OpenDSS-X Developer Guide

C++ Developer Guide

OpenDSS-X Version 1.0

**Prepared by**

Davis Montenegro

Electric Power Research Institute, Inc.

1325 G St. NW #530

Washington, DC 20005

*Last update 08-26-2022*

Table of Contents

[1 Purpose 1](#_Toc112409149)

[2 OpenDSS-X Repository 1](#_Toc112409150)

[3 OpenDSS-X Classes, objects, and hierarchy 4](#_Toc112409151)

[3.1 Power Conversion Elements 4](#_Toc112409152)

[3.1.1 TDSSClass 5](#_Toc112409153)

[3.1.2 TDSSCktElementClass 7](#_Toc112409154)

[3.1.3 TPCClass 7](#_Toc112409155)

[3.1.4 Specific classes 7](#_Toc112409156)

[3.1.5 TNamedObject 9](#_Toc112409157)

[3.1.6 TDSSObject 10](#_Toc112409158)

[3.1.7 TDSSCktElement 10](#_Toc112409159)

[3.1.8 TPCElement 13](#_Toc112409160)

[3.1.9 TLoadObj 14](#_Toc112409161)

[3.1.10 TISourceObj 17](#_Toc112409162)

[3.1.11 TVSourceObj 18](#_Toc112409163)

[3.1.12 TIndMach012Obj 18](#_Toc112409164)

[3.1.13 TGICSourceObj 20](#_Toc112409165)

[3.1.14 TGICLineObj 21](#_Toc112409166)

[3.1.15 TGeneratorObj 21](#_Toc112409167)

[3.1.16 TPVsystemObj 24](#_Toc112409168)

[3.1.17 TStorageObj 27](#_Toc112409169)

[3.1.18 TUPFCObj 30](#_Toc112409170)

[3.1.19 TVCCSObj 30](#_Toc112409171)

[3.1.20 TVSConverterObj 31](#_Toc112409172)

[3.2 Power Delivery Elements 31](#_Toc112409173)

[3.2.1 TPDClass 32](#_Toc112409174)

[3.2.2 TLine 32](#_Toc112409175)

[3.2.3 TAutoTrans 33](#_Toc112409176)

[3.2.4 TCapacitor 33](#_Toc112409177)

[3.2.5 TFault 33](#_Toc112409178)

[3.2.6 TGICTransformer 34](#_Toc112409179)

[3.2.7 TReactor 34](#_Toc112409180)

[3.2.8 TTransf 34](#_Toc112409181)

[3.2.9 TPDElement 35](#_Toc112409182)

[3.2.10 TLineObj 36](#_Toc112409183)

[3.2.11 TAutoTransObj 37](#_Toc112409184)

[3.2.12 TCapacitorObj 38](#_Toc112409185)

[3.2.13 TFaultObj 39](#_Toc112409186)

[3.2.14 TGICTransformerObj 40](#_Toc112409187)

[3.2.15 TReactorObj 40](#_Toc112409188)

[3.2.16 TTransfObj 40](#_Toc112409189)

[3.3 Controls 41](#_Toc112409190)

[3.3.1 TControlClass 42](#_Toc112409191)

[3.3.2 TCapControl 42](#_Toc112409192)

[3.3.3 TESPVLControl 43](#_Toc112409193)

[3.3.4 TExpControl 43](#_Toc112409194)

[3.3.5 TGenDispatcher 43](#_Toc112409195)

[3.3.6 TInvControl 43](#_Toc112409196)

[3.3.7 TRecloser 44](#_Toc112409197)

[3.3.8 TRegControl 44](#_Toc112409198)

[3.3.9 TRelay 44](#_Toc112409199)

[3.3.10 TStorageController 44](#_Toc112409200)

[3.3.11 TSwtControl 44](#_Toc112409201)

[3.3.12 TUPFCControl 45](#_Toc112409202)

[3.3.13 TFuse 45](#_Toc112409203)

[3.3.14 TControlElem 46](#_Toc112409204)

[3.3.15 TCapControlObj 47](#_Toc112409205)

[3.3.16 TESPVLControlObj 47](#_Toc112409206)

[3.3.17 TExpControlObj 48](#_Toc112409207)

[3.3.18 TGenDispatcherObj 48](#_Toc112409208)

[3.3.19 TInvControlObj 49](#_Toc112409209)

[3.3.20 TRecloserObj 51](#_Toc112409210)

[3.3.21 TRegControlObj 52](#_Toc112409211)

[3.3.22 TRelayObj 53](#_Toc112409212)

[3.3.23 TStorageControllerObj 54](#_Toc112409213)

[3.3.24 TSwtControlObj 55](#_Toc112409214)

[3.3.25 TUPFCControlObj 56](#_Toc112409215)

[3.3.26 TFuseObj 57](#_Toc112409216)

[3.4 General-purpose objects (libraries) 57](#_Toc112409217)

[3.4.1 TConductorData 58](#_Toc112409218)

[3.4.2 TCableData 58](#_Toc112409219)

[3.4.3 TCNData 59](#_Toc112409220)

[3.4.4 TTSData 59](#_Toc112409221)

[3.4.5 TWireData 59](#_Toc112409222)

[3.4.6 TGrowthShape 59](#_Toc112409223)

[3.4.7 TLoadShape 60](#_Toc112409224)

[3.4.8 TPriceShape 60](#_Toc112409225)

[3.4.9 TTShape 61](#_Toc112409226)

[3.4.10 TXYcurve 61](#_Toc112409227)

[3.4.11 TTCC\_Curve 62](#_Toc112409228)

[3.4.12 TXfmrCode 62](#_Toc112409229)

[3.4.13 TSpectrum 62](#_Toc112409230)

[3.4.14 TLineSpacing 63](#_Toc112409231)

[3.4.15 TLineGeometry 63](#_Toc112409232)

[3.4.16 TLineCode 63](#_Toc112409233)

[3.4.17 TDynamicExp 64](#_Toc112409234)

[3.4.18 TConductorDataObj 64](#_Toc112409235)

[3.4.19 TCableDataObj 65](#_Toc112409236)

[3.4.20 TCNDataObj 65](#_Toc112409237)

[3.4.21 TTSDataObj 65](#_Toc112409238)

[3.4.22 TWireDataObj 66](#_Toc112409239)

[3.4.23 TGrowthShapeObj 66](#_Toc112409240)

[3.4.24 TLoadShapeObj 66](#_Toc112409241)

[3.4.25 TPriceShapeObj 67](#_Toc112409242)

[3.4.26 TTShapeObj 67](#_Toc112409243)

[3.4.27 TXYcurveObj 68](#_Toc112409244)

[3.4.28 TTCC\_CurveObj 69](#_Toc112409245)

[3.4.29 TXfmrCodeObj 69](#_Toc112409246)

[3.4.30 TSpectrumObj 69](#_Toc112409247)

[3.4.31 TLineSpacingObj 70](#_Toc112409248)

[3.4.32 TLineGeometryObj 70](#_Toc112409249)

[3.4.33 TLineCodeObj 71](#_Toc112409250)

[3.4.34 TDynamicExpObj 72](#_Toc112409251)

[3.5 Meters 72](#_Toc112409252)

[3.5.1 TMeterClass 73](#_Toc112409253)

[3.5.2 TEnergyMeter 74](#_Toc112409254)

[3.5.3 TDSSFMonitor 75](#_Toc112409255)

[3.5.4 TDSSMonitor 75](#_Toc112409256)

[3.5.5 TSensor 76](#_Toc112409257)

[3.5.6 TMeterElement 76](#_Toc112409258)

[3.5.7 TEnergyMeterObj 77](#_Toc112409259)

[3.5.8 TFMonitorObj 78](#_Toc112409260)

[3.5.9 *TMonitorObj* 79](#_Toc112409261)

[3.5.10 TSensorObj 80](#_Toc112409262)

