBackupVariable, 266
 - TopologyForOverreachVariable, 266
 - TopologyForZoneVariable, 266
 - TransferDirectionalBackupResultsTo, 267
 - TransferNonDirectionalBackupResultsTo, 267
 - TransferOverreachResultsTo, 268
 - TransferResultsTo, 269
 - TransferZoneResultsTo, 269
- ComDllmanager, 270
 - Report, 270
- ComDpl, 270
 - CheckSyntax, 270
 - Encrypt, 271
 - Execute, 271
 - GetExternalObject, 272
 - GetInputParameterDouble, 272
 - GetInputParameterInt, 273
 - GetInputParameterString, 273
 - IsEncrypted, 273
 - ResetThirdPartyModule, 274
 - SetExternalObject, 274
 - SetInputParameterDouble, 274
 - SetInputParameterInt, 275
 - SetInputParameterString, 275
 - SetThirdPartyModule, 275
- ComFlickermeter, 276
 - Execute, 276
- ComGenrelinc, 276
 - GetCurrentIteration, 276
 - GetMaxNumIterations, 277
- ComGridtocim, 277
 - ConvertAndExport, 277
 - SetAuthorityUri, 277
 - SetBoundaries, 278

- SetGridsToExport, 278
- ComHostcap, 278
 - CalcMaxHostedPower, 278
- ComImport, 279
 - GetCreatedObjects, 279
 - GetModifiedObjects, 279
- ComInc, 279
 - ZeroDerivative, 279
- ComLdf, 280
 - CheckControllers, 280
 - DoNotResetCalc, 280
 - EstimateOutage, 280
 - Execute, 281
 - IsAC, 281
 - IsBalanced, 281
 - IsDC, 281
 - PrintCheckResults, 282
 - SetOldDistributeLoadMode, 282
- ComLink, 282
 - LoadMicroSCADAFile, 282
 - ReceiveData, 283
 - SendData, 283
 - SentDataStatus, 283
 - SetOPCReceiveQuality, 283
 - SetSwitchShcEventMode, 284
- Commands Methods, 234
 - Execute, 234
- ComMerge, 284
 - CheckAssignments, 284
 - Compare, 285
 - CompareActive, 285
 - ExecuteRecording, 285
 - ExecuteWithActiveProject, 285
 - GetCorrespondingObject, 285
 - GetModification, 286
 - GetModificationResult, 286
 - GetModifiedObjects, 286
 - Merge, 287
 - PrintComparisonReport, 287
 - PrintModifications, 287
 - Reset, 288
 - SetAutoAssignmentForAll, 288
 - SetObjectsToCompare, 288
 - ShowBrowser, 288
 - WereModificationsFound, 289
- CommitTransaction
 - Application Methods, 6
- ComMot, 289
 - GetMotorConnections, 289
 - GetMotorSwitch, 289
 - GetMotorTerminal, 290
- ComNmink, 290
 - AddRef, 290
 - Clear, 290
 - GenerateContingenciesForAnalysis, 290
 - GetAll, 291
- ComOmr, 291
 - GetFeeders, 291
 - GetOMR, 291
 - GetRegionCount, 292
- ComOpc, 292
 - ReceiveData, 292
 - SendData, 292
- ComOutage, 293
 - ContinueTrace, 293
 - ExecuteTime, 293
 - GetObject, 293
 - RemoveEvents, 294
 - SetObjs, 294
 - StartTrace, 294
 - StopTrace, 294
- Compare
 - ComMerge, 285
- CompareActive
 - ComMerge, 285
- ComPfdimport, 295
 - GetImportedObjects, 295
- Compile
 - BlkDef, 362
- ComPrjconnector, 295
 - GetSuccessfullyConnectedItems, 295
 - GetUnsuccessfullyConnectedItems, 295
- ComProtgraphic, 296
 - AddToUpdatePages, 296
 - ClearUpdatePages, 296
- ComPvcurves, 296
 - FindCriticalBus, 296
- ComPython, 296
 - GetExternalObject, 296
 - GetInputParameterDouble, 297
 - GetInputParameterInt, 297
 - GetInputParameterString, 298
 - SetExternalObject, 298
 - SetInputParameterDouble, 298
 - SetInputParameterInt, 299
 - SetInputParameterString, 299
- ComRed, 300
 - ReductionInMemory, 300
 - ResetReductionInMemory, 300
- ComRel3, 300
 - AnalyseElmRes, 300
 - ExeEvt, 301
 - OvlAlleviate, 301
 - RemoveEvents, 301
 - RemoveOutages, 301
 - ValidateConstraints, 302
- ComRelpost, 302
 - CalcContributions, 302
 - GetContributionOfComponent, 302
- ComRelreport, 303
 - GetContingencies, 303
 - GetContributionOfComponent, 303

- ComRes, 303
 - ExportFullRange, 303
 - FileNmResNm, 304
- ComShc, 304
 - ExecuteRXSweep, 304
 - GetFaultType, 304
 - GetOverLoadedBranches, 305
 - GetOverLoadedBuses, 305
- ComShctrace, 306
 - BlockSwitch, 306
 - ExecuteAllSteps, 306
 - ExecuteInitialStep, 306
 - ExecuteNextStep, 307
 - GetBlockedSwitches, 307
 - GetCurrentTimeStep, 307
 - GetDeviceSwitches, 307
 - GetDeviceTime, 308
 - GetNonStartedDevices, 308
 - GetStartedDevices, 308
 - GetSwitchTime, 308
 - GetTrippedDevices, 308
 - NextStepAvailable, 309
- ComSim, 309
 - GetSimulationTime, 309
 - GetTotalWarnA, 309
 - GetTotalWarnB, 309
 - GetTotalWarnC, 310
 - GetViolatedScanModules, 310
 - LoadSimulationState, 310
 - LoadSnapshot, 310
 - SaveSimulationState, 310
 - SaveSnapshot, 310
- ComSimoutage, 311
 - AddCntcy, 311
 - AddContingencies, 312
 - AddRas, 312
 - ClearCont, 312
 - CreateFaultCase, 312
 - Execute, 313
 - ExecuteAndCheck, 313
 - GetNTopLoadedElms, 314
 - MarkRegions, 315
 - RemoveAllRas, 315
 - RemoveContingencies, 315
 - RemoveRas, 315
 - Reset, 315
 - SetLimits, 316
 - Update, 316
- ComSvgexport, 316
 - SetFileName, 316
 - SetObject, 316
 - SetObjects, 317
- ComSvgimport, 317
 - SetFileName, 317
 - SetObject, 317
- ComTasks, 317
 - AppendCommand, 318
 - AppendStudyCase, 318
 - GetCommandsForStudyCase, 318
 - GetNumberOfCommandsForStudyCase, 319
 - GetNumberOfStudyCases, 319
 - GetStudyCases, 319
 - IsAdditionalResultsFlagSetForCommand, 320
 - IsCommandIgnored, 320
 - IsStudyCaseIgnored, 321
 - RemoveCmdsForStudyCaseRow, 321
 - RemoveCommand, 321
 - RemoveStudyCase, 322
 - RemoveStudyCases, 322
 - SetAdditionalResultsFlagForCommand, 322
 - SetIgnoreFlagForCommand, 323
 - SetIgnoreFlagForStudyCase, 324
 - SetResultsFolder, 324
- ComTececo, 325
 - UpdateTablesByCalcPeriod, 325
- ComTransfer, 325
 - GetTransferCalcData, 325
 - IsLastIterationFeasible, 325
- ComUcte, 326
 - SetBatchMode, 326
- ComUctexp, 326
 - BuildNodeNames, 326
 - DeleteCompleteQuickAccess, 327
 - ExportAndInitQuickAccess, 327
 - GetConnectedBranches, 327
 - GetFromToNodeNames, 327
 - GetOrderCode, 328
 - GetUcteNodeName, 328
 - InitQuickAccess, 328
 - QuickAccessAvailable, 329
 - ResetQuickAccess, 329
 - SetGridSelection, 329
- ComWktmp, 329
 - GetCreatedObjects, 330
 - GetModifiedObjects, 330
- ConnectShuntToBus
 - ComCapo, 247
- Consolidate
 - IntCase, 384
 - IntScheme, 444
- Contains
 - IntDplmap, 399
- ContainsNonAsciiCharacters
 - General Object Methods, 50
- ContinueTrace
 - ComContingency, 254
 - ComOutage, 293
- ConvertAndExport
 - ComGridtocim, 277
- ConvertToASCIIFormat
 - IntComtrade, 385
- ConvertToBinaryFormat

- IntComtrade, 386
- CopyData
 - General Object Methods, 50
- CopyDataExtensionFrom
 - IntPrj, 426
- CopyExtMeaStatusToStatusTmp
 - StaExtbrkmea, 168
 - StaExtcmdmea, 173
 - StaExtmdatmea, 178
 - StaExtfmea, 183
 - StaExtfuelmea, 188
 - StaExttimea, 193
 - StaExtpfmea, 198
 - StaExtptimea, 203
 - StaExtqmea, 208
 - StaExtsmea, 213
 - StaExttapmea, 218
 - StaExtv3mea, 223
 - StaExtvmea, 228
- Create
 - SetPath, 351
- CreateCBEvents
 - IntEvt, 404
- CreateDerivedProject
 - IntVersion, 458
- CreateEvent
 - ElmTr2, 144
 - ElmTr3, 147
 - ElmTr4, 151
 - ElmVOLTreg, 159
 - StaExtmdatmea, 178
- CreateFaultCase
 - Application Methods, 6
 - ComSimoutage, 312
- CreateFeederWithRoutes
 - ElmLne, 100
- CreateFilter
 - SetColscheme, 331
- CreateGroup
 - IntUserman, 452
- CreateModule
 - ComAddon, 235
- CreateObject
 - General Object Methods, 50
- CreateProject
 - Application Methods, 7
- CreateRecoveryInformation
 - ComContingency, 254
- CreateStageObject
 - IntSstage, 447
- CreateUser
 - IntUserman, 453
- CreateVersion
 - IntPrj, 426
- CreateVI
 - SetVipage, 359
- Date/Time Functions, 32
 - GetStudyTimeObject, 32
- Date
 - SetTime, 356
- Deactivate
 - ElmNet, 106
 - IntCase, 385
 - IntLibrary, 410
 - IntPrj, 427
 - IntScenario, 440
 - IntScensched, 442
 - IntScheme, 444
 - IntSscheduler, 446
- DefineBoundary
 - ElmArea, 75
 - ElmNet, 106
 - ElmZone, 163
- DefineDouble
 - ComAddon, 235
- DefineDoubleMatrix
 - ComAddon, 236
- DefineDoublePerConnection
 - ComAddon, 237
- DefineDoubleVector
 - ComAddon, 237
- DefineDoubleVectorPerConnection
 - ComAddon, 238
- DefineInteger
 - ComAddon, 238
- DefineIntegerPerConnection
 - ComAddon, 239
- DefineIntegerVector
 - ComAddon, 240
- DefineIntegerVectorPerConnection
 - ComAddon, 240
- DefineObject
 - ComAddon, 241
- DefineObjectPerConnection
 - ComAddon, 242
- DefineObjectVector
 - ComAddon, 242
- DefineObjectVectorPerConnection
 - ComAddon, 243
- DefineString
 - ComAddon, 243
- DefineStringPerConnection
 - ComAddon, 244
- DefineTransferAttributes
 - Application Methods, 7
- Delete
 - General Object Methods, 51
- DeleteCompleteQuickAccess
 - ComUctexp, 327
- DeleteModule
 - ComAddon, 245
- DeleteParameterAtIndex

- CimModel, 366
- CimObject, 370
- DeleteRow
 - IntScensched, 442
- DeleteUntouchedObjects
 - Application Methods, 7
- Derate
 - ElmGenstat, 94
 - ElmPvsys, 107
 - ElmSym, 135
- DevicesToReport
 - ComCoordreport, 261
- Dialogue Boxes Functions, 32
 - CloseTableReports, 32
 - GetTableReports, 32
 - ShowModalBrowser, 33
 - ShowModalSelectBrowser, 33
 - ShowModelessBrowser, 34
 - UpdateTableReports, 34
- DiscardChanges
 - IntScenario, 440
- Disconnect
 - ElmGenstat, 95
 - ElmPvsys, 107
 - ElmSym, 135
 - ElmXnet, 161
- DoAutoScale
 - GrpPage, 375
 - PltLinebarplot, 461
- DoAutoScaleOnAll
 - VisDraw, 483
- DoAutoScaleOnCharacteristics
 - VisDraw, 483
- DoAutoScaleOnImpedances
 - VisDraw, 483
- DoAutoScaleX
 - GrpPage, 375
 - SetDesktop, 341
 - SetVipage, 358
 - VisDraw, 483
 - VisHrm, 484
 - VisMagndiffplt, 487
 - VisOcplot, 489
 - VisPath, 490
 - VisPcompdiffplt, 492
 - VisPlot, 494
 - VisPlot2, 500
 - VisPlottz, 506
 - VisXyplot, 507
- DoAutoScaleY2
 - VisPlot2, 500
- DoAutoScaleY
 - GrpPage, 375
 - SetVipage, 358
 - VisDraw, 484
 - VisHrm, 484
- VisMagndiffplt, 488
- VisOcplot, 489
- VisPath, 490
- VisPcompdiffplt, 493
- VisPlot, 494
- VisPlot2, 500
- VisPlottz, 506
- VisXyplot, 507
- DoNotResetCalc
 - ComLdf, 280
- EchoOff
 - Environment Functions, 35
- EchoOn
 - Environment Functions, 35
- ElmArea, 74
 - CalculateInterchangeTo, 75
 - DefineBoundary, 75
 - GetAll, 75
 - GetBranches, 75
 - GetBuses, 76
 - GetObjs, 76
- ElmAsm, 76
 - CalcEfficiency, 76
 - GetAvailableGenPower, 76
 - GetElecTorque, 77
 - GetGroundingImpedance, 78
 - GetMechTorque, 78
 - GetMotorStartingFlag, 78
 - GetStepupTransformer, 79
 - IsPQ, 79
- ElmAsmsc, 79
 - GetAvailableGenPower, 79
 - GetGroundingImpedance, 80
 - GetStepupTransformer, 80
- ElmBbone, 81
 - CheckBbPath, 81
 - GetBbOrder, 81
 - GetCompleteBbPath, 82
 - GetFOR, 82
 - GetMeanCs, 82
 - GetMinCs, 82
 - GetTieOpenPoint, 83
 - GetTotLength, 83
 - HasGnrlMod, 83
- ElmBmu, 83
 - Apply, 83
 - Update, 84
- ElmBoundary, 84
 - AddCubicle, 84
 - CalcShiftedReversedBoundary, 84
 - Clear, 85
 - GetInterior, 85
 - IsSplitting, 85
 - Resize, 85
 - Update, 86