[3.6 Containers 81](#_Toc112409263)

[3.6.1 TDSSSolution 81](#_Toc112409264)

[3.6.2 TSolutionObj 82](#_Toc112409265)

[3.6.3 TDSSCircuit 85](#_Toc112409266)

[3.7 Other objects 87](#_Toc112409267)

[4 Support routines and global context 88](#_Toc112409268)

[5 The solution algorithm 89](#_Toc112409269)

[5.1 Control Actions 90](#_Toc112409270)

[6 References 91](#_Toc112409271)

[7 Acronyms 92](#_Toc112409272)

List of Figures

[Figure 1. Snapshot of the OpenDSS-X copy obtained from the repository 2](#_Toc112409273)

[Figure 2. PCE class inheritance tree 5](#_Toc112409274)

[Figure 3. PCE object inheritance tree 9](#_Toc112409275)

[Figure 4. PDE class inheritance tree 32](#_Toc112409276)

[Figure 5. PDE object inheritance tree 35](#_Toc112409277)

[Figure 6. Control class inheritance tree 42](#_Toc112409278)

[Figure 7. Control object inheritance tree 46](#_Toc112409279)

[Figure 8. general-purpose classes inheritance tree 58](#_Toc112409280)

[Figure 9. General-purpose objects inheritance tree 64](#_Toc112409281)

[Figure 10. Meter classe inheritance tree. 73](#_Toc112409282)

[Figure 11. Meter object inheritance tree. 76](#_Toc112409283)

[Figure 12. Containers class inheritance tree 81](#_Toc112409284)

[Figure 13. Containers object inheritance tree 82](#_Toc112409285)

[Figure 14. Generalized solution algorithm in OpenDSS-X 91](#_Toc112409286)

List of Tables

[Table 1. IDEs supported by OpenDSS-X 1](#_Toc112409287)

[Table 2. OpenDSS-X Compiler Requirements 1](#_Toc112409288)

[Table 3. Contents of OpenDSS-X software repository 2](#_Toc112409289)

# Purpose

This guide presents the technical details of the C++ deployment of OpenDSS (henceforth referred to as OpenDSS-X) for the Microsoft Windows 10 and Red Hat Enterprise Linux 8 (RHEL 8) Operating System (OS). In this guide, the reader can find information about the class dependencies, the general structure and architecture of the OpenDSS-X application. This information is useful for developers interested in modifying the source code for expanding the base OpenDSS-X capabilities, or just for filling out their own simulation needs.

The IDEs presented in Table 1 are the ones tested for compiling, debugging, and running the application under MS Windows OS. The OpenDSS-X was developed and compiled in Red Hat Enterprise Linux RHEL 8 using Clang/CMake (Table 2).

Table . IDEs supported by OpenDSS-X

| IDE | OS |
| --- | --- |
| Visual Studio Community 2019 | Windows 10 |
| Visual Studio Community 2022 | Windows 10 |
| Visual Studio Professional 2019 | Windows 10 |
| Visual Studio Professional 2022 | Windows 10 |

Table . OpenDSS-X Compiler Requirements

| Item | Description |
| --- | --- |
| C++ | GCC 4.8.5 (Red Hat 4.8.5-4)  Microsoft Visual Studio Community 2022 (64-bit) - Version 17.1.3 (Windows 10) |
| Build Engine | Clang 12.0.1 C++ compiler (Red Hat 12.0.1-4)  CMake 3.20.2 for build process |

The Visual Studio IDEs were selected because of their compatibility with CMake. Information on CMake can be found on the distribution site for the software application (CMake, 2022). The OpenDSS-X project files do not include a Visual Studio project because Visual Studio operates as an IDE around CMake.

For information on how to setup and compile the project using Visual Studio please refer to the OpenDSS-X Build Guide. The latter also includes the hardware requirements of computer platforms for which OpenDSS-X has been tested to date.

Note that implementations of OpenDSS-X on Linux systems using a GCC compiler do not support the use of the \_\_declspec attribute. The latter is Microsoft's macro or tag that defines features used in their version of C++. Software developers implementing new capabilities to OpenDSS-X are advised against using such features to avoid incompatibilities when building the application for use on Linux systems.

# OpenDSS-X Repository

OpenDSS-X[[1]](#footnote-2) is an electric power distribution system simulator (DSS) designed to support distributed energy resource (DER) grid integration and grid modernization. It enables engineers to perform complex analyses using a flexible, customization, and easy to use platform intended specifically to meet current and future distribution system challenges and provides a foundation for understanding and integrating new technologies and resources. This power system analysis software tool was translated from the Delphi computer language (Electric Power Research Institute, 2022) into C++ to make it accessible to a larger community of software developers and used as a core library upon which to build new capabilities. More information on OpenDSS can be found in the Reference Guide (Dugan & Montenegro, 2020). OpenDSS (the Delphi version) that runs on a Windows operating system is available (Electric Power Research Institute, 2022) along with a PowerPoint tutorial (Fu, 2019).

The OpenDSS-X software repository has a file distribution as shown in Figure 1.

A picture containing text

Description automatically generated

Figure 1. Snapshot of the OpenDSS-X copy obtained from the repository

The content of the folders contained in the project structure is as indicated in Table 3.

Table . Contents of OpenDSS-X software repository

| *Folder name* | *Description* |
| --- | --- |
| CMD | Contains the main project files. The project was designed for compatibility with CMake, facilitating the cross-platform adoption without losing some of the C++ semantics introduced by other vendors. In this folder the user will find the main project file (OpenDSSX.cpp), the CMake configuration files and the external libraries such as KLUSolve.dll and its Linux equivalents. |
| Common | Contains the routines and objects commonly used across the program for simulation purposes. Utilities, Solution and Circuit are some examples of the units that can be found in this folder. |
| Controls | Contains the objects implemented as controls in the simulation. Controls are objects that can modify the behavior of a set of elements (it can be 1) using the feedback from a monitored part of the circuit model. Regulators, capacitor controls, smart inverters are some examples of the control that can be found in this folder. |
| Executive | Contains all the routines and objects that interpret commands given using the OpenDSS-X scripting language, redirecting the program internally to perform the actions commanded by the user using this language. |
| Forms | Contains the routines for returning messages to the user. The messages can be confirmation, warnings, or errors. This is the equivalent for command line of the VCL forms implemented in the original OpenDSS written in Delphi. |
| General | Contains the objects representing libraries that can be use to simplify the elements’ description within a script. LineCodes, LineGeometries, TransformerCodes and XYCurves are examples of these objects. |
| GISCommands | Contains the commands for interacting with OpenDSS-GIS. Not available in OpenDSS-X but is in the repository to mimic the file distribution in the original version (Delphi). |
| Meters | Contains the code for the objects representing meters, monitors and sensor like elements that can be deployed across the circuit model for capturing simulation values. |
| MyOpenDSS | Contains a set of files that can be used as template for external user models. |
| Parallel\_Lib | Implements the routines and objects needed for performing parallel processing related operations. |
| Parser | Contains the objects implementing parsers and RPN for interpreting numbers and mathematical expressions when processing commands in OpenDSS-X scripting language. |
| PCElements | Contains the code for the objects describing Power Conversion Elements (PCEs). These are normally connected in shunt across the model. Loads, capacitors, energy storage are some examples of PCEs. |
| PDElements | Contains the code for the objects describing Power Delivery Elements (PDEs). These can have 2 terminals and are normally connected in series across the model for energy transport. Transformers, lines, and series connected capacitors are some examples of PDEs. |
| Shared | Contains routines shared by multiple objects across the program. They are vital for the program operation. |
| Support | Contains the routines implemented for reproducing the operation of Delphi specialized commands in C++. These routines are fundamental for the program operation since many of the commands that can be fund across the code are not C++ native. |