- ElmBranch, 86
 - Update, 86
- ElmCabsys, 86
 - FitParams, 86
 - GetLineCable, 86
 - Update, 87
- ElmComp, 87
 - slotupd, 87
 - SlotUpdate, 87
- ElmCoup, 87
 - Close, 87
 - GetRemoteBreakers, 88
 - IsBreaker, 88
 - IsClosed, 89
 - IsOpen, 89
 - Open, 89
- ElmDsl, 89
 - ExportToClipboard, 89
 - ExportToFile, 90
- ElmFeeder, 90
 - CalcAggrVarsInRadFeed, 90
 - GetAll, 91
 - GetBranches, 91
 - GetBuses, 92
 - GetNodesBranches, 92
 - GetObjs, 92
- ElmFile, 93
 - LoadFile, 93
 - SaveFile, 93
- ElmFilter, 93
 - GetGroundingImpedance, 93
- ElmGenstat, 94
 - CalcEfficiency, 94
 - Derate, 94
 - Disconnect, 95
 - GetAvailableGenPower, 95
 - GetGroundingImpedance, 95
 - GetStepupTransformer, 96
 - IsConnected, 96
 - Reconnect, 96
 - ResetDerating, 97
- ElmGndswt, 97
 - Close, 97
 - GetGroundingImpedance, 97
 - IsClosed, 98
 - IsOpen, 98
 - Open, 98
- ElmLne, 99
 - AreDistParamsPossible, 99
 - CreateFeederWithRoutes, 100
 - FitParams, 100
 - GetIthr, 100
 - GetType, 101
 - GetY0m, 101
 - GetY1m, 101
 - GetZ0m, 102
 - GetZ1m, 102
 - GetZmatDist, 103
 - HasRoutes, 103
 - HasRoutesOrSec, 103
 - IsCable, 103
 - IsNetCoupling, 104
 - MeasureLength, 104
 - SetDetailed, 104
- ElmLneSec, 104
 - IsCable, 105
- ElmNec, 105
 - GetGroundingImpedance, 105
- ElmNet, 105
 - Activate, 105
 - CalculateInterchangeTo, 106
 - Deactivate, 106
 - DefineBoundary, 106
- ElmPvsys, 107
 - CalcEfficiency, 107
 - Derate, 107
 - Disconnect, 107
 - GetAvailableGenPower, 108
 - GetGroundingImpedance, 108
 - IsConnected, 109
 - Reconnect, 109
 - ResetDerating, 109
- ElmRelay, 109
 - CheckRanges, 110
 - GetCalcRX, 110
 - GetMaxFdetectCalcI, 110
 - GetSlot, 111
 - GetUnom, 111
 - IsStarted, 111
 - SetImpedance, 112
 - SetMaxI, 113
 - SetMaxIearth, 113
 - SetMinI, 113
 - SetMinIearth, 113
 - SetOutOfService, 113
 - SetTime, 114
 - slotupd, 114
 - SlotUpdate, 114
- ElmRes, 115
 - AddVariable, 115
 - AddVars, 115
 - Clear, 116
 - Close, 118
 - FindColumn, 116
 - FindMaxInColumn, 116
 - FindMaxOfVariableInRow, 117
 - FindMinInColumn, 117
 - FindMinOfVariableInRow, 117
 - FinishWriting, 118
 - Flush, 118
 - GetDescription, 118
 - GetFirstValidObject, 119

- GetFirstValidObjectVariable, 120
- GetFirstValidVariable, 120
- GetNextValidObject, 120
- GetNextValidObjectVariable, 122
- GetNextValidVariable, 122
- GetNumberOfColumns, 122
- GetNumberOfRows, 122
- GetObj, 123
- GetObject, 123
- GetObjectValue, 123
- GetRelCase, 124
- GetSubElmRes, 124
- GetUnit, 124
- GetValue, 125
- GetVariable, 125
- Init, 125
- InitialiseWriting, 125
- Load, 126
- NCol, 122
- NRow, 122
- Release, 126
- SetAsDefault, 126
- SetObj, 126
- SetSubElmResKey, 127
- SizeX, 122
- SizeY, 122
- SortAccordingToColumn, 127
- Write, 127
- WriteDraw, 127
- ElmShnt, 128
 - GetGroundingImpedance, 128
- ElmStactrl, 128
 - GetControlledHVNode, 128
 - GetControlledLVNode, 129
 - GetStepupTransformer, 129
 - Info, 129
- ElmSubstat, 130
 - ApplyAndResetRA, 130
 - GetSplit, 130
 - GetSplitCal, 131
 - GetSplitIndex, 132
 - GetSuppliedElements, 132
 - OverwriteRA, 133
 - ResetRA, 133
 - SaveAsRA, 133
 - SetRA, 133
- ElmSvs, 134
 - GetStepupTransformer, 134
- ElmSym, 134
 - CalcEfficiency, 135
 - Derate, 135
 - Disconnect, 135
 - GetAvailableGenPower, 135
 - GetGroundingImpedance, 136
 - GetMotorStartingFlag, 136
 - GetStepupTransformer, 137
 - IsConnected, 137
 - Reconnect, 137
 - ResetDerating, 137
- ElmTerm, 138
 - GetBusType, 138
 - GetCalcRelevantCubicles, 138
 - GetConnectedBrkCubicles, 138
 - GetConnectedCubicles, 139
 - GetConnectedMainBuses, 139
 - GetConnectionInfo, 139
 - GetEquivalentTerminals, 140
 - GetMinDistance, 140
 - GetNextHVBUS, 141
 - GetNodeName, 141
 - GetSepStationAreas, 141
 - HasCreatedCalBus, 142
 - IsElectrEquivalent, 142
 - IsEquivalent, 143
 - IsInternalNodeInStation, 143
 - UpdateSubstationTerminals, 143
- ElmTr2, 144
 - CreateEvent, 144
 - GetGroundingImpedance, 144
 - GetSuppliedElements, 145
 - GetTapPhi, 145
 - GetTapRatio, 146
 - GetZ0pu, 146
 - GetZpu, 146
 - IsQuadBooster, 147
 - NTap, 147
- ElmTr3, 147
 - CreateEvent, 147
 - GetGroundingImpedance, 148
 - GetSuppliedElements, 148
 - GetTapPhi, 149
 - GetTapRatio, 149
 - GetTapZDependentSide, 149
 - GetZ0pu, 150
 - GetZpu, 150
 - IsQuadBooster, 151
 - NTap, 151
- ElmTr4, 151
 - CreateEvent, 151
 - GetGroundingImpedance, 152
 - GetSuppliedElements, 152
 - GetTapPhi, 153
 - GetTapRatio, 153
 - GetTapZDependentSide, 154
 - GetZ0pu, 154
 - GetZpu, 155
 - IsQuadBooster, 155
 - NTap, 155
- ElmTrfstat, 156
 - GetSplit, 156
 - GetSplitCal, 157
 - GetSplitIndex, 157

- GetSuppliedElements, 158
- ElmVac, 158
 - GetGroundingImpedance, 159
- ElmVoltreg, 159
 - CreateEvent, 159
 - GetGroundingImpedance, 159
 - GetZpu, 160
 - NTap, 160
- ElmXnet, 160
 - CalcEfficiency, 161
 - Disconnect, 161
 - GetGroundingImpedance, 161
 - GetStepupTransformer, 161
 - Reconnect, 162
- ElmZone, 162
 - CalculateInterchangeTo, 162
 - DefineBoundary, 163
 - GetAll, 163
 - GetBranches, 163
 - GetBuses, 163
 - GetObjs, 163
 - SetLoadScaleAbsolute, 164
- EnableDiffMode
 - IntSstage, 447
- Encrypt
 - BlkDef, 362
 - ComDpl, 271
 - TypQdsl, 478
- EndDataExtensionModification
 - IntPrj, 427
- Energize
 - General Object Methods, 51
- Environment Functions, 34
 - EchoOff, 35
 - EchoOn, 35
 - IsAutomaticCalculationResetEnabled, 35
 - IsFinalEchoOnEnabled, 35
 - SetAutomaticCalculationResetEnabled, 35
 - SetEnableUserBreak, 37
 - SetFinalEchoOnEnabled, 36
 - SetGraphicUpdate, 36
 - SetGuiUpdateEnabled, 36
 - SetProgressBarUpdatesEnabled, 37
 - SetRescheduleFlag, 36
 - SetUserBreakEnabled, 37
- EstimateOutage
 - ComLdf, 280
- Execute
 - ComAddlabel, 234
 - ComCimdbexp, 249
 - ComCimdbimp, 249
 - ComCimvalidate, 250
 - ComConreq, 254
 - ComDpl, 271
 - ComFlickermeter, 276
 - ComLdf, 281
 - Commands Methods, 234
 - ComSimoutage, 313
- ExecuteAllSteps
 - ComShctrace, 306
- ExecuteAmpacityCalc
 - ComAmpacity, 246
- ExecuteAndCheck
 - ComSimoutage, 313
- ExecuteCmd
 - Application Methods, 8
- ExecuteInitialStep
 - ComShctrace, 306
- ExecuteNextStep
 - ComShctrace, 307
- ExecuteRecording
 - ComMerge, 285
- ExecuteRXSweep
 - ComShc, 304
- ExecuteTime
 - ComOutage, 293
- ExecuteWithActiveProject
 - ComMerge, 285
- ExeEvt
 - ComRel3, 301
- Export
 - IntDocument, 398
 - IntGrfgroup, 406
 - IntGrflayer, 407
 - IntIcon, 409
- ExportAndInitQuickAccess
 - ComUctexp, 327
- ExportFullRange
 - ComRes, 303
- ExportToClipboard
 - ElmDsl, 89
- ExportToFile
 - ElmDsl, 90
- ExportToVec
 - IntGrflayer, 407
- File System Functions, 31
 - GetInstallationDirectory, 31
 - GetInstallDir, 31
 - GetTempDir, 31
 - GetTemporaryDirectory, 31
 - GetWorkingDir, 31
 - GetWorkspaceDirectory, 31
- FileNmResNm
 - ComRes, 304
- FinaliseModule
 - ComAddon, 245
- FindColumn
 - ElmRes, 116
 - IntComtrade, 386
 - IntComtradeset, 392
- FindCriticalBus

- ComPvcurses, 296
- FindMaxInColumn
 - ElmRes, 116
 - IntComtrade, 386
 - IntComtradeset, 392
- FindMaxOfVariableInRow
 - ElmRes, 117
- FindMinInColumn
 - ElmRes, 117
 - IntComtrade, 387
 - IntComtradeset, 392
- FindMinOfVariableInRow
 - ElmRes, 117
- FinishWriting
 - ElmRes, 118
- First
 - IntDplmap, 399
- FitParams
 - ElmCabsys, 86
 - ElmLne, 100
- Flush
 - ElmRes, 118
- Freeze
 - SetDesktop, 341
- General Object Methods, 48
 - AddCopy, 49
 - ContainsNonAsciiCharacters, 50
 - CopyData, 50
 - CreateObject, 50
 - Delete, 51
 - Energize, 51
 - GetAttribute, 52
 - GetAttributeDescription, 52
 - GetAttributeLength, 52
 - GetAttributes, 53
 - GetAttributeShape, 53
 - GetAttributeType, 53
 - GetAttributeUnit, 54
 - GetChildren, 54
 - GetClassName, 55
 - GetCombinedProjectSource, 55
 - GetConnectedElements, 55
 - GetConnectionCount, 56
 - GetContents, 56
 - GetControlledNode, 56
 - GetCubicle, 57
 - GetFullName, 57
 - GetImpedance, 57
 - GetInom, 58
 - GetNode, 58
 - GetOperator, 59
 - GetOwner, 59
 - GetParent, 59
 - GetReferences, 60
 - GetRegion, 60
 - GetSupplyingSubstations, 60
 - GetSupplyingTransformers, 61
 - GetSupplyingTrfstations, 61
 - GetSystemGrounding, 61
 - GetUnom, 62
 - GetZeroImpedance, 62
 - HasAttribute, 62
 - HasResults, 63
 - IsCalcRelevant, 63
 - IsDeleted, 63
 - IsEarthed, 64
 - IsEnergized, 64
 - IsHidden, 64
 - IsInFeeder, 64
 - IsNetworkDataFolder, 65
 - IsNode, 66
 - IsObjectActive, 66
 - IsObjectModifiedByVariation, 66
 - Isolate, 67
 - IsOutOfService, 67
 - IsReducible, 67
 - IsShortCircuited, 68
 - MarkInGraphics, 68
 - Move, 68
 - PasteCopy, 69
 - PurgeUnusedObjects, 69
 - ReplaceNonAsciiCharacters, 70
 - ReportNonAsciiCharacters, 70
 - ReportUnusedObjects, 70
 - SearchObject, 70
 - SetAttribute, 71
 - SetAttributeLength, 71
 - SetAttributes, 71
 - SetAttributeShape, 72
 - ShowEditDialog, 72
 - ShowModalSelectTree, 72
 - SwitchOff, 73
 - SwitchOn, 73
 - WriteChangesToDb, 74
- GenerateContingenciesForAnalysis
 - ComNmink, 290
- Get
 - IntDplvec, 402
 - IntMat, 411
 - IntVec, 454
 - IntVecobj, 456
 - SetFilt, 346
- GetActiveCalculationStr
 - Application Methods, 8
- GetActiveModule
 - ComAddon, 245
- GetActiveNetworkVariations
 - Application Methods, 9
- GetActivePage
 - SetDesktop, 341
- GetActiveProject