# OpenDSS-X Classes, objects, and hierarchy

The hierarchy tree in OpenDSS-X is composed by several levels. These levels are derived for feeding the properties and methods used by the circuit elements in OpenDSS-X. There are 6 groups of elements within OpenDSS-X, these are:

1. PCE – Power Conversion Elements, these are elements modeled to absorb/deliver energy to the circuit model and are normally connected in shunt (between phases or line to ground – delta/wye), implying that only 1 terminal is enabled for connection to the power system.
2. PDE – Power delivery Elements, these are elements modeled to transport energy across the circuit model. They have normally 2 terminals but some of them can have more than 2 terminals like in the case of transformers. Normally, these types of elements will be connected in series for interconnecting nodes across the circuit model, however, they can also be connected in shunt like in the case of capacitors and reactors for modeling purposes.
3. Controls- Control devices are algorithms that emulate control actions affecting PCE and PDE in the same way real controls act in the field. Need to be declared after the PCE/PDE they are linked to, and their actions take place after a successful simulation step.
4. General-purpose- These elements act as reference libraries that can be used by PCEs and PDEs to inherit configuration, simplifying the amount of code needed for describing a circuit model in OpenDSS-X.
5. Meters- these objects are created for recording values during the simulation and sometimes, to provide those values to automations built-in OpenDSS-X. Meters can be assigned to PCE and PDE and have different operational modes.
6. Containers- These objects are the higher layer in the simulation process, they can contain the whole circuit (PDE, PCE, Meters), implement the solution algorithms and simulation routines, among other high-level operations. These containers are the heart of the simulation and bring together all the other objects around it.

Depending on their nature, OpenDSS-X objects and classes derive from a different master class. Classes and objects are defined separately, leaving the element class as reference for the element type properties and methods, and the objects are used for creating elements within the circuit and interact with them during the simulation.

## Power Conversion Elements

Power Conversion Elements (PCE) are elements derived from TDSSClass and NamedObject classes. TDSSClass is the reference for creating a new PCE class and NamedObject is the base for creating new PCE objects. The inheritance tree for PCE classes and objects is shown in Figure 2 and Figure 3 respectively.

Diagram

Description automatically generated

Figure . PCE class inheritance tree

### TDSSClass

| *TDSSClass* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Method- private | Set\_active | Sets the class/element given by index at the argument active. |
| Property-private | Get\_Element | Returns a pointer to the element’s index given in the argument. |
| Property-private | Get\_First | Returns the index of the first element in the list for the active class. If none, returns 0. Lists start in 1. |
| Property-private | Get\_Next | Returns the index of the next element in the list of for the active class. If none, returns 0. Lists start in 1. |
| Property-protected | AddObjectToList | Adds a new object to the list of the active class. |
| Property-protected | Get\_FirstPropertyName | Returns the name of the first property of the active element. Sets the element’s property cursor in 1. If none, returns 0. |
| Property-protected | Get\_NextPropertyName | Returns the name of the next property of the active element. increments the element’s property cursor in 1. If none, returns 0. |
| Property-protected | MakeLike | Created as reference for copying the properties of another element into the caller. It needs to be detailed at a more specific class. |
| Method-protected | CountProperties | Counts the number of properties defined for the caller. |
| Method-protected | AllocatePropertyArrays | Allocates memory for storing the property arrays of the caller. |
| Method-protected | DefineProperties | Created for adding a property to the caller. |
| Property-protected | ClassEdit | Routine to continue parsing with contents of Parser. |
| Method-public | AddProperty | Helper routine for building Property strings |
| Method-public | ReallocateElementNameList | Reallocate the device name list to improve the performance of searches. |
| Property- public | Edit | Implements the base for editing the caller’s properties. The content of this routine needs to be defined at local level for the caller. Reaching this instance produces an error. |
| Property- public | Init | Implements the base for initializing the caller’s properties and variables. The content of this routine needs to be defined at local level for the caller. Reaching this instance produces an error. |
| Property- public | NewObject | Implements the routine for adding a new element to the list of the active class. |
| Property- public | SetActive | Activates an element within the active class by name. |
| Property- public | GetActiveObj | Returns a pointer to the active element within the active class. |
| Property- public | Find | Returns a pointer to the element’s name given in the argument. If not found, the pointer will be NULL (nil). |
| Property- public | PropertyIndex | Returns the index of the property name given in the argument. If not found, returns 0. |
| Property-public | FirstPropertyName | Returns the index to the first property in the property list for the active element. PA the same routine in the private/protected context. |
| Property-public | NextPropertyName | Returns the index to the next property in the property list for the active element. PA the same routine in the private/protected context. |
| Property-public | Active | Set/get by index the active element in the active class. |
| Property-public | ElementCount | Returns the number of elements within the active class. |
| Property-public | First | Activates the first element of the list for the active class. Returns 1 if success, otherwise, returns 0. |
| Property-public | Next | Activates the next element of the list for the active class. Returns the elements index if success, otherwise, returns 0. |
| Property-public | Name | Returns the name of the active class. |

### TDSSCktElementClass

|  |  |  |
| --- | --- | --- |
| *TDSSCktElementClass* | | |
| *Type-access* | *Command* | *Description* |
| Property- protected | ClassEdit | Extension of the class edit command for the caller, it is used for editing the inherited values basefrequency and enabled. |
| Method- protected | ClassMakeLike | Copy the basefrequency and enabled values from another object within the same class. |
| Method- protected | CountProperties | Add number of intrinsic properties to the number of properties of the caller object. |
| Method- protected | DefineProperties | Add Properties of this class to propName of the caller. |

### TPCClass

|  |  |  |
| --- | --- | --- |
| *TPCClass* | | |
| *Type-access* | *Command* | *Description* |
| Property- protected | ClassEdit | Extension of the class edit command for the caller, it is used for editing the inherited Spectrum. |
| Method- protected | ClassMakeLike | Copy the Spectrum values from another object within the same class. |
| Method- protected | CountProperties | Add number of intrinsic properties to the number of properties of the caller object. |
| Method- protected | DefineProperties | Add Properties of this class to propName of the caller. |

The properties and methods for the more specific classes in this tree are in essence the same, variations are included in the following table for addressing them.

### Specific classes

| *Specific classes* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Method- protected | InterpretConnection | Interprets the connection string provided when declaring a load connection type (wye/delta). –(TLoad, TGenerator, TPVSystem, TStorage) |
| Method- protected | SetNcondsForConnection | Sets the number of conductors for the load depending on the connection type. –(TLoad, TIndMach012, TGenerator, TPVSystem, TStorage) |
| Property- protected | MakeLike | Copy the properties from another element within the same class. |
| Method- protected | DefineProperties | Add Properties of this class to for this object. It also initializes the more generic class properties. |
| Method- public | Edit | Overrides the method at the more generic class definition, it implements the routine for editing the properties of the element when declared/edit. |
| Method- public | Init | Overrides the method at the more generic class definition, it implements the routine for initializing the properties of the element when declared/edit. |
| Method- public | NewObject | Overrides the method at the more generic class definition, it implements the routine for adding a new object to the list of elements within the active class. |
| Method- private | IsourceSetBus1 | Sets the name and terminals for bus 1 of this element. – (TISource) |
| Method- private | VsourceSetBus1 | Sets the name and terminals for bus 1 of this element. –(TVSource) |
| Method- protected | GICLineSetBus1 | Sets the name and terminals for bus 1 of this element. –(TGICLine) |
| Method- public | ResetRegistersAll | Resets the registers (Local energy meter) for this element. – (TGenerator, YPVSystem) |
| Method- public | SampleAll | Force all registers (local energy meter) for this element take a sample. – (TGenerator) |
| Method- public | UpdateAll | Intended to update all enabled energy system levels, not implemented to date for TPVSystem. – (TPVSystem, TStorage) |
| Method- private | VscSetBus1 | Sets the name and terminals for bus 1 of this element. – (TVSConveter) |

Diagram

Description automatically generated

Figure . PCE object inheritance tree

### TNamedObject

| *TNamedObject* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Property- private | Get\_QualifiedName | Returns the name of the caller including class. E.g. Load.myload. |
| Property- private | Get\_DisplayName | Returns the name to be displayed for the caller, if not assigned, returns the class, underscore character and element names. |
| Property- private | Set\_DisplayName | Sets the name to be displayed (different from the objects name) for the caller. |
| Property- private | Get\_UUID | Returns the unique identifier for the caller object. |
| Property- private | Get\_ID | Returns the unique identifier for the caller object as a string. |
| Property- private | Set\_UUID | Sets the unique identifier for the caller object. |
| property- public | DSSClassName | Returns the name of the class the caller belongs to. |
| property- public | LocalName | PA the objects name within the circuit. It is not the class nor display name. E.g. load.myload, LocalName = myload. |
| property- public | QualifiedName | PA Get\_QualifiedName. |
| property- public | DisplayName | PA Get\_DisplayName, Set\_DisplayName. |
| property- public | UUID | PA Get\_UUID, Get\_UUID. |
| property- public | ID | PA Get\_ID. |
| property- public | CIM\_ID | Returns the string defining the CIM\_ID for the caller object. |

### TDSSObject

|  |  |  |
| --- | --- | --- |
| *TDSSObject* | | |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Property- private | Get\_PropertyValue | Returns the value of the property given by index. The value is returned as string. |
| Method- private | Set\_PropertyValue | Sets the property value (String) for the property given by index in the argument. |
| Property- private | Get\_Name | Returns the name of the caller. |
| Method- private | Set\_Name | Sets the name of the caller using the string given in the argument. |
| Property- protected | GetNextPropertySet | Find next larger property sequence number returns 0 if none found. |
| Property- public | GetPropertyValue | Uses dssclass.propertyindex to get index by name, returns the value as string. |
| Method- public | InitPropertyValues | Routine for initializing the caller’s property values. |
| Method- public | DumpProperties | Implements the routine for writing the local properties down into a plain text file at the end. |
| Method- public | SaveWrite | Writes into the given plain text file only properties that were explicitly set in the final order they were actually set. |
| Method- public | ClearPropSeqArray | Clears the property array for the caller. |
| Property- public | PropertyValue | PA Get\_PropertyValue, Set\_PropertyValue. |
| Property- public | Name | PA Get\_Name, Set\_Name. |

### TDSSCktElement

| *TDSSCktElement* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Method- private | Set\_Freq | Set frequency and recompute YPrim. |
| Method- private | Set\_Nconds | Sets the number of conductors and creates the conductor array within the class for the caller. |
| Method- private | Set\_NPhases | Sets the number of phases for the device. |
| Method- private | Set\_ActiveTerminal | Activates the given terminal for future queries/operations. |
| Property- private | Get\_ConductorClosed | Evaluates if the conductor given by index is closed. |
| Method- private | Set\_YprimInvalid | Sets the YPrim invalid flag (Boolean) in the value given in the argument. |
| Property- private | Get\_YprimInvalid | Returns the value of the YPrim invalid Boolean flag. |
| Property- private | Get\_FirstBus | Returns the name of bus 1 for the element. |
| Property- private | Get\_NextBus | Returns the name of next bus of the element. |
| Property- private | Get\_Losses | Returns the total losses for the caller. |
| Property- private | Get\_Power | Gets total complex power in active terminal. |
| Property- private | Get\_MaxPower | Gets equivalent total complex power in active terminal based on phase with max current. |
| Property- private | Get\_MaxCurrent | Gets equivalent total complex current on phase with max current. Uses the terminal number given in the argument. |
| Property- private | Get\_MaxCurrentAng | Gets equivalent angle of the total complex current on phase with max current. Uses the terminal number given in the argument. |
| Property- private | Get\_MaxVoltage | Gets equivalent total complex voltage on phase using the terminal index given in the argument. |
| Property- private | Get\_MaxVoltageAng | Gets equivalent total complex voltage phase angle on phase using the terminal index given in the argument. |
| Property- private | Get\_PCE\_Value | Gets a value for the active PCE such as P, Q, Vmag, IMag, etc. The value type is given in the argument as an integer number. |
| Method- private | DoYprimCalcs | Calculates the YPrim matrix for the caller. |
| Method- protected | Set\_Enabled | Sets the enabled flag using the Boolean value given in the argument. |
| Method- protected | Set\_ConductorClosed | Sets the conductor closed flag using the index and Boolean values given in the argument. |
| Method- protected | Set\_NTerms | Sets the number of terminals for the calling object. |
| Method- protected | Set\_Handle | Sets a handle in the form of an integer number for the calling object. |
| Property- public | GetYPrim | Returns a handler to the object’s YPrim depending on the type of matrix specified in the argument (series, shunt, full). |
| Property- public | GetYPrimValues | Returns the values of the object’s YPrim specified in the argument (series, shunt, full). |
| Property- public | MaxTerminalOneIMag | Get max of phase currents on the first terminal; Requires computing Iterminal. |
| Method- public | ComputeIterminal | Computes Iterminal for this device and put them into a local array. |
| Method - public | ComputeVterminal | Extracts the voltages in the active terminal of the calling object and put terminal voltages in an array. |
| Method - public | ZeroITerminal | Sets the local current array to 0. |
| Method - public | GetCurrents | Get present value of terminal Currents for reports. |
| Method- public | GetInjCurrents | Returns last Injection currents calculated for this object (caller). |
| Property- public | InjCurrents | Applies to PC Elements. Puts the injection currents locally calculated straight into the Solution Array. |
| Property- public | GetBus | Returns the name of the bus addressed using the index in the argument. |
| Method- public | SetBus | Sets the name of the bus addressed using the index in the argument. |
| Method- public | SetNodeRef | Set NodeRef Array for fast solution with intrinsics. |
| Method- public | RecalcElementData | Default for recalculating the foundation of the calling element. It is expected to be override by a more specific class. If not overridden, when called it will trigger an error message. |
| Method- public | CalcYPrim | Calculates the YPrim type given in the argument (series, shunt, full). |
| Method- public | MakePosSequence | Make a positive Sequence Model for the calling object. |
| Method- public | GetTermVoltages | Fills Vbuffer array which must be adequately allocated by calling routine. Vbuffer is the local array for storing the element voltages. |
| Method- public | GetPhasePower | Gets the power in each phase (complex losses) of active terminal neutral conductors are ignored by this routine. |
| Method- public | GetPhaseLosses | Gets the losses in each phase (complex losses); Power difference coming out each phase. Note: This can be misleading if the nodev voltage is greatly unbalanced. Neutral conductors are ignored by this routine. |
| Method- public | GetLosses | Returns total losses and set LoadLosses=total losses and noload losses =0. |
| Method- public | GetSeqLosses | For the base class, just return Complex ZERO. |
| Property- public | GetPropertyValue | Returns the value (string) for the property indicated by index in the argument. |
| Method- public | InitPropertyValues | Initializes the property values for the class with the default values. |
| Method- public | DumpProperties | Implements the routine for writing the local properties down into a plain text file at the end. |
| Property- public | Handle | PA FHandle (local handle) and Set\_Handle. |
| Property- public | Enabled | PA FEnabled (variable) and Set\_Enabled. |
| Property- public | YPrimInvalid | PA get\_YprimInvalid and set\_YprimInvalid. |
| Property- public | YPrimFreq | PA FYprimFreq (variable) and Set\_Freq. |
| Property- public | NTerms | PA Fnterms (variable) and Set\_NTerms. |
| Property- public | NConds | PA Fnconds (variable) and Set\_Nconds. |
| Property- public | NPhases | PA Fnphases (variable) and Set\_NPhases. |
| Property- public | FirstBus | PA Get\_FirstBus. |
| Property- public | NextBus | PA Get\_NextBus. |
| Property- public | Losses | PA Get\_Losses. |
| Property- public | Power | PA Get\_Power. |
| Property- public | MaxPower | PA Get\_MaxPower. |
| Property- public | MaxCurrent | PA Get\_MaxCurrent. |
| Property- public | MaxVoltage | PA Get\_MaxVoltage. |
| Property- public | MaxCurrentAng | PA Get\_MaxCurrentAng. |
| Property- public | MaxVoltageAng | PA Get\_MaxVoltageAng. |
| Property- public | ActiveTerminalIdx | PA FActiveTerminal (variable) and Set\_ActiveTerminal. |
| Property- public | Closed | PA Get\_ConductorClosed and Set\_ConductorClosed. |
| Property- public | PCEValue | PA Get\_PCE\_Value. |
| Method- public | SumCurrents | Sums the terminal Currents into System Currents Array primarily for Newton Iteration. |
| Method- public | Get\_Current\_Mags | Returns the Currents vector in magnitude. |