- Application Methods, 9
- GetActiveScenario
 - Application Methods, 9
- GetActiveScenarioScheduler
 - Application Methods, 9
- GetActiveScheduler
 - IntScheme, 445
- GetActiveStages
 - Application Methods, 9
- GetActiveStudyCase
 - Application Methods, 10
- GetAll
 - ComNmink, 291
 - ElmArea, 75
 - ElmFeeder, 91
 - ElmZone, 163
 - IntDataset, 397
 - SetPath, 351
 - SetSelect, 355
 - StaCubic, 165
- GetAllUsers
 - Application Methods, 10
- GetAnalogueDescriptions
 - IntComtrade, 387
 - IntComtradeset, 393
- GetApplication
 - Module Functions, 2
- GetApplicationExt
 - Module Functions, 3
- GetAttribute
 - General Object Methods, 52
- GetAttributeDescription
 - Application Methods, 10
 - General Object Methods, 52
- GetAttributeEnumerationType
 - CimModel, 366
 - CimObject, 370
- GetAttributeLength
 - General Object Methods, 52
- GetAttributes
 - General Object Methods, 53
- GetAttributeShape
 - General Object Methods, 53
- GetAttributeType
 - General Object Methods, 53
- GetAttributeUnit
 - Application Methods, 11
 - General Object Methods, 54
- GetAvailableButtons
 - SetTboxconfig, 355
- GetAvailableGenPower
 - ElmAsm, 76
 - ElmAsmsc, 79
 - ElmGenstat, 95
 - ElmPvsys, 108
 - ElmSym, 135
- GetAxisX
 - PltLinebarplot, 461
- GetAxisY
 - PltLinebarplot, 462
- GetBbOrder
 - ElmBbone, 81
- GetBlockedSwitches
 - ComShctrace, 307
- GetBorderCubicles
 - Application Methods, 11
- GetBranch
 - StaCubic, 165
- GetBranches
 - ElmArea, 75
 - ElmFeeder, 91
 - ElmZone, 163
 - SetPath, 351
- GetBrowserSelection
 - Application Methods, 11
- GetBuses
 - ElmArea, 76
 - ElmFeeder, 92
 - ElmZone, 163
 - SetPath, 352
- GetBusType
 - ElmTerm, 138
- GetCalcRelevantCubicles
 - ElmTerm, 138
- GetCalcRelevantObjects
 - Application Methods, 12
- GetCalcRX
 - ElmRelay, 110
- GetCanvasSize
 - SetDesktop, 341
- GetChecksum
 - BlkDef, 363
- GetChildren
 - General Object Methods, 54
- GetClassDescription
 - Application Methods, 12
- GetClassId
 - Application Methods, 13
- GetClassName
 - General Object Methods, 55
- GetClassType
 - ComCimvalidate, 251
- GetColumnLabel
 - IntMat, 412
- GetColumnLabelIndex
 - IntMat, 412
- GetCombinedProjectSource
 - General Object Methods, 55
- GetCommandsForStudyCase
 - ComTasks, 318
- GetCompleteBbPath
 - ElmBbone, 82

- GetConfiguration
 - IntSubset, [450](#)
 - SetDataext, [338](#)
- GetConfigurations
 - SetDataext, [338](#)
- GetConnectedBranches
 - ComUcteexp, [327](#)
- GetConnectedBrkCubicles
 - ElmTerm, [138](#)
- GetConnectedCubicles
 - ElmTerm, [139](#)
- GetConnectedElements
 - General Object Methods, [55](#)
- GetConnectedMainBuses
 - ElmTerm, [139](#)
- GetConnectedMajorNodes
 - StaCubic, [165](#)
- GetConnectionCount
 - General Object Methods, [56](#)
- GetConnectionInfo
 - ElmTerm, [139](#)
- GetConnections
 - StaCubic, [166](#)
- GetContent
 - Output Window Methods, [46](#)
- GetContents
 - General Object Methods, [56](#)
- GetContingencies
 - ComRelreport, [303](#)
- GetContributionOfComponent
 - ComRelpost, [302](#)
 - ComRelreport, [303](#)
- GetControlledHVNode
 - ElmStactrl, [128](#)
- GetControlledLVNode
 - ElmStactrl, [129](#)
- GetControlledNode
 - General Object Methods, [56](#)
- GetCorrespondingObject
 - ComMerge, [285](#)
- GetCreatedBoundaries
 - ComBoundary, [247](#)
- GetCreatedObjects
 - ComImport, [279](#)
 - ComWktmp, [330](#)
- GetCriticalTimePhase
 - IntThrating, [450](#)
- GetCubicle
 - General Object Methods, [57](#)
- GetCurrentDiagram
 - Application Methods, [13](#)
- GetCurrentIteration
 - ComGenrelinc, [276](#)
- GetCurrentScript
 - Application Methods, [13](#)
- GetCurrentSelection
 - Application Methods, [13](#)
- GetCurrentTimeStep
 - ComShctrace, [307](#)
- GetCurrentUser
 - Application Methods, [14](#)
- GetCurrentZoomScaleLevel
 - Application Methods, [14](#)
- GetDataFolder
 - Application Methods, [14](#)
- GetDataSeries
 - PltLinebarplot, [462](#)
- GetDataSource
 - PltDataseries, [460](#)
 - VisHrm, [485](#)
 - VisPlot, [494](#)
 - VisPlot2, [500](#)
 - VisXyplot, [507](#)
- GetDerivedProjects
 - IntPrj, [428](#)
 - IntVersion, [458](#)
- GetDescription
 - ElmRes, [118](#)
 - IntComtrade, [387](#)
 - IntComtradeset, [393](#)
- GetDescriptionText
 - ComCimvalidate, [251](#)
- GetDeviceSwitches
 - ComShctrace, [307](#)
- GetDeviceTime
 - ComShctrace, [308](#)
- GetDiagramSelection
 - Application Methods, [15](#)
- GetDigitalDescriptions
 - IntComtrade, [388](#)
 - IntComtradeset, [393](#)
- GetDisplayedButtons
 - SetTboxconfig, [356](#)
- GetElecTorque
 - ElmAsm, [77](#)
- GetEquivalentTerminals
 - ElmTerm, [140](#)
- GetExternalObject
 - ComDpl, [272](#)
 - ComPython, [296](#)
- GetExternalReferences
 - IntPrj, [428](#)
- GetFaultType
 - ComShc, [304](#)
- GetFeeders
 - ComOmr, [291](#)
- GetFirstValidObject
 - ElmRes, [119](#)
- GetFirstValidObjectVariable
 - ElmRes, [120](#)
- GetFirstValidVariable
 - ElmRes, [120](#)

- GetFlowOrientation
 - Application Methods, 15
- GetFOR
 - ElmBbone, 82
- GetFromSigName
 - BlkSig, 364
- GetFromStudyCase
 - Application Methods, 15
- GetFromToNodeNames
 - ComUcteexp, 327
- GetFullName
 - General Object Methods, 57
- GetGeneratorEvent
 - ComContingency, 255
- GetGeoCoordinateSystem
 - IntPrj, 428
- GetGlobalLibrary
 - Application Methods, 16
- GetGraphicsBoard
 - Application Methods, 16
- GetGroundingImpedance
 - ElmAsm, 78
 - ElmAsmsc, 80
 - ElmFilter, 93
 - ElmGenstat, 95
 - ElmGndswt, 97
 - ElmNec, 105
 - ElmPvsys, 108
 - ElmShnt, 128
 - ElmSym, 136
 - ElmTr2, 144
 - ElmTr3, 148
 - ElmTr4, 152
 - ElmVac, 159
 - ElmVltreg, 159
 - ElmXnet, 161
- GetGroups
 - IntUserman, 453
- GetHistoricalProject
 - IntVersion, 458
- GetImpedance
 - General Object Methods, 57
- GetImportedObjects
 - ComPfdimport, 295
- GetInom
 - General Object Methods, 58
- GetInputObject
 - ComCimvalidate, 251
- GetInputParameterDouble
 - ComDpl, 272
 - ComPython, 297
- GetInputParameterInt
 - ComDpl, 273
 - ComPython, 297
- GetInputParameterString
 - ComDpl, 273
- ComPython, 298
- GetInstallationDirectory
 - File System Functions, 31
- GetInstallDir
 - File System Functions, 31
- GetIntCalcrec
 - PltDataseries, 460
 - VisPlot, 495
- GetInterfaceVersion
 - Application Methods, 16
- GetInterior
 - ElmBoundary, 85
- GetInterruptedPowerAndCustomersForStage
 - ComContingency, 255
- GetInterruptedPowerAndCustomersForTimeStep
 - ComContingency, 256
- GetIthr
 - ElmLne, 100
- GetLanguage
 - Application Methods, 16
- GetLatestVersion
 - IntPrj, 429
- GetLegend
 - PltLinebarplot, 462
- GetLimit
 - ScnFreq, 466
 - ScnFrt, 467
 - ScnSpeed, 469
 - ScnSync, 471
 - ScnVar, 472
 - ScnVolt, 474
- GetLineCable
 - ElmCabsys, 86
- GetLoadEvent
 - ComContingency, 256
- GetLocalLibrary
 - Application Methods, 17
- GetMaxFdetectCalcl
 - ElmRelay, 110
- GetMaxNumIterations
 - ComGenrelinec, 277
- GetMeanCs
 - ElmBbone, 82
- GetMeaValue
 - StaExtbrkmea, 168
 - StaExtcmdmea, 173
 - StaExtatmea, 178
 - StaExtfmea, 183
 - StaExtfuelmea, 188
 - StaExttimea, 193
 - StaExtpfmea, 198
 - StaExtptimea, 203
 - StaExtqmea, 208
 - StaExttapmea, 218
 - StaExtv3mea, 223
 - StaExtvmea, 228

- GetMechTorque
 - ElmAsm, [78](#)
- GetMem
 - Application Methods, [17](#)
- GetMinCs
 - ElmBbone, [82](#)
- GetMinDistance
 - ElmTerm, [140](#)
- GetModel
 - ComCimvalidate, [251](#)
- GetModelId
 - ComCimvalidate, [252](#)
- GetModelsReferencingThis
 - CimModel, [366](#)
- GetModification
 - ComMerge, [286](#)
- GetModificationResult
 - ComMerge, [286](#)
- GetModifiedObjects
 - ComImport, [279](#)
 - ComMerge, [286](#)
 - ComWktimp, [330](#)
- GetMotorConnections
 - ComMot, [289](#)
- GetMotorStartingFlag
 - ElmAsm, [78](#)
 - ElmSym, [136](#)
- GetMotorSwitch
 - ComMot, [289](#)
- GetMotorTerminal
 - ComMot, [290](#)
- GetNearestBusbars
 - StaCubic, [166](#)
- GetNextHVBus
 - ElmTerm, [141](#)
- GetNextValidObject
 - ElmRes, [120](#)
- GetNextValidObjectVariable
 - ElmRes, [122](#)
- GetNextValidVariable
 - ElmRes, [122](#)
- GetNode
 - General Object Methods, [58](#)
- GetNodeName
 - ElmTerm, [141](#)
- GetNodesBranches
 - ElmFeeder, [92](#)
- GetNonStartedDevices
 - ComShctrace, [308](#)
- GetNTopLoadedElms
 - ComSimoutage, [314](#)
- GetNumberOfAnalogueSignalDescriptions
 - IntComtrade, [388](#)
 - IntComtradeset, [393](#)
- GetNumberOfClusters
 - SetCluster, [331](#)
- GetNumberOfColumns
 - ElmRes, [122](#)
 - IntComtrade, [388](#)
 - IntComtradeset, [394](#)
 - IntMat, [412](#)
- GetNumberOfCommandsForStudyCase
 - ComTasks, [319](#)
- GetNumberOfDigitalSignalDescriptions
 - IntComtrade, [388](#)
 - IntComtradeset, [394](#)
- GetNumberOfGeneratorEventsForTimeStep
 - ComContingency, [257](#)
- GetNumberOfLoadEventsForTimeStep
 - ComContingency, [257](#)
- GetNumberOfRows
 - ElmRes, [122](#)
 - IntComtrade, [389](#)
 - IntComtradeset, [394](#)
 - IntMat, [413](#)
- GetNumberOfStudyCases
 - ComTasks, [319](#)
- GetNumberOfSwitchEventsForTimeStep
 - ComContingency, [257](#)
- GetNumberOfTimeSteps
 - ComContingency, [258](#)
- GetNumberOfValidationMessages
 - ComCimvalidate, [252](#)
- GetNumberOfViolations
 - ScnFreq, [466](#)
 - ScnFrt, [468](#)
 - ScnSpeed, [469](#)
 - ScnSync, [471](#)
 - ScnVar, [473](#)
 - ScnVolt, [474](#)
- GetNumProcesses
 - SetUser, [358](#)
- GetNumSlave
 - SetParalman, [349](#)
- GetObj
 - ComContingency, [258](#)
 - ElmRes, [123](#)
- GetObject
 - ComCimvalidate, [252](#)
 - ComOutage, [293](#)
 - ElmRes, [123](#)
- GetObjectId
 - ComCimvalidate, [252](#)
- GetObjects
 - IntScenario, [440](#)
 - IntSubset, [450](#)
- GetObjectsReferencingThis
 - CimObject, [370](#)
- GetObjectsWithSameId
 - CimObject, [371](#)
- GetObjectValue
 - ElmRes, [123](#)