### TPCElement

| *TPCElement* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Method- private | GetTerminalCurrents | Returns the currents at the active terminal |
| Property- private | Get\_Variable | Returns the numeric value (double) of the state variable indicated in the argument by index. |
| Method- private | Set\_Variable | Sets the numeric value (double) of the state variable indicated in the argument by index. |
| Method- public | ZeroInjCurrent | Sets the local injection currents array to zero. |
| Method- public | InitPropertyValues | Initializes the local property array with the default values. |
| Method- public | GetCurrents | Gets the present current values at the active terminal. |
| Method- public | GetInjCurrents | Gets the injection currents array at the active terminal. |
| Method- public | ComputeIterminal | Overrides the generic class call to define the local procedure for computing the current at the element’s terminal. |
| Property- public | InjCurrents | Overrides the generic class call to define the local procedure for adding the local injection currents into System currents array. |
| Method- public | CalcYPrimContribution | Calculates the Yprim contribution to the injection currents array using the present values. |
| Method- public | DumpProperties | Implements the routine for writing the local properties down into a plain text file at the end. |
| Method- public | InitHarmonics | Initializes the variables of the calling object for harmonics simulation mode. |
| Method- public | set\_ITerminalUpdated | Sets the ITerminalUpdated flag using the value given in the argument. |
| Method- public | InitStateVars | Sets the foundation for initializing the state variables of the calling object. It is expected to be override by a more specific class. If not overridden, when called it does nothing. |
| Method- public | IntegrateStates | Sets the foundation for defining the procedure to integrate the state variables of the calling object. It is expected to be override by a more specific class. If not overridden, when called it does nothing. |
| Property- public | NumVariables | Sets the foundation for defining the number of state variables of the calling object. It is expected to be override by a more specific class. If not overridden, when called it does nothing. |
| Method- public | GetAllVariables | Sets the foundation for returning the values for all the state variables of the calling object. It is expected to be override by a more specific class. If not overridden, when called it does nothing. |
| Property- public | VariableName | Sets the foundation for returning the value the state variable indicated in the argument by index. It is expected to be override by a more specific class. If not overridden, when called it does nothing. |
| Property- public | LookupVariable | Search through variable name list and return index if found. Otherwise, returns -1. |
| Property- public | Property- public | PA Get\_Variable and Set\_Variable. |

### TLoadObj

| *TLoadObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Property- private | AllTerminalsClosed | Gets a Boolean flag indicating if all the terminals for the active element are closed (true). |
| Method- private | Set\_CalcDailyMult | Calculates the multiplier for daily simulation at the simulation time step given in the argument, using the given load shape for daily simulation if any. |
| Method- private | CalcDutyMult | Calculates the multiplier for duty simulation at the simulation time step given in the argument, using the given load shape for duty simulation if any. |
| Method- private | CalcInjCurrentArray | Fills the Injection Current array with the current values to use for injections. |
| Method- private | CalcLoadModelContribution | Calculates total load current and adds it properly into the Injection Current array. |
| Method- private | CalcVTerminalPhase | Establishes the phase voltages and stick them into Vtemp (local voltage array). |
| Method- private | CalcYearlyMult | Calculates the multiplier for yearly simulation at the simulation time step given in the argument, using the given load shape for yearly simulation if any. |
| Method- private | CalcCVRMult | Calculates the multiplier for yearly simulation using the CVR curve. CVR curve is assumed to be used in a yearly simulation. |
| Method- private | CalcYPrimMatrix | Calculates the Y primitive matrix for the model. |
| Method- private | DoConstantILoad | Implements the routine for calculating the contribution of the load as a constant current load model. |
| Method- private | DoConstantPQLoad | Implements the routine for calculating the contribution of the load as a constant PQ load model. |
| Method- private | DoConstantZLoad | Implements the routine for calculating the contribution of the load as a constant impedance load model. |
| Method- private | DoFixedQ | Implements the routine for calculating the contribution of the load as a Constant P, Fixed Q load model. Q is always kvarBase. |
| Method- private | DoFixedQZ | Implements the routine for calculating the contribution of the load as a Constant P, Fixed Q load model. Q is always a fixed Z derived from kvarBase. |
| Method- private | DoHarmonicMode | Implements the routine for calculating the injection currents for the model in harmonics simulation mode. |
| Method- private | DoCVRModel | Implements the routine for calculating the contribution of the load as a Linear P, quadratic Q load model. |
| Method- private | DoZIPVModel | Implements the routine for calculating the contribution of the load as a ZIP model. |
| Method- private | SetZIPVSize | Allocates the memory needed for storing the ZIP coefficients for the load. |
| Method- private | DoMotorTypeLoad | Implements the routine for calculating the contribution of the load as a motor load model. |
| Property- private | GrowthFactor | Returns the multiplier for the year given in the argument to be used as load growth in time-based simulation modes. |
| Method- private | StickCurrInTerminalArray | Puts the current into the proper location according to connection. |
| Property- private | InterpolateY95\_YLow | Routine for Vmag between V95 and Vlow, interpolate for equivalent Y. |
| Property- private | InterpolateY95I\_YLow | Routine for Vmag between V95 and Vlow, interpolate for equivalent Y. |
| Property- private | Get\_Unserved | Returns the number of unserved customers at this load. |
| Method- private | Set\_kVAAllocationFactor | Sets the KVA allocation factor for the active load. |
| Method- private | Set\_ConnectedkVA | Sets the amount of KVA represented by this load. |
| Method- private | ComputeAllocatedLoad | Calculates the amount of load allocated at the current connection point. Fixed loads defined by kW, kvar or kW, pf are ignored. |
| Method- private | Set\_CFactor | Sets the kWh billed spec. Computes the allocated load. |
| Method- private | Set\_kWh | Sets the amount of kWh for the load at present. Computes the allocated load. |
| Method- private | Set\_kWhDays | Sets the amount of kWh days for the load at present. Computes the allocated load. |
| Method- private | Set\_AllocationFactor | Sets the allocation factor for the load. |
| Method- private | SetkWkvar | Sets kW and kvar for the load, both given in the arguments. |
| Method- private | set\_nZIPV | Reallocates memory for ZIP coefficients. |
| Method- protected | GetTerminalCurrents | Overrides the more generic foundation. Obtains the present current values at the active terminal. |
| Property- public | Get\_ExceedsNormal | Routine for determining if the voltage values at present for the load exceed the normal limits. |
| Method- public | RecalcElementData | Overrides the more generic function. Recalculates the values required for the element. |
| Method- public | CalcYPrim | Overrides the more generic function. Recalculates the YPrim. |
| Property- public | InjCurrents | Overrides the more generic function. Gets the injection currents and add them directly into the Currents array. |
| Method- public | GetInjCurrents | Overrides the more generic function. Gets the injection currents for the last solution performed. |
| Method- public | InitHarmonics | Overrides the more generic function. Initializes the load values for harmonic simulation. |
| Method- public | MakePosSequence | Overrides the more generic function. Makes the positive sequence model for the element. |
| Method- public | SetNominalLoad | Sets nominal load for different simulation modes. |
| Method- public | Randomize | Randomizes the load multiplier value locally using 3 distribution options (0=none, 1=Gaussian, 2=Uniform). |
| Property- public | GetPropertyValue | Overrides the more generic function. Returns the value (string) for the property indicated by index in the argument. |
| Method- public | InitPropertyValues | Overrides the more generic function. Initializes the local property array with the default values. |
| Method- public | DumpProperties | Overrides the more generic function. Implements the routine for writing the local properties down into a plain text file at the end. |
| Method- public | UpdateVoltageBases | Updates the voltage bases for the load. |
| Property- public | Unserved | PA Get\_Unserved. |
| Property- public | AllocationFactor | PA FAllocationFactor (variable) and Set\_AllocationFactor. |
| Property- public | kVAAllocationFactor | PA FkVAAllocationFactor (variable) and Set\_kVAAllocationFactor. |
| Property- public | ConnectedkVA | PA FConnectedkVA (variable) Set\_ConnectedkVA. |
| Property- public | kWh | PA FkWh (variable) and Set\_kWh. |
| Property- public | kWhDays | PA FkWhDays (variable) and Set\_kWhDays. |
| Property- public | CFactor | PA FCFactor (variable) and Set\_CFactor. |
| Property- public | puMean | PA FpuMean (variable). |
| Property- public | puStdDev | PA FpuStdDev (variable). |
| Property- public | CVRwatts | PA FCVRwattFactor (variable). |
| Property- public | CVRvars | PA FCVRvarFactor (variable). |
| Property- public | MaxPU | PA Vmaxpu (variable). |
| Property- public | MinEmerg | PA VminEmerg (variable). |
| Property- public | MinNormal | PA VminNormal (variable). |
| Property- public | MinPU | PA Vminpu (variable). |
| Property- public | ExemptLoad | PA ExemptFromLDCurve (variable). |
| Property- public | FixedLoad | PA Fixed (variable). |
| Property- public | nZIPV | PA FnZIPV (variable) and set\_nZIPV. |
| Property- public | IsPFSpecified | PA PFSpecified (variable). |

### TISourceObj

| *TISourceObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Implements the following properties/methods as in TLoadObj:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties | | |

### TVSourceObj

|  |  |  |  |
| --- | --- | --- | --- |
| *TVSourceObj* | | | |
| *Type-access* | *Command* | | *Description – (Specific class) otherwise generic* |
| Implements the following properties/methods as in TLoadObj:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties 9. CalcDailyMult 10. CalcDutyMult 11. CalcYearlyMult 12. GetPropertyValue | | | |
| Method- private | GetVterminalForSource | Determines the form of the VSource terminal to handle voltage sources of any number of phases assuming they are equally displaced in time. | |
| Property- private | InterpretSourceModel | Interprets the voltage source model. | |

### TIndMach012Obj

| *TIndMach012Obj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Implements the following properties/methods as in TLoadObj:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties 9. CalcDailyMult 10. CalcDutyMult 11. CalcYearlyMult 12. GetPropertyValue 13. CalcYPrimMatrix 14. CalcInjCurrentArray 15. GetTerminalCurrents 16. DoHarmonicMode 17. InitHarmonics | | |
| Method- private | InterpretOption | Interprets the option for slip option. |
| Method - private | set\_Localslip | Sets the slip locally using the slip option set by the user. |
| Method - private | Get\_PFlowModelCurrent | Gets the power flow model current. |
| Method - private | Get\_DynamicModelCurrent | Uploads the injection currents for dynamics simulation mode. |
| Method - private | Set\_Slip | Sets the motor slip value. |
| Property - private | GetRotorLosses | Returns the rotor losses. |
| Property - private | GetStatorLosses | Returns the stator losses. |
| Property - private | Compute\_dSdP | Calculates and returns the dSdP based on rated slip and rated voltage. |
| Method - private | Randomize | Implements the typical process for handling randomization in DSS fashion. |
| Method - private | InitModel | Implements the routine for initializing the state variables and local variables of the model. |
| Method - private | CalcIndMach012ModelContribution | Calculates the current injections for the model. |
| Method - private | DoIndMach012Model | Computes the total terminal current for the model. |
| Method - private | CalcModel | Given the present voltages returns currents in accordance to the simulation mode. |
| Method - private | InitTraceFile | Initializes file for debugging purposes. |
| Method - private | WriteTraceRecord | Adds a new line to the trace file. |
| Property - private | Get\_PresentkV | Gets the present voltage base at the model terminals. |
| Method - private | Set\_PresentkV | Sets the present voltage base at the model terminals. |
| Method - private | SetPowerkW | Sets the kW base for the model. |
| Method - protected | Set\_ConductorClosed | Overrides the more generic class implementation to open/close model’s conductors. |
| Method - protected | DoDynamicMode | Implements the routine for calculating the injection currents for the model in dynamics simulation mode. |
| Method - public | Integrate | Implements the procedure for integrating the state variables of the model in dynamics simulation mode. |
| Method - public | CalcDynamic | Provides the structure for calculating the dynamic injection currents for the model. |
| Method - public | CalcPFlow | Calculates the power flow for the model using the local symmetric components. |
| Method - public | SetNominalPower | Sets the power output for the model. |
| Property - public | NumVariables | Overrides the more generic class implementation, returns the number of state variables in this model. |
| Method - public | GetAllVariables | Overrides the more generic class implementation, uploads the values for the state variables in this model into the given pointer. |
| Property - public | Get\_Variable | Overrides the more generic class implementation, returns the value for the state variable pointed by given index. |
| Method - public | Set\_Variable | Overrides the more generic class implementation, sets the value for the state variable pointed by given index. |
| Property - public | VariableName | Overrides the more generic class implementation, returns name of the state variable pointed by given index. |
| Method - public | InitStateVars | Overrides the more generic class implementation, initializes the state variables. |
| Method - public | IntegrateStates | Overrides the more generic class implementation, solves, and integrates the differential equation describing the model dynamics. |
| Property - public | LocalSlip | PA S! (variable) and set\_Localslip. |
| Property - public | Slip | PA Set\_Slip. |
| Property - public | PresentkV | PA Get\_PresentkV and Set\_PresentkV. |

### TGICSourceObj

|  |  |  |  |
| --- | --- | --- | --- |
| *TGICSourceObj* | | | |
| *Type-access* | *Command* | | *Description – (Specific class) otherwise generic* |
| Implements the following properties/methods as in TLoadObj:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties 9. GetPropertyValue | | | |
| Method- private | GetVterminalForSource | Determines the form of the GICSource terminal to handle voltage sources of any number of phases assuming they are equally displaced in time. | |
| Property- private | Compute\_VLine | Computes the GICSource voltage based on the GIS coordinates (latitude-longitude) . | |

### TGICLineObj

| *TGICLineObj* | | | |
| --- | --- | --- | --- |
| *Type-access* | *Command* | | *Description – (Specific class) otherwise generic* |
| Implements the following properties/methods as in TLoadObj:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties 9. GetPropertyValue | | | |
| Method- private | GetVterminalForSource | Determines the form of the GICLine terminal to handle voltage sources of any number of phases assuming they are equally displaced in time. | |
| Property- private | Compute\_VLine | Computes the line voltage based on the GIS coordinates (latitude-longitude) . | |