- IntComtrade, 389
- IntComtradeset, 394
- GetObjs
 - ElmArea, 76
 - ElmFeeder, 92
 - ElmZone, 163
- GetOMR
 - ComOmr, 291
- GetOperationValue
 - IntScenario, 440
- GetOperator
 - General Object Methods, 59
- GetOrderCode
 - ComUctexp, 328
- GetOrInsertCurvePlot
 - GrpPage, 375
- GetOrInsertDiscreteBarPlot
 - GrpPage, 375
- GetOrInsertPlot
 - SetVipage, 359
- GetOrInsertXYPlot
 - GrpPage, 376
- GetOutputWindow
 - Output Window Functions, 44
- GetOverLoadedBranches
 - ComShc, 305
- GetOverLoadedBuses
 - ComShc, 305
- GetOwner
 - General Object Methods, 59
- GetPage
 - SetDesktop, 341
- GetParameterCount
 - CimModel, 366
 - CimObject, 371
- GetParameterNamespace
 - CimModel, 366
 - CimObject, 371
- GetParameterValue
 - CimModel, 367
 - CimObject, 371
- GetParent
 - General Object Methods, 59
- GetPathFolder
 - SetPath, 352
- GetPathToNearestBusbar
 - StaCubic, 167
- GetPfObjects
 - CimObject, 372
- GetPlot
 - GrpPage, 376
- GetProfile
 - ComCimvalidate, 253
- GetProjectFolder
 - Application Methods, 17
- GetProjectFolderType
 - IntPrjfolder, 435
- GetQlim
 - IntQlim, 437
- GetRandomNumber
 - Mathematical Functions, 38
- GetRandomNumberEx
 - Mathematical Functions, 38
- GetRating
 - IntThrating, 451
- GetRecordingStage
 - Application Methods, 18
- GetReferences
 - General Object Methods, 60
- GetRegion
 - General Object Methods, 60
- GetRegionCount
 - ComOmr, 292
- GetRelCase
 - ElmRes, 124
- GetRemoteBreakers
 - ElmCoup, 88
- GetRowLabel
 - IntMat, 413
- GetRowLabelIndex
 - IntMat, 413
- GetScaleObjX
 - VisHrm, 485
 - VisPlot, 495
 - VisPlot2, 501
- GetScaleObjY
 - VisHrm, 485
 - VisPlot, 495
 - VisPlot2, 501
- GetScenario
 - IntScensched, 443
- GetScheme
 - IntSstage, 447
- GetSepStationAreas
 - ElmTerm, 141
- GetSettings
 - Application Methods, 18
- GetSeverity
 - ComCimvalidate, 253
- GetSignalHeader
 - IntComtrade, 389
 - IntComtradeset, 395
- GetSimulationTime
 - ComSim, 309
- GetSlot
 - ElmRelay, 111
- GetSplit
 - ElmSubstat, 130
 - ElmTrfstat, 156
- GetSplitCal
 - ElmSubstat, 131
 - ElmTrfstat, 157

- GetSplitIndex
 - ElmSubstat, 132
 - ElmTrfstat, 157
- GetStartedDevices
 - ComShctrace, 308
- GetStartEndTime
 - IntScensched, 443
- GetStatus
 - StaExtbrkmea, 168
 - StaExtcmdmea, 173
 - StaExtmdatmea, 179
 - StaExtfmea, 183
 - StaExtfuelmea, 188
 - StaExtimea, 193
 - StaExtpfmea, 198
 - StaExtpmea, 203
 - StaExtqmea, 208
 - StaExtsmea, 213
 - StaExttapmea, 218
 - StaExtv3mea, 223
 - StaExtvmea, 228
- GetStatusTmp
 - StaExtbrkmea, 169
 - StaExtcmdmea, 174
 - StaExtmdatmea, 179
 - StaExtfmea, 184
 - StaExtfuelmea, 189
 - StaExtimea, 194
 - StaExtpfmea, 199
 - StaExtpmea, 204
 - StaExtqmea, 209
 - StaExtsmea, 213
 - StaExttapmea, 219
 - StaExtv3mea, 224
 - StaExtvmea, 229
- GetStepupTransformer
 - ElmAsm, 79
 - ElmAsmsc, 80
 - ElmGenstat, 96
 - ElmStactrl, 129
 - ElmSvs, 134
 - ElmSym, 137
 - ElmXnet, 161
- GetStudyCases
 - ComTasks, 319
- GetStudyTimeObject
 - Date/Time Functions, 32
- GetSubElmRes
 - ElmRes, 124
- GetSuccessfullyConnectedItems
 - ComPrjconnector, 295
- GetSummaryGrid
 - Application Methods, 18
- GetSuppliedElements
 - ElmSubstat, 132
 - ElmTr2, 145
 - ElmTr3, 148
 - ElmTr4, 152
 - ElmTrfstat, 158
- GetSupplyingSubstations
 - General Object Methods, 60
- GetSupplyingTransformers
 - General Object Methods, 61
- GetSupplyingTrfstations
 - General Object Methods, 61
- GetSwitchEvent
 - ComContingency, 258
- GetSwitchStatus
 - IntRunarrange, 438
- GetSwitchTime
 - ComShctrace, 308
- GetSystemGrounding
 - General Object Methods, 61
- GetTableReports
 - Dialogue Boxes Functions, 32
- GetTapPhi
 - ElmTr2, 145
 - ElmTr3, 149
 - ElmTr4, 153
- GetTapRatio
 - ElmTr2, 146
 - ElmTr3, 149
 - ElmTr4, 153
- GetTapZDependentSide
 - ElmTr3, 149
 - ElmTr4, 154
- GetTempDir
 - File System Functions, 31
- GetTemporaryDirectory
 - File System Functions, 31
- GetTieOpenPoint
 - ElmBbone, 83
- GetTimeOfStepInSeconds
 - ComContingency, 259
- GetTitleObject
 - PltLinebarplot, 462
- GetToSigName
 - BlkSig, 365
- GetTotalInterruptedPower
 - ComContingency, 259
- GetTotalWarnA
 - ComSim, 309
- GetTotalWarnB
 - ComSim, 309
- GetTotalWarnC
 - ComSim, 310
- GetTotLength
 - ElmBbone, 83
- GetTransferCalcData
 - ComTransfer, 325
- GetTrippedDevices
 - ComShctrace, 308

- GetType
 - ComCimvalidate, 253
 - ElmLne, 101
- GetUcteNodeName
 - ComUcteexp, 328
- GetUnit
 - ElmRes, 124
 - IntComtrade, 389
 - IntComtradeset, 395
- GetUnom
 - ElmRelay, 111
 - General Object Methods, 62
- GetUnsuccessfullyConnectedItems
 - ComPrjconnector, 295
- GetUserManager
 - Application Methods, 18
- GetUsers
 - IntUserman, 453
- GetValue
 - ElmRes, 125
 - IntComtrade, 390
 - IntComtradeset, 395
 - IntDplmap, 400
 - ScnFreq, 466
 - ScnFrt, 468
 - ScnSpeed, 470
 - ScnSync, 471
 - ScnVar, 473
 - ScnVolt, 475
- GetVar
 - IntMon, 418
- GetVariable
 - ElmRes, 125
 - IntComtrade, 390
 - IntComtradeset, 396
 - ScnFreq, 466
 - ScnFrt, 468
 - ScnSpeed, 470
 - ScnSync, 471
 - ScnVar, 473
 - ScnVolt, 475
- GetVariation
 - IntSstage, 447
- GetVersions
 - IntPrj, 429
- GetVI
 - SetVipage, 359
- GetViolatedElement
 - ScnFreq, 467
 - ScnFrt, 468
 - ScnSpeed, 470
 - ScnSync, 472
 - ScnVar, 473
 - ScnVolt, 475
- GetViolatedScanModules
 - ComSim, 310
- GetViolationTime
 - ScnFreq, 467
 - ScnFrt, 469
 - ScnSpeed, 470
 - ScnSync, 472
 - ScnVar, 474
 - ScnVolt, 475
- GetWorkingDir
 - File System Functions, 31
- GetWorkspaceDirectory
 - File System Functions, 31
- GetY0m
 - ElmLne, 101
- GetY1m
 - ElmLne, 101
- GetZ0m
 - ElmLne, 102
- GetZ0pu
 - ElmTr2, 146
 - ElmTr3, 150
 - ElmTr4, 154
- GetZ1m
 - ElmLne, 102
- GetZeroImpedance
 - General Object Methods, 62
- GetZeroSequenceHVLVT
 - TypTr2, 479
- GetZmatDist
 - ElmLne, 103
- GetZpu
 - ElmTr2, 146
 - ElmTr3, 150
 - ElmTr4, 155
 - ElmVltreg, 160
- GrpPage, 374
 - DoAutoScale, 375
 - DoAutoScaleX, 375
 - DoAutoScaleY, 375
 - GetOrInsertCurvePlot, 375
 - GetOrInsertDiscreteBarPlot, 375
 - GetOrInsertXYPlot, 376
 - GetPlot, 376
 - RemovePage, 376
 - SetAutoScaleModeX, 377
 - SetAutoScaleModeY, 377
 - SetLayoutMode, 377
 - SetResults, 377
 - SetScaleTypeX, 378
 - SetScaleTypeY, 378
 - SetScaleX, 378
 - SetScaleY, 378
 - Show, 379
- HasAttribute
 - General Object Methods, 62
- HasCreatedCalBus

- ElmTerm, [142](#)
- HasExternalReferences
 - IntPrj, [429](#)
- HasGnrlMod
 - ElmBbone, [83](#)
- HasParameter
 - CimModel, [367](#)
 - CimObject, [372](#)
- HasResults
 - General Object Methods, [63](#)
- HasResultsForDirectionalBackup
 - ComCoordreport, [261](#)
- HasResultsForNonDirectionalBackup
 - ComCoordreport, [261](#)
- HasResultsForOverreach
 - ComCoordreport, [261](#)
- HasResultsForZone
 - ComCoordreport, [262](#)
- HasRoutes
 - ElmLne, [103](#)
- HasRoutesOrSec
 - ElmLne, [103](#)
- Hide
 - Application Methods, [19](#)
- Import
 - IntDocument, [398](#)
 - IntGrfgroup, [406](#)
 - IntGrflayer, [407](#)
 - IntIcon, [410](#)
- ImportAndConvert
 - ComCimdbimp, [250](#)
- ImportDz
 - Application Methods, [19](#)
- ImportFromVec
 - IntGrflayer, [408](#)
- ImportSnapshot
 - Application Methods, [19](#)
- IndexOf
 - IntDplvec, [402](#)
- Info
 - ElmStactrl, [129](#)
- Init
 - ElmRes, [125](#)
 - IntMat, [414](#)
 - IntVec, [454](#)
- InitialiseWriting
 - ElmRes, [125](#)
- InitQuickAccess
 - ComUctexp, [328](#)
- InitTmp
 - StaExtbrkmea, [169](#)
 - StaExtcmdmea, [174](#)
 - StaExtmdatmea, [179](#)
 - StaExtfmea, [184](#)
 - StaExtfuelmea, [189](#)
 - StaExttimea, [194](#)
 - StaExtpfmea, [199](#)
 - StaExtpmea, [204](#)
 - StaExtqmea, [209](#)
 - StaExtsmea, [213](#)
 - StaExttapmea, [219](#)
 - StaExtv3mea, [224](#)
 - StaExtvmea, [229](#)
- Insert
 - IntDplmap, [400](#)
 - IntDplvec, [403](#)
- InsertPlot
 - SetVipage, [359](#)
- IntAddonvars, [379](#)
 - AddDouble, [379](#)
 - AddDoubleMatrix, [379](#)
 - AddDoubleVector, [380](#)
 - AddInteger, [381](#)
 - AddIntegerVector, [381](#)
 - AddObject, [381](#)
 - AddObjectVector, [382](#)
 - AddString, [382](#)
 - RemoveParameter, [383](#)
- IntCase, [383](#)
 - Activate, [383](#)
 - ApplyNetworkState, [383](#)
 - ApplyStudyTime, [384](#)
 - Consolidate, [384](#)
 - Deactivate, [385](#)
 - SetStudyTime, [385](#)
- IntComtrade, [385](#)
 - ConvertToASCIIFormat, [385](#)
 - ConvertToBinaryFormat, [386](#)
 - FindColumn, [386](#)
 - FindMaxInColumn, [386](#)
 - FindMinInColumn, [387](#)
 - GetAnalogueDescriptions, [387](#)
 - GetDescription, [387](#)
 - GetDigitalDescriptions, [388](#)
 - GetNumberOfAnalogueSignalDescriptions, [388](#)
 - GetNumberOfColumns, [388](#)
 - GetNumberOfDigitalSignalDescriptions, [388](#)
 - GetNumberOfRows, [389](#)
 - GetObjectValue, [389](#)
 - GetSignalHeader, [389](#)
 - GetUnit, [389](#)
 - GetValue, [390](#)
 - GetVariable, [390](#)
 - Load, [391](#)
 - NCol, [388](#)
 - NRow, [389](#)
 - Release, [391](#)
 - SizeX, [389](#)
 - SizeY, [388](#)
 - SortAccordingToColumn, [391](#)

- IntComtradeset, 391
 - FindColumn, 392
 - FindMaxInColumn, 392
 - FindMinInColumn, 392
 - GetAnalogueDescriptions, 393
 - GetDescription, 393
 - GetDigitalDescriptions, 393
 - GetNumberOfAnalogueSignalDescriptions, 393
 - GetNumberOfColumns, 394
 - GetNumberOfDigitalSignalDescriptions, 394
 - GetNumberOfRows, 394
 - GetObjectValue, 394
 - GetSignalHeader, 395
 - GetUnit, 395
 - GetValue, 395
 - GetVariable, 396
 - Load, 396
 - NCol, 394
 - NRow, 394
 - Release, 396
 - SizeX, 394
 - SizeY, 394
 - SortAccordingToColumn, 396
- IntDataset, 397
 - AddRef, 397
 - All, 397
 - Clear, 397
 - GetAll, 397
- IntDocument, 398
 - Export, 398
 - Import, 398
 - Reset, 398
 - View, 399
- IntDplmap, 399
 - Clear, 399
 - Contains, 399
 - First, 399
 - GetValue, 400
 - Insert, 400
 - Next, 401
 - Remove, 401
 - Size, 401
 - Update, 402
- IntDplvec, 402
 - Clear, 402
 - Get, 402
 - IndexOf, 402
 - Insert, 403
 - Remove, 403
 - Size, 403
 - Sort, 403
- IntEvt, 404
 - CreateCBEvents, 404
 - RemoveSwitchEvents, 404
- IntExtaccess, 405
 - CheckUrl, 405
- IntGate, 405
 - AddTrigger, 405
- IntGrf, 405
 - MoveToLayer, 406
- IntGrfgroup, 406
 - ClearData, 406
 - Export, 406
 - Import, 406
- IntGrflayer, 407
 - ClearData, 407
 - Export, 407
 - ExportToVec, 407
 - Import, 407
 - ImportFromVec, 408
- IntGrfnet, 408
 - Close, 408
 - SetLayerVisibility, 408
 - SetSymbolComponentVisibility, 408
 - Show, 409
- IntIcon, 409
 - Export, 409
 - Import, 410
- IntLibrary, 410
 - Activate, 410
 - Deactivate, 410
- IntMat, 411
 - ColLbl, 411, 412
 - Get, 411
 - GetColumnLabel, 412
 - GetColumnLabelIndex, 412
 - GetNumberOfColumns, 412
 - GetNumberOfRows, 413
 - GetRowLabel, 413
 - GetRowLabelIndex, 413
 - Init, 414
 - Invert, 414
 - Multiply, 414
 - NCol, 412
 - NRow, 413
 - Resize, 415
 - RowLbl, 413, 415
 - Save, 415
 - Set, 416
 - SetColumnLabel, 416
 - SetRowLabel, 416
 - SizeX, 413
 - SizeY, 412
 - SortToColumn, 417
 - SortToColumn, 417
- IntMon, 417
 - AddVar, 418
 - AddVars, 418
 - ClearVars, 418
 - GetVar, 418
 - NVars, 418

- PrintAllVal, 419
- PrintVal, 419
- RemoveVar, 419
- IntOutage, 419
 - Apply, 419
 - ApplyAll, 420
 - Check, 420
 - CheckAll, 420
 - IsInStudyTime, 421
 - IsInStudytime, 421
 - ResetAll, 421
- IntPlannedout, 421
 - SetRecurrence, 421
- IntPlot, 422
 - SetAdaptY, 422
 - SetAutoScaleY, 422
 - SetScaleY, 423
- IntPrj, 423
 - Activate, 424
 - AddProjectToCombined, 424
 - AddProjectToRemoteDatabase, 424
 - Archive, 424
 - BeginDataExtensionModification, 425
 - CanAddProjectToRemoteDatabase, 425
 - CanSubscribeProjectReadOnly, 425
 - CanSubscribeProjectReadWrite, 425
 - ClearInvalidReferences, 426
 - CopyDataExtensionFrom, 426
 - CreateVersion, 426
 - Deactivate, 427
 - EndDataExtensionModification, 427
 - GetDerivedProjects, 428
 - GetExternalReferences, 428
 - GetGeoCoordinateSystem, 428
 - GetLatestVersion, 429
 - GetVersions, 429
 - HasExternalReferences, 429
 - LoadData, 430
 - MergeToBaseProject, 430
 - Migrate, 430
 - NormaliseCombined, 431
 - PackExternalReferences, 431
 - Purge, 431
 - RemoveProjectFromCombined, 431
 - Restore, 432
 - SetGeoCoordinateSystem, 432
 - SubscribeProjectReadOnly, 432
 - SubscribeProjectReadWrite, 432
 - TransformGeoCoordinates, 432
 - UnsubscribeProject, 433
 - UpdateStatistics, 433
 - UpdateToDefaultStructure, 433
 - UpdateToMostRecentBaseVersion, 434
- IntPrjfolder, 435
 - GetProjectFolderType, 435
 - IsProjectFolderType, 436
- IntQlim, 436
 - GetQlim, 437
- IntRas, 437
 - AddEvent, 437
 - AddTrigger, 437
 - IsValid, 438
- IntRunarrange, 438
 - GetSwitchStatus, 438
- IntScenario, 438
 - Activate, 439
 - Apply, 439
 - ApplySelective, 439
 - Deactivate, 440
 - DiscardChanges, 440
 - GetObjects, 440
 - GetOperationValue, 440
 - ReleaseMemory, 441
 - Save, 441
 - SetOperationValue, 441
- IntScensched, 442
 - Activate, 442
 - Deactivate, 442
 - DeleteRow, 442
 - GetScenario, 443
 - GetStartEndTime, 443
 - SearchScenario, 443
- IntScheme, 444
 - Activate, 444
 - Consolidate, 444
 - Deactivate, 444
 - GetActiveScheduler, 445
 - NewStage, 445
- IntSscheduler, 445
 - Activate, 445
 - Deactivate, 446
 - Update, 446
- IntSstage, 446
 - Activate, 446
 - CreateStageObject, 447
 - EnableDiffMode, 447
 - GetScheme, 447
 - GetVariation, 447
 - IsExcluded, 447
 - PrintModifications, 448
 - ReadValue, 448
 - WriteValue, 448
- IntSubset, 449
 - Apply, 449
 - ApplySelective, 449
 - Clear, 450
 - GetConfiguration, 450
 - GetObjects, 450
- IntThrating, 450
 - GetCriticalTimePhase, 450
 - GetRating, 451
- IntUrl, 451

- View, 451
- IntUser, 451
 - Purge, 452
 - SetPassword, 452
 - TerminateSession, 452
- IntUserman, 452
 - CreateGroup, 452
 - CreateUser, 453
 - GetGroups, 453
 - GetUsers, 453
 - UpdateGroups, 453
- IntVec, 454
 - Get, 454
 - Init, 454
 - Max, 454
 - Mean, 454
 - Min, 455
 - Resize, 455
 - Save, 455
 - Set, 455
 - Size, 456
 - Sort, 456
- IntVecobj, 456
 - Get, 456
 - Resize, 456
 - Save, 457
 - Search, 457
 - Set, 457
 - Size, 457
- IntVersion, 458
 - CreateDerivedProject, 458
 - GetDerivedProjects, 458
 - GetHistoricalProject, 458
 - Rollback, 458
- IntViewbookmark, 459
 - JumpTo, 459
 - UpdateFromCurrentView, 459
- Invert
 - IntMat, 414
- InvertMatrix
 - Mathematical Functions, 39
- IsAC
 - ComLdf, 281
- IsAdditionalResultsFlagSetForCommand
 - ComTasks, 320
- IsAttributeModelInternal
 - Application Methods, 20
- IsAutomaticCalculationResetEnabled
 - Environment Functions, 35
- IsBalanced
 - ComLdf, 281
- IsBreaker
 - ElmCoup, 88
- IsCable
 - ElmLne, 103
 - ElmLnsec, 105
- TypLne, 477
- IsCalcRelevant
 - General Object Methods, 63
- IsClosed
 - ElmCoup, 89
 - ElmGndswt, 98
 - StaCubic, 167
 - StaSwitch, 233
- IsCommandIgnored
 - ComTasks, 320
- IsConnected
 - ElmGenstat, 96
 - ElmPvsys, 109
 - ElmSym, 137
 - StaCubic, 167
- IsDC
 - ComLdf, 281
- IsDeleted
 - General Object Methods, 63
- IsEarthed
 - General Object Methods, 64
- IsElectrEquivalent
 - ElmTerm, 142
- IsEncrypted
 - ComDpl, 273
 - TypQdsl, 478
- IsEnergized
 - General Object Methods, 64
- IsEquivalent
 - ElmTerm, 143
- IsExcluded
 - IntSstage, 447
- IsFinalEchoOnEnabled
 - Environment Functions, 35
- IsFrozen
 - SetDesktop, 342
- IsHidden
 - General Object Methods, 64
- IsInFeeder
 - General Object Methods, 64
- IsInStudyTime
 - IntOutage, 421
- IsInStudytime
 - IntOutage, 421
- IsInternalNodeInStation
 - ElmTerm, 143
- IsLastIterationFeasible
 - ComTransfer, 325
- IsLdfValid
 - Application Methods, 20
- IsNetCoupling
 - ElmLne, 104
- IsNetworkDataFolder
 - General Object Methods, 65
- IsNode
 - General Object Methods, 66

- IsObjectActive
 - General Object Methods, 66
- IsObjectModifiedByVariation
 - General Object Methods, 66
- Isolate
 - General Object Methods, 67
- IsOpen
 - ElmCoup, 89
 - ElmGndswt, 98
 - StaSwitch, 233
- IsOpened
 - SetDesktop, 342
- IsOutOfService
 - General Object Methods, 67
- IsPQ
 - ElmAsm, 79
- IsProjectFolderType
 - IntPrjfolder, 436
- IsQuadBooster
 - ElmTr2, 147
 - ElmTr3, 151
 - ElmTr4, 155
- IsReducible
 - General Object Methods, 67
- IsRmsValid
 - Application Methods, 20
- IsScenarioAttribute
 - Application Methods, 20
- IsShcValid
 - Application Methods, 21
- IsShortCircuited
 - General Object Methods, 68
- IsSimValid
 - Application Methods, 21
- IsSplitting
 - ElmBoundary, 85
- IsStarted
 - ElmRelay, 111
- IsStatusBitSet
 - StaExtbrkmea, 169
 - StaExtcmdmea, 174
 - StaExtmdatmea, 179
 - StaExtfmea, 184
 - StaExtfuelmea, 189
 - StaExtimea, 194
 - StaExtpfmea, 199
 - StaExtpmea, 204
 - StaExtqmea, 209
 - StaExtsmea, 214
 - StaExttapmea, 219
 - StaExtv3mea, 224
 - StaExtvmea, 229
- IsStatusBitSetTmp
 - StaExtbrkmea, 169
 - StaExtcmdmea, 174
 - StaExtmdatmea, 179
- StaExtfmea, 184
- StaExtfuelmea, 189
- StaExtimea, 194
- StaExtpfmea, 199
- StaExtpmea, 204
- StaExtqmea, 209
- StaExtsmea, 214
- StaExttapmea, 219
- StaExtv3mea, 224
- StaExtvmea, 229
- IsStudyCaseIgnored
 - ComTasks, 321
- IsValid
 - IntRas, 438
- IsWriteCacheEnabled
 - Application Methods, 21
- JumpTo
 - IntViewbookmark, 459
- JumpToLastStep
 - ComContingency, 259
- Load
 - ElmRes, 126
 - IntComtrade, 391
 - IntComtradeset, 396
- LoadData
 - IntPrj, 430
- LoadFile
 - ElmFile, 93
- LoadMicroSCADAFile
 - ComLink, 282
- LoadProfile
 - Application Methods, 21
- LoadSimulationState
 - ComSim, 310
- LoadSnapshot
 - ComSim, 310
- LossCostAtBusTech
 - ComCapo, 248
- Mark
 - SetLevelvis, 348
- MarkInGraphics
 - Application Methods, 22
 - General Object Methods, 68
- MarkRegions
 - ComSimoutage, 315
- Mathematical Functions, 38
 - GetRandomNumber, 38
 - GetRandomNumberEx, 38
 - InvertMatrix, 39
 - RndExp, 40
 - RndGetMethod, 40
 - RndGetSeed, 41
 - RndNormal, 41
 - RndSetup, 41