### TGeneratorObj

| *TGeneratorObj* | | | |
| --- | --- | --- | --- |
| *Type-access* | *Command* | | *Description – (Specific class) otherwise generic* |
| Implements the following properties/methods as in TLoadObj and *TIndMach012Obj*:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties 9. CalcDailyMult 10. CalcDutyMult 11. CalcYearlyMult 12. GetPropertyValue 13. CalcYPrimMatrix 14. CalcInjCurrentArray 15. GetTerminalCurrents 16. DoHarmonicMode 17. InitHarmonics 18. Integrate 19. StickCurrInTerminalArray 20. WriteTraceRecord 21. Set\_ConductorClosed 22. VariableName 23. NumVariables 24. GetAllVariables 25. Get\_Variable 26. Set\_Variable 27. VariableName 28. Randomize 29. InitStateVars 30. IntegrateStates 31. CalcVTerminalPhase | | | |
| Method- private | CalcGenModelContribution | Calculates the injection currents for the element based on the model type. | |
| Method- private | CalcVterminal | Calculates and uploads the voltage at the element’s terminals using the model and connection type as reference. | |
| Method- private | CalcVthev\_Dyn | Calculates the Thevenin voltage equivalent for dynamics simulation. | |
| Method- private | DoConstantPQGen | Implements the routine for calculating the injection currents for generator model when working in constant PQ mode. | |
| Method- private | DoConstantZGen | Implements the routine for calculating the injection currents for generator model when working as constant admittance. | |
| Method- private | DoCurrentLimitedPQ | Implements the routine for calculating the injection currents for generator model when working as constant kW, kvar, but current-limited below Vminpu. Approximates a simple inverter. | |
| Method- private | DoDynamicMode | Implements the routine for calculating the injection currents for generator model when working in dynamics simulation mode. | |
| Method- private | DoFixedQGen | Implements the routine for calculating the injection currents for generator model when working as constant kW, fixed Q. | |
| Method- private | DoFixedQZGen | Implements the routine for calculating the injection currents for generator model when working as constant kW, fixed Q (as a constant reactance). | |
| Method- private | DoPVTypeGen | Implements the routine for calculating the injection currents for generator model when working as constant kW, constant kV. Somewhat like a conventional transmission power flow P-V generator. | |
| Method- private | DoUserModel | Implements the routine for calculating the injection currents for a user generator model (DLL). | |
| Property- private | CheckOnFuel | Returns a boolean flag indicating if the fuel tank (if any) is empty. | |
| Method- private | SetDragHandRegister | Moves the internal shift register according to the given value. | |
| Method- private | SyncUpPowerQuantities | Keeps kvar nominal up to date with kW and PF. | |
| Property- private | Get\_PresentkW | Returns the present kW delivered by the model. | |
| Property- private | Get\_Presentkvar | Returns the present kvar delivered/absorbed by the model. | |
| Property- private | Get\_PresentkV | Returns the present kV at the terminals of the model. Considers the connection mode (wye/delta). | |
| Method- private | Set\_PresentkV | Sets the present kV at the terminals of the model. Considers the connection mode (wye/delta). | |
| Method- private | Set\_Presentkvar | Sets the present kvar delivered/absorbed by the model. | |
| Method- private | Set\_PresentkW | Sets the present kW delivered by the model. | |
| Method- private | Set\_PowerFactor | Sets the present power factor delivered by the model. | |
| Method- private | SetkWkvar | Sets the present kW and kvar for the model. | |
| Method- public | SetNominalGeneration | Implements the calls for calculating the injection currents for the model considering its features. | |
| Method- public | ResetRegisters | Resets all the registers needed for integration and the local energy meter. | |
| Method- public | TakeSample | Commands the local energy meter to take a sample. | |
| Method- public | InitDQDVCalc | Initializes DQ DV registers. | |
| Method- public | BumpUpQ | Bump up vars by 10% of range for next calculation. | |
| Method- public | RememberQV | Stores Q and V for further use. | |
| Method- public | CalcDQDV | Calculates DQ and DV. | |
| Method- public | ResetStartPoint | Resets the Q nominal per phase to the initial (default) value. | |
| Property- public | CheckIfDynVar | Evaluates if the value provided corresponds to a constant value or to an operand for calculating the value using the simulation results, this is useful when a DynamicExp is linked to the generator. | |
| Method- public | SetDynOutput | Obtains the indexes of the given variables to use them as reference for setting the dynamic output for the generator when using DynamicExp with the generator. | |
| Property- public | PresentkW | PA Get\_PresentkW and Set\_PresentkW. | |
| Property- public | Presentkvar | PA Get\_Presentkvar and Set\_Presentkvar. | |
| Property- public | ForcedON | PA FForcedON (variable). | |
| Property- public | PresentkV | PA Get\_PresentkV and Set\_PresentkV. | |
| Property- public | PowerFactor | PA PFNominal (variable) and Set\_PowerFactor. | |