- RndUnifInt, [42](#)
- RndUnifReal, [43](#)
- RndWeibull, [43](#)
- SetRandomSeed, [43](#)
- Max
 - IntVec, [454](#)
- MaxZoneNumberFor
 - ComCoordreport, [262](#)
- Mean
 - IntVec, [454](#)
- MeasureLength
 - ElmLne, [104](#)
- Merge
 - ComMerge, [287](#)
- MergeToBaseProject
 - IntPrj, [430](#)
- Migrate
 - IntPrj, [430](#)
- MigratePage
 - SetVipage, [359](#)
- Min
 - IntVec, [455](#)
- Module Functions, [2](#)
 - __version__, [2](#)
 - GetApplication, [2](#)
 - GetApplicationExt, [3](#)
- ModuleExists
 - ComAddon, [245](#)
- Move
 - General Object Methods, [68](#)
- MoveToLayer
 - IntGrf, [406](#)
- Multiply
 - IntMat, [414](#)
- NCol
 - ElmRes, [122](#)
 - IntComtrade, [388](#)
 - IntComtradeset, [394](#)
 - IntMat, [412](#)
- Network Elements Methods, [74](#)
- NewStage
 - IntScheme, [445](#)
- Next
 - IntDplmap, [401](#)
- NextStepAvailable
 - ComShctrace, [309](#)
- NormaliseCombined
 - IntPrj, [431](#)
- NRow
 - ElmRes, [122](#)
 - IntComtrade, [389](#)
 - IntComtradeset, [394](#)
 - IntMat, [413](#)
- NTap
 - ElmTr2, [147](#)
 - ElmTr3, [151](#)
 - ElmTr4, [155](#)
 - ElmVoltreg, [160](#)
- NVars
 - IntMon, [418](#)
- Object Methods, [48](#)
- Open
 - ElmCou, [89](#)
 - ElmGndswt, [98](#)
 - StaSwitch, [234](#)
- Other Objects Methods, [362](#)
- Output Window Functions, [44](#)
 - ClearOutputWindow, [44](#)
 - GetOutputWindow, [44](#)
 - PrintError, [44](#)
 - PrintInfo, [44](#)
 - PrintPlain, [45](#)
 - PrintWarn, [45](#)
 - SetOutputWindowState, [45](#)
- Output Window Methods, [46](#)
 - Clear, [46](#)
 - GetContent, [46](#)
 - Print, [46](#)
 - Save, [47](#)
 - SetState, [47](#)
- OutputFlexibleData
 - Application Methods, [22](#)
- OverwriteRA
 - ElmSubstat, [133](#)
- OviAlleviate
 - ComRel3, [301](#)
- Pack
 - BlkDef, [363](#)
- PackAsMacro
 - BlkDef, [363](#)
- PackExternalReferences
 - IntPrj, [431](#)
- PasteCopy
 - General Object Methods, [69](#)
- PltDataseries, [459](#)
 - AddCurve, [459](#)
 - AddXYCurve, [460](#)
 - ClearCurves, [460](#)
 - GetDataSource, [460](#)
 - GetIntCalces, [460](#)
- PltLinebarplot, [461](#)
 - DoAutoScale, [461](#)
 - GetAxisX, [461](#)
 - GetAxisY, [462](#)
 - GetDataSet, [462](#)
 - GetLegend, [462](#)
 - GetTitleObject, [462](#)
 - SetAutoScaleModeX, [462](#)
 - SetAutoScaleModeY, [463](#)

- SetAxisSharingLevelX, 463
- SetAxisSharingLevelY, 463
- SetScaleTypeX, 464
- SetScaleTypeY, 464
- SetScaleX, 464
- SetScaleY, 464
- PostCommand
 - Application Methods, 22
- PrepForUntouchedDelete
 - Application Methods, 23
- Print
 - Output Window Methods, 46
- PrintAllVal
 - IntMon, 419
- PrintCheckResults
 - ComLdf, 282
- PrintComparisonReport
 - ComMerge, 287
- PrintError
 - Output Window Functions, 44
- PrintInfo
 - Output Window Functions, 44
- PrintModifications
 - ComMerge, 287
 - IntSstage, 448
- PrintPlain
 - Output Window Functions, 45
- PrintVal
 - IntMon, 419
- PrintWarn
 - Output Window Functions, 45
- Purge
 - IntPrj, 431
 - IntUser, 452
 - SetTboxconfig, 356
- PurgeUnusedObjects
 - General Object Methods, 69
- QuickAccessAvailable
 - ComUctexp, 329
- ReadValue
 - IntSstage, 448
- Rebuild
 - Application Methods, 23
- ReceiveData
 - ComLink, 283
 - ComOpc, 292
- Reconnect
 - ElmGenstat, 96
 - ElmPvsys, 109
 - ElmSym, 137
 - ElmXnet, 162
- ReductionInMemory
 - ComRed, 300
- Refresh
 - VisMagndiffplt, 488
 - VisOcplot, 489
- Release
 - ElmRes, 126
 - IntComtrade, 391
 - IntComtradeset, 396
- ReleaseMemory
 - IntScenario, 441
- ReloadProfile
 - Application Methods, 23
- RelZpol, 465
 - AssumeCompensationFactor, 465
 - AssumeReRI, 465
 - AssumeXeXI, 465
- Remove
 - IntDplmap, 401
 - IntDplvec, 403
- RemoveAllConfigurations
 - SetDataext, 339
- RemoveAllRas
 - ComSimoutage, 315
- RemoveCmdsForStudyCaseRow
 - ComTasks, 321
- RemoveCommand
 - ComTasks, 321
- RemoveConfiguration
 - SetDataext, 339
- RemoveContingencies
 - ComSimoutage, 315
- RemoveEvents
 - ComContingency, 259
 - ComOutage, 294
 - ComRel3, 301
- RemoveOutages
 - ComRel3, 301
- RemovePage
 - GrpPage, 376
 - SetDesktop, 342
- RemoveParameter
 - CimModel, 367
 - CimObject, 372
 - IntAddonvars, 383
- RemoveProjectFromCombined
 - IntPrj, 431
- RemoveRas
 - ComSimoutage, 315
- RemoveRatio
 - TypCtcore, 477
- RemoveRatioByIndex
 - TypCtcore, 477
- RemoveStudyCase
 - ComTasks, 322
- RemoveStudyCases
 - ComTasks, 322
- RemoveSwitchEvents
 - IntEvt, 404

- RemoveVar
 - IntMon, [419](#)
- ReplaceNonAsciiCharacters
 - General Object Methods, [70](#)
- Report
 - ComDllmanager, [270](#)
- ReportNonAsciiCharacters
 - General Object Methods, [70](#)
- ReportUnusedObjects
 - General Object Methods, [70](#)
- Reset
 - ComMerge, [288](#)
 - ComSimoutage, [315](#)
 - IntDocument, [398](#)
 - SetLevelvis, [349](#)
- ResetAll
 - IntOutage, [421](#)
- ResetCalculation
 - Application Methods, [23](#)
- ResetDerating
 - ElmGenstat, [97](#)
 - ElmPvsys, [109](#)
 - ElmSym, [137](#)
- ResetQuickAccess
 - ComUcteexp, [329](#)
- ResetRA
 - ElmSubstat, [133](#)
- ResetReductionInMemory
 - ComRed, [300](#)
- ResetStatusBit
 - StaExtbrkmea, [169](#)
 - StaExtcmdmea, [175](#)
 - StaExtatmea, [180](#)
 - StaExtfmea, [185](#)
 - StaExtfuelmea, [190](#)
 - StaExtmea, [195](#)
 - StaExtpfmea, [200](#)
 - StaExtpmea, [205](#)
 - StaExtqmea, [210](#)
 - StaExtsmea, [214](#)
 - StaExttapmea, [219](#)
 - StaExtv3mea, [225](#)
 - StaExtvmea, [230](#)
- ResetStatusBitTmp
 - StaExtbrkmea, [170](#)
 - StaExtcmdmea, [175](#)
 - StaExtatmea, [180](#)
 - StaExtfmea, [185](#)
 - StaExtfuelmea, [190](#)
 - StaExtmea, [195](#)
 - StaExtpfmea, [200](#)
 - StaExtpmea, [205](#)
 - StaExtqmea, [210](#)
 - StaExtsmea, [214](#)
 - StaExttapmea, [220](#)
 - StaExtv3mea, [225](#)
- StaExtvmea, [230](#)
- ResetThirdPartyModule
 - BlkDef, [364](#)
 - ComDpl, [274](#)
 - TypQdsl, [479](#)
- ResGetData
 - Application Methods, [23](#)
- ResGetDescription
 - Application Methods, [24](#)
- ResGetFirstValidObject
 - Application Methods, [24](#)
- ResGetFirstValidObjectVariable
 - Application Methods, [24](#)
- ResGetFirstValidVariable
 - Application Methods, [24](#)
- ResGetIndex
 - Application Methods, [24](#)
- ResGetMax
 - Application Methods, [25](#)
- ResGetMin
 - Application Methods, [25](#)
- ResGetNextValidObject
 - Application Methods, [25](#)
- ResGetNextValidObjectVariable
 - Application Methods, [25](#)
- ResGetNextValidVariable
 - Application Methods, [25](#)
- ResGetObject
 - Application Methods, [25](#)
- ResGetUnit
 - Application Methods, [26](#)
- ResGetValueCount
 - Application Methods, [26](#)
- ResGetVariable
 - Application Methods, [26](#)
- ResGetVariableCount
 - Application Methods, [26](#)
- Resize
 - ElmBoundary, [85](#)
 - IntMat, [415](#)
 - IntVec, [455](#)
 - IntVecobj, [456](#)
- ResLoadData
 - Application Methods, [26](#)
- ResReleaseData
 - Application Methods, [26](#)
- ResSortToVariable
 - Application Methods, [26](#)
- Restore
 - IntPrj, [432](#)
- ResultForDirectionalBackupVariable
 - ComCoordreport, [262](#)
- ResultForNonDirectionalBackupVariable
 - ComCoordreport, [263](#)
- ResultForOverreachVariable
 - ComCoordreport, [264](#)

- ResultForZoneVariable
 - ComCoordreport, 264
- RndExp
 - Mathematical Functions, 40
- RndGetMethod
 - Mathematical Functions, 40
- RndGetSeed
 - Mathematical Functions, 41
- RndNormal
 - Mathematical Functions, 41
- RndSetup
 - Mathematical Functions, 41
- RndUnifInt
 - Mathematical Functions, 42
- RndUnifReal
 - Mathematical Functions, 43
- RndWeibull
 - Mathematical Functions, 43
- Rollback
 - IntVersion, 458
- RowLbl
 - IntMat, 413, 415
- Save
 - IntMat, 415
 - IntScenario, 441
 - IntVec, 455
 - IntVecobj, 457
 - Output Window Methods, 47
- SaveAsRA
 - ElmSubstat, 133
- SaveAsScenario
 - Application Methods, 27
- SaveFile
 - ElmFile, 93
- SaveSimulationState
 - ComSim, 310
- SaveSnapshot
 - ComSim, 310
- ScnFreq, 466
 - GetLimit, 466
 - GetNumberOfViolations, 466
 - GetValue, 466
 - GetVariable, 466
 - GetViolatedElement, 467
 - GetViolationTime, 467
- ScnFrt, 467
 - GetLimit, 467
 - GetNumberOfViolations, 468
 - GetValue, 468
 - GetVariable, 468
 - GetViolatedElement, 468
 - GetViolationTime, 469
- ScnSpeed, 469
 - GetLimit, 469
 - GetNumberOfViolations, 469
 - GetValue, 470
 - GetVariable, 470
 - GetViolatedElement, 470
 - GetViolationTime, 470
- ScnSync, 471
 - GetLimit, 471
 - GetNumberOfViolations, 471
 - GetValue, 471
 - GetVariable, 471
 - GetViolatedElement, 472
 - GetViolationTime, 472
- ScnVar, 472
 - GetLimit, 472
 - GetNumberOfViolations, 473
 - GetValue, 473
 - GetVariable, 473
 - GetViolatedElement, 473
 - GetViolationTime, 474
- ScnVolt, 474
 - GetLimit, 474
 - GetNumberOfViolations, 474
 - GetValue, 475
 - GetVariable, 475
 - GetViolatedElement, 475
 - GetViolationTime, 475
- Search
 - IntVecobj, 457
- SearchObject
 - General Object Methods, 70
- SearchObjectByForeignKey
 - Application Methods, 27
- SearchScenario
 - IntScensched, 443
- SelectToolbox
 - Application Methods, 27
- SendData
 - ComLink, 283
 - ComOpc, 292
- SentDataStatus
 - ComLink, 283
- Set
 - IntMat, 416
 - IntVec, 455
 - IntVecobj, 457
- SetActiveModule
 - ComAddon, 246
- SetAdaptX
 - SetDesktop, 343
 - SetVipage, 360
 - VisPath, 490
 - VisPlot, 495
 - VisPlot2, 501
- SetAdaptY
 - IntPlot, 422
 - VisPath, 490
 - VisPlot, 496

- VisPlot2, 501
- SetAdditionalResultsFlagForCommand
 - ComTasks, 322
- SetAsDefault
 - ElmRes, 126
- SetAssociationValue
 - CimModel, 367, 368
 - CimObject, 372, 373
- SetAttribute
 - General Object Methods, 71
- SetAttributeEnumeration
 - CimModel, 368
 - CimObject, 373
- SetAttributeLength
 - General Object Methods, 71
- SetAttributeModelInternal
 - Application Methods, 28
- SetAttributes
 - General Object Methods, 71
- SetAttributeShape
 - General Object Methods, 72
- SetAttributeValue
 - CimModel, 369
 - CimObject, 374
- SetAuthorityUri
 - ComGridtocim, 277
- SetAutoAssignmentForAll
 - ComMerge, 288
- SetAutomaticCalculationResetEnabled
 - Environment Functions, 35
- SetAutoScaleModeX
 - GrpPage, 377
 - PltLinebarplot, 462
- SetAutoScaleModeY
 - GrpPage, 377
 - PltLinebarplot, 463
- SetAutoScaleX
 - SetDesktop, 343
 - SetVipage, 360
 - VisHrm, 485
 - VisPlot, 496
 - VisPlot2, 502
- SetAutoScaleY
 - IntPlot, 422
 - VisHrm, 486
 - VisPlot, 496
 - VisPlot2, 502
- SetAxisSharingLevelX
 - PltLinebarplot, 463
- SetAxisSharingLevelY
 - PltLinebarplot, 463
- SetBatchMode
 - ComUcte, 326
- SetBoundaries
 - ComGridtocim, 278
- SetCluster, 330
 - CalcCluster, 330
 - GetNumberOfClusters, 331
- SetColouring
 - SetColscheme, 332
- SetColscheme, 331
 - CreateFilter, 331
 - SetColouring, 332
 - SetFilter, 337
- SetColumnLabel
 - IntMat, 416
- SetCrvDesc
 - VisHrm, 486
 - VisPlot, 496
 - VisPlot2, 502
- SetCrvDescX
 - VisXyplot, 508
- SetCrvDescY
 - VisXyplot, 508
- SetDataext, 337
 - AddConfiguration, 337
 - GetConfiguration, 338
 - GetConfigurations, 338
 - RemoveAllConfigurations, 339
 - RemoveConfiguration, 339
- SetDefScaleX
 - VisHrm, 486
 - VisPlot, 497
 - VisPlot2, 503
- SetDefScaleY
 - VisHrm, 486
 - VisPlot, 497
 - VisPlot2, 503
- SetDeskpage, 339
 - Close, 339
 - Show, 339
- SetDesktop, 340
 - AddPage, 340
 - Close, 340
 - DoAutoScaleX, 341
 - Freeze, 341
 - GetActivePage, 341
 - GetCanvasSize, 341
 - GetPage, 341
 - IsFrozen, 342
 - IsOpened, 342
 - RemovePage, 342
 - SetAdaptX, 343
 - SetAutoScaleX, 343
 - SetResults, 343
 - SetScaleX, 344
 - SetXVar, 344
 - Show, 344
 - Unfreeze, 345
 - WriteWMF, 345
 - ZoomAll, 345
- SetDetailed

- ElmLne, 104
- SetDisplayedButtons
 - SetTboxconfig, 356
- SetDistrstate, 346
 - CalcCluster, 346
- SetElms
 - StoMaint, 476
- SetEnableUserBreak
 - Environment Functions, 37
- SetExternalObject
 - ComDpl, 274
 - ComPython, 298
- SetFileName
 - ComSvgexport, 316
 - ComSvgimport, 317
- SetFilt, 346
 - Get, 346
- SetFilter
 - SetColscheme, 337
- SetFinalEchoOnEnabled
 - Environment Functions, 36
- SetGeoCoordinateSystem
 - IntPrj, 432
- SetGraphicUpdate
 - Environment Functions, 36
- SetGridSelection
 - ComUctexp, 329
- SetGridsToExport
 - ComGridtocim, 278
- SetGuiUpdateEnabled
 - Environment Functions, 36
- SetIgnoreFlagForCommand
 - ComTasks, 323
- SetIgnoreFlagForStudyCase
 - ComTasks, 324
- SetImpedance
 - ElmRelay, 112
- SetInputParameterDouble
 - ComDpl, 274
 - ComPython, 298
- SetInputParameterInt
 - ComDpl, 275
 - ComPython, 299
- SetInputParameterString
 - ComDpl, 275
 - ComPython, 299
- SetInterfaceVersion
 - Application Methods, 28
- SetLayerVisibility
 - IntGrfnet, 408
- SetLayoutMode
 - GrpPage, 377
- SetLevelvis, 346
 - AdaptWidth, 347
 - Align, 347
 - ChangeFont, 347
- ChangeFrameAndWidth, 347
- ChangeLayer, 347
- ChangeRefPoints, 348
- ChangeWidthVisibilityAndColour, 348
- Mark, 348
- Reset, 349
- SetLimits
 - ComSimoutage, 316
- SetLoadScaleAbsolute
 - ElmZone, 164
- SetMaxI
 - ElmRelay, 113
- SetMaxlearth
 - ElmRelay, 113
- SetMeaValue
 - StaExtbrkmea, 170
 - StaExtcmdmea, 175
 - StaExtatmea, 180
 - StaExtfmea, 185
 - StaExtfuelmea, 190
 - StaExttimea, 195
 - StaExtpfmea, 200
 - StaExtpmea, 205
 - StaExtqmea, 210
 - StaExtsumea, 215
 - StaExttapmea, 220
 - StaExtv3mea, 225
 - StaExtvmea, 230
- SetMinI
 - ElmRelay, 113
- SetMinlearth
 - ElmRelay, 113
- SetNumSlave
 - SetParalman, 349
- SetObj
 - ElmRes, 126
- SetObject
 - ComSvgexport, 316
 - ComSvgimport, 317
- SetObjects
 - ComSvgexport, 317
- SetObjectsToCompare
 - ComMerge, 288
- SetObjs
 - ComOutage, 294
- SetOldDistributeLoadMode
 - ComLdf, 282
- SetOPCReceiveQuality
 - ComLink, 283
- SetOperationValue
 - IntScenario, 441
- SetOutOfService
 - ElmRelay, 113
- SetOutputWindowState
 - Output Window Functions, 45
- SetParalman, 349

- GetNumSlave, 349
- SetNumSlave, 349
- SetTransfType, 349
- SetPassword
 - IntUser, 452
- SetPath, 350
 - AllBreakers, 350
 - AllClosedBreakers, 350
 - AllOpenBreakers, 350
 - AllProtectionDevices, 351
 - Create, 351
 - GetAll, 351
 - GetBranches, 351
 - GetBuses, 352
 - GetPathFolder, 352
- SetPrimaryTap
 - StaCt, 164
- SetProgressBarUpdatesEnabled
 - Environment Functions, 37
- SetRA
 - ElmSubstat, 133
- SetRandomSeed
 - Mathematical Functions, 43
- SetRecurrence
 - IntPlannedout, 421
- SetRescheduleFlag
 - Environment Functions, 36
- SetResults
 - GrpPage, 377
 - SetDesktop, 343
 - SetVipage, 360
- SetResultsFolder
 - ComTasks, 324
- SetRowLabel
 - IntMat, 416
- SetScaleTypeX
 - GrpPage, 378
 - PltLinebarplot, 464
- SetScaleTypeY
 - GrpPage, 378
 - PltLinebarplot, 464
- SetScaleX
 - GrpPage, 378
 - PltLinebarplot, 464
 - SetDesktop, 344
 - SetVipage, 360
 - VisPath, 491
 - VisPlot, 497
 - VisPlot2, 503
- SetScaleY
 - GrpPage, 378
 - IntPlot, 423
 - PltLinebarplot, 464
 - VisBdia, 481
 - VisPath, 491
 - VisPlot, 497
- VisPlot2, 504
- SetSelect, 352
 - AddRef, 352
 - All, 353
 - AllAsm, 353
 - AllBars, 353
 - AllBreakers, 353
 - AllClosedBreakers, 353
 - AllElm, 354
 - AllLines, 354
 - AllLoads, 354
 - AllOpenBreakers, 354
 - AllSym, 354
 - AllTypLne, 354
 - Clear, 355
 - GetAll, 355
- SetShowAllUsers
 - Application Methods, 28
- SetState
 - Output Window Methods, 47
- SetStatus
 - StaExtbrkmea, 170
 - StaExtcmdmea, 175
 - StaExtatmea, 180
 - StaExtfmea, 185
 - StaExtfuelmea, 190
 - StaExtimea, 195
 - StaExtpfmea, 200
 - StaExtpmea, 205
 - StaExtqmea, 210
 - StaExtsmea, 215
 - StaExttapmea, 220
 - StaExtv3mea, 225
 - StaExtvmea, 230
- SetStatusBit
 - StaExtbrkmea, 171
 - StaExtcmdmea, 176
 - StaExtatmea, 181
 - StaExtfmea, 186
 - StaExtfuelmea, 191
 - StaExtimea, 196
 - StaExtpfmea, 201
 - StaExtpmea, 206
 - StaExtqmea, 211
 - StaExtsmea, 216
 - StaExttapmea, 221
 - StaExtv3mea, 226
 - StaExtvmea, 231
- SetStatusBitTmp
 - StaExtbrkmea, 171
 - StaExtcmdmea, 176
 - StaExtatmea, 181
 - StaExtfmea, 186
 - StaExtfuelmea, 191
 - StaExtimea, 196
 - StaExtpfmea, 201

- StaExtpmea, 206
- StaExtqmea, 211
- StaExtsmea, 216
- StaExttapmea, 221
- StaExtv3mea, 226
- StaExtvmea, 231
- SetStatusTmp
 - StaExtbrkmea, 172
 - StaExtcmdmea, 177
 - StaExtatmea, 182
 - StaExtfmea, 187
 - StaExtfuelmea, 192
 - StaExttimea, 197
 - StaExtpfmea, 202
 - StaExtpmea, 207
 - StaExtqmea, 212
 - StaExtsmea, 216
 - StaExttapmea, 221
 - StaExtv3mea, 227
 - StaExtvmea, 232
- SetStudyTime
 - IntCase, 385
- SetStyle
 - SetVipage, 361
- SetSubElmResKey
 - ElmRes, 127
- SetSwitchShcEventMode
 - ComLink, 284
- SetSymbolComponentVisibility
 - IntGrfnet, 408
- SetTboxconfig, 355
 - Check, 355
 - GetAvailableButtons, 355
 - GetDisplayedButtons, 356
 - Purge, 356
 - SetDisplayedButtons, 356
- SetThirdPartyModule
 - BlkDef, 364
 - ComDpl, 275
 - TypQdsl, 479
- SetTile
 - SetVipage, 361
- SetTime, 356
 - Date, 356
 - ElmRelay, 114
 - SetTime, 357
 - SetTimeUTC, 357
 - Time, 357
- SetTimeUTC
 - SetTime, 357
- Settings Methods, 330
- SetTransfType
 - SetParalman, 349
- SetUser, 357
 - GetNumProcesses, 358
- SetUserBreakEnabled
 - Environment Functions, 37
- SetVipage, 358
 - Close, 358
 - CreateVI, 359
 - DoAutoScaleX, 358
 - DoAutoScaleY, 358
 - GetOrInsertPlot, 359
 - GetVI, 359
 - InsertPlot, 359
 - MigratePage, 359
 - SetAdaptX, 360
 - SetAutoScaleX, 360
 - SetResults, 360
 - SetScaleX, 360
 - SetStyle, 361
 - SetTile, 361
 - SetXVar, 361
 - Show, 362
- SetWriteCacheEnabled
 - Application Methods, 29
- SetXVar
 - SetDesktop, 344
 - SetVipage, 361
 - VisPlot, 498
 - VisPlot2, 504
- SetXVariable
 - VisBdia, 481
- SetYVariable
 - VisBdia, 481
- Show
 - Application Methods, 29
 - GrpPage, 379
 - IntGrfnet, 409
 - SetDesktop, 339
 - SetDesktop, 344
 - SetVipage, 362
- ShowBrowser
 - ComMerge, 288
- ShowEditDialog
 - General Object Methods, 72
- ShowModalBrowser
 - Dialogue Boxes Functions, 33
- ShowModalSelectBrowser
 - Dialogue Boxes Functions, 33
- ShowModalSelectTree
 - General Object Methods, 72
- ShowModelessBrowser
 - Dialogue Boxes Functions, 34
- ShowY2
 - VisPlot2, 504
- Size
 - IntDplmap, 401
 - IntDplvec, 403
 - IntVec, 456
 - IntVecobj, 457
- SizeX

- ElmRes, 122
 - IntComtrade, 389
 - IntComtradeset, 394
 - IntMat, 413
- SizeY
 - ElmRes, 122
 - IntComtrade, 388
 - IntComtradeset, 394
 - IntMat, 412
- slotupd
 - ElmComp, 87
 - ElmRelay, 114
- SlotUpdate
 - ElmComp, 87
 - ElmRelay, 114
- Sort
 - IntDplvec, 403
 - IntVec, 456
- SortAccordingToColumn
 - ElmRes, 127
 - IntComtrade, 391
 - IntComtradeset, 396
- SortToColumn
 - IntMat, 417
- SortToColumn
 - IntMat, 417
- SplitLine
 - Application Methods, 29
- StaCt, 164
 - SetPrimaryTap, 164
- StaCubic, 164
 - GetAll, 165
 - GetBranch, 165
 - GetConnectedMajorNodes, 165
 - GetConnections, 166
 - GetNearestBusbars, 166
 - GetPathToNearestBusbar, 167
 - IsClosed, 167
 - IsConnected, 167
- StaExtbrkmea, 168
 - CopyExtMeaStatusToStatusTmp, 168
 - GetMeaValue, 168
 - GetStatus, 168
 - GetStatusTmp, 169
 - InitTmp, 169
 - IsStatusBitSet, 169
 - IsStatusBitSetTmp, 169
 - ResetStatusBit, 169
 - ResetStatusBitTmp, 170
 - SetMeaValue, 170
 - SetStatus, 170
 - SetStatusBit, 171
 - SetStatusBitTmp, 171
 - SetStatusTmp, 172
 - UpdateControl, 172
 - UpdateCtrl, 172
- StaExtcmdmea, 173
 - CopyExtMeaStatusToStatusTmp, 173
 - GetMeaValue, 173
 - GetStatus, 173
 - GetStatusTmp, 174
 - InitTmp, 174
 - IsStatusBitSet, 174
 - IsStatusBitSetTmp, 174
 - ResetStatusBit, 175
 - ResetStatusBitTmp, 175
 - SetMeaValue, 175
 - SetStatus, 175
 - SetStatusBit, 176
 - SetStatusBitTmp, 176
 - SetStatusTmp, 177
 - UpdateControl, 177
 - UpdateCtrl, 177
- StaExtdatmea, 178
 - CopyExtMeaStatusToStatusTmp, 178
 - CreateEvent, 178
 - GetMeaValue, 178
 - GetStatus, 179
 - GetStatusTmp, 179
 - InitTmp, 179
 - IsStatusBitSet, 179
 - IsStatusBitSetTmp, 179
 - ResetStatusBit, 180
 - ResetStatusBitTmp, 180
 - SetMeaValue, 180
 - SetStatus, 180
 - SetStatusBit, 181
 - SetStatusBitTmp, 181
 - SetStatusTmp, 182
 - UpdateControl, 182
 - UpdateCtrl, 182
- StaExtfmea, 183
 - CopyExtMeaStatusToStatusTmp, 183
 - GetMeaValue, 183
 - GetStatus, 183
 - GetStatusTmp, 184
 - InitTmp, 184
 - IsStatusBitSet, 184
 - IsStatusBitSetTmp, 184
 - ResetStatusBit, 185
 - ResetStatusBitTmp, 185
 - SetMeaValue, 185
 - SetStatus, 185
 - SetStatusBit, 186
 - SetStatusBitTmp, 186
 - SetStatusTmp, 187
 - UpdateControl, 187
 - UpdateCtrl, 187
- StaExtfuelmea, 188
 - CopyExtMeaStatusToStatusTmp, 188
 - GetMeaValue, 188
 - GetStatus, 188

- GetStatusTmp, 189
- InitTmp, 189
- IsStatusBitSet, 189
- IsStatusBitSetTmp, 189
- ResetStatusBit, 190
- ResetStatusBitTmp, 190
- SetMeaValue, 190
- SetStatus, 190
- SetStatusBit, 191
- SetStatusBitTmp, 191
- SetStatusTmp, 192
- UpdateControl, 192
- UpdateCtrl, 192
- StaExtimea, 193
 - CopyExtMeaStatusToStatusTmp, 193
 - GetMeaValue, 193
 - GetStatus, 193
 - GetStatusTmp, 194
 - InitTmp, 194
 - IsStatusBitSet, 194
 - IsStatusBitSetTmp, 194
 - ResetStatusBit, 195
 - ResetStatusBitTmp, 195
 - SetMeaValue, 195
 - SetStatus, 195
 - SetStatusBit, 196
 - SetStatusBitTmp, 196
 - SetStatusTmp, 197
 - UpdateControl, 197
 - UpdateCtrl, 197
- StaExtpfmea, 198
 - CopyExtMeaStatusToStatusTmp, 198
 - GetMeaValue, 198
 - GetStatus, 198
 - GetStatusTmp, 199
 - InitTmp, 199
 - IsStatusBitSet, 199
 - IsStatusBitSetTmp, 199
 - ResetStatusBit, 200
 - ResetStatusBitTmp, 200
 - SetMeaValue, 200
 - SetStatus, 200
 - SetStatusBit, 201
 - SetStatusBitTmp, 201
 - SetStatusTmp, 202
 - UpdateControl, 202
 - UpdateCtrl, 202
- StaExtpmea, 203
 - CopyExtMeaStatusToStatusTmp, 203
 - GetMeaValue, 203
 - GetStatus, 203
 - GetStatusTmp, 204
 - InitTmp, 204
 - IsStatusBitSet, 204
 - IsStatusBitSetTmp, 204
 - ResetStatusBit, 205
- ResetStatusBitTmp, 205
- SetMeaValue, 205
- SetStatus, 205
- SetStatusBit, 206
- SetStatusBitTmp, 206
- SetStatusTmp, 207
- UpdateControl, 207
- UpdateCtrl, 207
- StaExtqmea, 208
 - CopyExtMeaStatusToStatusTmp, 208
 - GetMeaValue, 208
 - GetStatus, 208
 - GetStatusTmp, 209
 - InitTmp, 209
 - IsStatusBitSet, 209
 - IsStatusBitSetTmp, 209
 - ResetStatusBit, 210
 - ResetStatusBitTmp, 210
 - SetMeaValue, 210
 - SetStatus, 210
 - SetStatusBit, 211
 - SetStatusBitTmp, 211
 - SetStatusTmp, 212
 - UpdateControl, 212
 - UpdateCtrl, 212
- StaExtsmea, 213
 - CopyExtMeaStatusToStatusTmp, 213
 - GetStatus, 213
 - GetStatusTmp, 213
 - InitTmp, 213
 - IsStatusBitSet, 214
 - IsStatusBitSetTmp, 214
 - ResetStatusBit, 214
 - ResetStatusBitTmp, 214
 - SetMeaValue, 215
 - SetStatus, 215
 - SetStatusBit, 216
 - SetStatusBitTmp, 216
 - SetStatusTmp, 216
 - UpdateControl, 217
 - UpdateCtrl, 217
- StaExttapmea, 217
 - CopyExtMeaStatusToStatusTmp, 218
 - GetMeaValue, 218
 - GetStatus, 218
 - GetStatusTmp, 219
 - InitTmp, 219
 - IsStatusBitSet, 219
 - IsStatusBitSetTmp, 219
 - ResetStatusBit, 219
 - ResetStatusBitTmp, 220
 - SetMeaValue, 220
 - SetStatus, 220
 - SetStatusBit, 221
 - SetStatusBitTmp, 221
 - SetStatusTmp, 221

- UpdateControl, 222
- UpdateCtrl, 222
- StaExtv3mea, 223
 - CopyExtMeaStatusToStatusTmp, 223
 - GetMeaValue, 223
 - GetStatus, 223
 - GetStatusTmp, 224
 - InitTmp, 224
 - IsStatusBitSet, 224
 - IsStatusBitSetTmp, 224
 - ResetStatusBit, 225
 - ResetStatusBitTmp, 225
 - SetMeaValue, 225
 - SetStatus, 225
 - SetStatusBit, 226
 - SetStatusBitTmp, 226
 - SetStatusTmp, 227
 - UpdateControl, 227
 - UpdateCtrl, 227
- StaExtvmea, 228
 - CopyExtMeaStatusToStatusTmp, 228
 - GetMeaValue, 228
 - GetStatus, 228
 - GetStatusTmp, 229
 - InitTmp, 229
 - IsStatusBitSet, 229
 - IsStatusBitSetTmp, 229
 - ResetStatusBit, 230
 - ResetStatusBitTmp, 230
 - SetMeaValue, 230
 - SetStatus, 230
 - SetStatusBit, 231
 - SetStatusBitTmp, 231
 - SetStatusTmp, 232
 - UpdateControl, 232
 - UpdateCtrl, 232
- StartTrace
 - ComContingency, 260
 - ComOutage, 294
- StaSwitch, 233
 - Close, 233
 - IsClosed, 233
 - IsOpen, 233
 - Open, 234
- StatFileGetXrange
 - Application Methods, 30
- StatFileResetXrange
 - Application Methods, 30
- StatFileSetXrange
 - Application Methods, 30
- Station Elements Methods, 164
- StoMaint, 476
 - SetElms, 476
- StopTrace
 - ComContingency, 260
 - ComOutage, 294
- SubscribeProjectReadOnly
 - IntPrj, 432
- SubscribeProjectReadWrite
 - IntPrj, 432
- SwitchOff
 - General Object Methods, 73
- SwitchOn
 - General Object Methods, 73
- TerminateSession
 - IntUser, 452
- Time
 - SetTime, 357
- TopologyForDirectionalBackupVariable
 - ComCoordreport, 265
- TopologyForNonDirectionalBackupVariable
 - ComCoordreport, 266
- TopologyForOverreachVariable
 - ComCoordreport, 266
- TopologyForZoneVariable
 - ComCoordreport, 266
- TotalLossCost
 - ComCapo, 248
- TransferDirectionalBackupResultsTo
 - ComCoordreport, 267
- TransferNonDirectionalBackupResultsTo
 - ComCoordreport, 267
- TransferOverreachResultsTo
 - ComCoordreport, 268
- TransferResultsTo
 - ComCoordreport, 269
- TransferZoneResultsTo
 - ComCoordreport, 269
- TransformGeoCoordinates
 - IntPrj, 432
- TypAsmo, 476
 - CalcElParams, 476
- TypCtcore, 476
 - AddRatio, 476
 - RemoveRatio, 477
 - RemoveRatioByIndex, 477
- TypLne, 477
 - IsCable, 477
- TypQdsl, 478
 - Encrypt, 478
 - IsEncrypted, 478
 - ResetThirdPartyModule, 479
 - SetThirdPartyModule, 479
- TypTr2, 479
 - GetZeroSequenceHVLVT, 479
- Unfreeze
 - SetDesktop, 345
- UnsubscribeProject
 - IntPrj, 433
- Update

- ChaVecfile, 365
- ComSimoutage, 316
- ElmBmu, 84
- ElmBoundary, 86
- ElmBranch, 86
- ElmCabsys, 87
- IntDplmap, 402
- IntSscheduler, 446
- UpdateControl
 - StaExtbrkmea, 172
 - StaExtcmdmea, 177
 - StaExtdatmea, 182
 - StaExtfmea, 187
 - StaExtfuelmea, 192
 - StaExttimea, 197
 - StaExtpfmea, 202
 - StaExtpmea, 207
 - StaExtqmea, 212
 - StaExtsmea, 217
 - StaExttapmea, 222
 - StaExtv3mea, 227
 - StaExtvmea, 232
- UpdateCtrl
 - StaExtbrkmea, 172
 - StaExtcmdmea, 177
 - StaExtdatmea, 182
 - StaExtfmea, 187
 - StaExtfuelmea, 192
 - StaExttimea, 197
 - StaExtpfmea, 202
 - StaExtpmea, 207
 - StaExtqmea, 212
 - StaExtsmea, 217
 - StaExttapmea, 222
 - StaExtv3mea, 227
 - StaExtvmea, 232
- UpdateFromCurrentView
 - IntViewbookmark, 459
- UpdateGroups
 - IntUserman, 453
- UpdateStatistics
 - IntPrj, 433
- UpdateSubstationTerminals
 - ElmTerm, 143
- UpdateTableReports
 - Dialogue Boxes Functions, 34
- UpdateTablesByCalcPeriod
 - ComTececo, 325
- UpdateToDefaultStructure
 - IntPrj, 433
- UpdateToMostRecentBaseVersion
 - IntPrj, 434
- ValidateConstraints
 - ComRel3, 302
- View
 - IntDocument, 399
 - IntUrl, 451
 - VisBdia, 480
 - AddObjs, 480
 - AddResObjs, 480
 - Clear, 481
 - SetScaleY, 481
 - SetXVariable, 481
 - SetYVariable, 481
 - VisDraw, 482
 - AddRelay, 482
 - AddRelays, 482
 - CentreOrigin, 483
 - Clear, 483
 - DoAutoScaleOnAll, 483
 - DoAutoScaleOnCharacteristics, 483
 - DoAutoScaleOnImpedances, 483
 - DoAutoScaleX, 483
 - DoAutoScaleY, 484
 - VisHrm, 484
 - Clear, 484
 - DoAutoScaleX, 484
 - DoAutoScaleY, 484
 - GetDataSource, 485
 - GetScaleObjX, 485
 - GetScaleObjY, 485
 - SetAutoScaleX, 485
 - SetAutoScaleY, 486
 - SetCrvDesc, 486
 - SetDefScaleX, 486
 - SetDefScaleY, 486
 - VisMagndiffplt, 486
 - AddRelay, 487
 - AddRelays, 487
 - Clear, 487
 - DoAutoScaleX, 487
 - DoAutoScaleY, 488
 - Refresh, 488
 - VisOcplot, 488
 - AddRelay, 488
 - AddRelays, 489
 - Clear, 489
 - DoAutoScaleX, 489
 - DoAutoScaleY, 489
 - Refresh, 489
 - VisPath, 490
 - Clear, 490
 - DoAutoScaleX, 490
 - DoAutoScaleY, 490
 - SetAdaptX, 490
 - SetAdaptY, 490
 - SetScaleX, 491
 - SetScaleY, 491
 - VisPcompdiffplt, 491
 - AddRelay, 492
 - AddRelays, 492

- CentreOrigin, 492
- Clear, 492
- DoAutoScaleX, 492
- DoAutoScaleY, 493
- VisPlot2, 498
 - AddResVars, 499
 - AddVars, 499
 - Clear, 499
 - DoAutoScaleX, 500
 - DoAutoScaleY, 500
 - DoAutoScaleY2, 500
 - GetDataSource, 500
 - GetScaleObjX, 501
 - GetScaleObjY, 501
 - SetAdaptX, 501
 - SetAdaptY, 501
 - SetAutoScaleX, 502
 - SetAutoScaleY, 502
 - SetCrvDesc, 502
 - SetDefScaleX, 503
 - SetDefScaleY, 503
 - SetScaleX, 503
 - SetScaleY, 504
 - SetXVar, 504
 - ShowY2, 504
- VisPlot, 493
 - AddResVars, 493
 - AddVars, 493
 - Clear, 494
 - DoAutoScaleX, 494
 - DoAutoScaleY, 494
 - GetDataSource, 494
 - GetIntCalcres, 495
 - GetScaleObjX, 495
 - GetScaleObjY, 495
 - SetAdaptX, 495
 - SetAdaptY, 496
 - SetAutoScaleX, 496
 - SetAutoScaleY, 496
 - SetCrvDesc, 496
 - SetDefScaleX, 497
 - SetDefScaleY, 497
 - SetScaleX, 497
 - SetScaleY, 497
 - SetXVar, 498
- VisPlottz, 505
 - AddRelay, 505
 - AddRelays, 505
 - Clear, 505
 - DoAutoScaleX, 506
 - DoAutoScaleY, 506
- VisVec, 506
 - CentreOrigin, 506
- VisVecres, 506
 - CentreOrigin, 506
- VisXyplot, 507
 - Clear, 507
 - DoAutoScaleX, 507
 - DoAutoScaleY, 507
 - GetDataSource, 507
 - SetCrvDescX, 508
 - SetCrvDescY, 508
- WereModificationsFound
 - ComMerge, 289
- Write
 - ElmRes, 127
- WriteChangesToDb
 - Application Methods, 30
 - General Object Methods, 74
- WriteDraw
 - ElmRes, 127
- WriteValue
 - IntSstage, 448
- WriteWMF
 - SetDesktop, 345
- ZeroDerivative
 - ComInc, 279
- ZoomAll
 - SetDesktop, 345