### TPVsystemObj

| *TPVsystemObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Implements the following properties/methods as in TLoadObj, *TIndMach012Obj and TGeneratorObj*:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties 9. CalcDailyMult 10. CalcDutyMult 11. CalcYearlyMult 12. GetPropertyValue 13. CalcYPrimMatrix 14. CalcInjCurrentArray 15. GetTerminalCurrents 16. DoHarmonicMode 17. InitHarmonics 18. Integrate 19. StickCurrInTerminalArray 20. WriteTraceRecord 21. Set\_ConductorClosed 22. VariableName 23. NumVariables 24. GetAllVariables 25. Get\_Variable 26. Set\_Variable 27. VariableName 28. Randomize 29. InitStateVars 30. IntegrateStates 31. CalcVTerminalPhase 32. DoDynamicMode 33. DoUserModel 34. SetDragHandRegister 35. Get\_PresentkW 36. Get\_Presentkvar 37. Get\_PresentkV 38. Set\_PresentkV 39. Set\_Presentkvar 40. Set\_PresentkW 41. Set\_PowerFactor | | |
| Method- private | CalcDailyTemperature | Calculates the temperature for the present simulation time in daily simulation mode. |
| Method- private | CalcDutyTemperature | Calculates the temperature for the present simulation time in duty cycle simulation mode. |
| Method- private | CalcYearlyTemperature | Calculates the temperature for the present simulation time in Yearly cycle simulation mode. |
| Method- private | CalcYearlyTemperature | Calculates the temperature for the present simulation time in Yearly cycle simulation mode. |
| Method- private | ComputePanelPower | Calculates the total PV power production. |
| Method- private | ComputeInverterPower | Calculates the total inverter power. |
| Method- private | ComputekWkvar | Calculates the total P and Q contributions of the inverter. |
| Method- private | CalcPVSystemModelContribution | Implements the routine for calculating the PV panel power injection. |
| Method- private | DoConstantPQPVSystemObj | Routine for calculating PV contribution as model 1. |
| Method- private | DoConstantZPVSystemObj | Routine for calculating PV contribution as model 2. |
| Method- private | DoConstantZPVSystemObj | Routine for calculating PV contribution as model 2. |
| Method- private | UpdatePVSystem | Updates PVSystem elements based on present kW and IntervalHrs variable. |
| Property-private | Get\_PresentIrradiance | Returns the present irradiance (pu). |
| Method- private | Set\_PresentIrradiance | Sets the present irradiance (pu). |
| Method- private | Set\_pf\_wp\_nominal | Sets the variable Fpf\_wp\_nominal with the given value. |
| Method- private | Set\_kVARating | Sets the KVA rating for the model. |
| Method- private | Set\_Pmpp | Sets the DC power rating for the model. |
| Method- private | Set\_puPmpp | Sets the DC power rating for the model (pu). |
| Method- private | Get\_Varmode/Set\_Varmode | Sets/gets the var control mode. |
| Method- private | Get\_VWmode/Set\_VWmode | Sets/gets the volt-watt control mode. |
| Method- private | Get\_WPmode/Set\_WPmode | Sets/gets if PV is in WP mode. |
| Method- private | Get\_WVmode/Set\_WVmode | Sets/gets if PV is in WV mode. |
| Method- private | Get\_DRCmode/Set\_DRCmode | Sets/gets if PV is in DRC mode. |
| Method- private | Get\_AVRmode/Set\_AVRmode | Sets/gets if PV is in AVR mode. |
| Method- private | kWOut\_Calc | Calculates the kW production. |
| Property-public | Get\_InverterON/ Set\_InverterON | Sets/gets ON/OFF the inverter output. |
| Method- public | Get\_VarFollowInverter/ Set\_VarFollowInverter | Sets/gets ON/OFF the inverter var following functionality of the inverter. |
| Method- public | Set\_Maxkvar | Sets the maximum kvar for the inverter. |
| Method- public | Set\_Maxkvarneg | Sets the minimum limit for kvar production/ absorption. |
| Method- public | SetNominalPVSystemOuput | Implements the routine for setting the nominal PV power output. |
| Property-public | PresentIrradiance | PA Get\_PresentIrradiance and Set\_PresentIrradiance. |
| Property-public | PresentkW | PA Get\_PresentkW and Set\_PresentkW. |
| Property-public | Presentkvar | PA Get\_Presentkvar and Set\_Presentkvar. |
| Property-public | PresentkV | PA Get\_PresentkV and Set\_PresentkV. |
| Property-public | PowerFactor | PA PFnominal (variable) and Set\_PowerFactor. |
| Property-public | kVARating | PA PVSystemVars.FkVARating (variable) and Set\_kVARating. |
| Property-public | Pmpp | PA PVSystemVars.FPmpp and Set\_pmpp. |
| Property-public | puPmpp | PA PVSystemVars.FpuPmpp and Set\_puPmpp. |
| Property-public | Varmode | PA Get\_Varmode and Set\_Varmode. |
| Property-public | VWmode | PA Get\_VWmode and Set\_VWmode. |
| Property-public | VVmode | PA Get\_VVmode and Set\_VVmode. |
| Property-public | WPmode | PA Get\_WPmode and Set\_WPmode. |
| Property-public | WVmode | PA Get\_WVmode and Set\_WVmode. |
| Property-public | AVRmode | PA Get\_AVRmode and Set\_AVRmode. |
| Property-public | DRCmode | PA Get\_DRCmode and Set\_DRCmode. |
| Property-public | InverterON | PA Get\_InverterON and Set\_InverterON. |
| Property-public | VarFollowInverter | PA Get\_VarFollowInverter and Set\_VarFollowInverter. |
| Property-public | kvarLimit | PA PVSystemVars.Fkvarlimit (variable) and Set\_Maxkvar. |
| Property-public | kvarLimitneg | PA PVSystemVars.Fkvarlimitneg (variable) and Set\_Maxkvarneg. |
| Property-public | MinModelVoltagePU | PA VminPu. |
| Property-public | pf\_wp\_nominal | PA Set\_pf\_wp\_nominal. |
| Property-public | IrradianceNow | PA ShapeFactor.re. |
| Property-public | Pmin | PA Get\_Pmin. |
| Property-public | Pmax | PA Get\_Pmax. |
| Property-public | qMaxInj | PA Get\_qMaxInj. |
| Property-public | qMaxAbs | PA Get\_qMaxAbs. |
| Property-public | acVmin | PA Get\_acVmin. |
| Property-public | acVmax | PA Get\_acVmax. |
| Property-public | acVnom | PA Get\_acVnom. |
| Property-public | pMaxUnderPF | PA Get\_pMaxUnderPF. |
| Property-public | pMaxOverPF | PA Get\_pMaxOverPF. |
| Property-public | pMaxCharge | PA Get\_Zero. |
| Property-public | apparentPowerChargeMax | PA Get\_Zero. |
| Property-public | UsingCIMDynamics | PA Get\_CIMDynamicMode. |

### TStorageObj

| *TStorageObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Implements the following properties/methods as in TLoadObj, *TIndMach012Obj, TPVSystemObj and TGeneratorObj*:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties 9. CalcDailyMult 10. CalcDutyMult 11. CalcYearlyMult 12. GetPropertyValue 13. CalcYPrimMatrix 14. CalcInjCurrentArray 15. GetTerminalCurrents 16. DoHarmonicMode 17. InitHarmonics 18. Integrate 19. StickCurrInTerminalArray 20. WriteTraceRecord 21. Set\_ConductorClosed 22. VariableName 23. NumVariables 24. GetAllVariables 25. Get\_Variable 26. Set\_Variable 27. VariableName 28. Randomize 29. InitStateVars 30. IntegrateStates 31. CalcVTerminalPhase 32. DoDynamicMode 33. DoUserModel 34. SetDragHandRegister 35. Get\_PresentkW 36. Get\_Presentkvar 37. Get\_PresentkV 38. Set\_PresentkV 39. Set\_Presentkvar 40. Set\_PresentkW 41. Set\_PowerFactor 42. ComputeInverterPower 43. ComputekWkvar 44. Set\_pf\_wp\_nominal 45. Get\_Varmode 46. Set\_Varmode 47. Get\_VWmode 48. Set\_VVmode 49. Get\_VVmode 50. Set\_DRCmode 51. Get\_DRCmode 52. Set\_AVRmode 53. Get\_AVRmode 54. Set\_VWmode 55. kWOut\_Calc 56. Get\_WPmode 57. Set\_WPmode 58. Get\_WVmode 59. Set\_WVmode 60. Get\_InverterON 61. Set\_InverterON 62. Get\_VarFollowInverter 63. Set\_VarFollowInverter 64. Set\_Maxkvar 65. Set\_Maxkvarneg 66. ResetRegisters 67. TakeSample | | |
| Method- private | ComputePresentkW | Calculates the present power output considering the model features. |
| Method- private | ComputeDCkW | Computes actual DC power (kW) to Update Storage State of Charge (SoC). |
| Method- private | CalcStorageModelContribution | Calculates Storage element current and adds it properly into the injcurrent array routines may also compute ITerminal (ITerminalUpdated flag). |
| Method- private | DoConstantPQStorageObj | Compute total terminal current for Constant PQ model. |
| Method- private | DoConstantZStorageObj | Compute total terminal current for constant Z model. |
| Method- private | DoDynaModel | Implements the routine for doing the user written dynamics model. |
| Method- private | CheckStateTriggerLevel | This is where we set the state of the Storage element. |
| Method- private | UpdateStorage | Updates Storage elements based on present kW and IntervalHrs variable. |
| Property- private | NormalizeToTOD | Normalizes time to a floating-point number representing time of day If Hour > 24, time should be 0 to 24. |
| Property- private | InterpretState | Interprets the present state of the storage device (charging, discharging, idle). |
| Property- private | DecodeState | Returns the name of the storage state (charging, discharging, idle) as string. |
| Property- private | Get\_kvarRequested | Returns the present kvar requested. |
| Property- private | Get\_kWRequested | Returns the present kW requested. |
| Method- private | Set\_kW | Sets the present kW rating. |
| Property- private | Get\_kW | Returns the present kW rating. |
| Method- private | Set\_kWRequested | Sets the present kW requested. |
| Method- private | Set\_kvarRequested | Sets the present kvar requested. |
| Method- private | Set\_StorageState | Sets the storage state using the given integer. |
| Method- private | Set\_pctkWOut | Sets the discharging efficiency for the storage. |
| Method- private | Set\_pctkWIn | Sets the charging efficiency for the storage. |
| Property- private | Get\_DCkW | Returns the DC power rating for the storage. |
| Property- private | Get\_kWTotalLosses | Returns total losses: Idling + inverter + interface losses. |
| Property- private | Get\_InverterLosses | Returns the inverter losses. |
| Property- private | Get\_kWIdlingLosses | Returns Idling losses. |
| Property- private | Get\_kWChDchLosses | Returns the DC/AC interface losses. |
| Method- private | Update\_EfficiencyFactor | Updates the efficiency factor in time. |
| Method- private | Set\_StateDesired | Sets the state for the storage device using the given integer. |
| Property- private | Get\_kWDesired | Returns the kW desired, normally set by a controller. |
| Method- private | Set\_pctkWrated | Sets the kW rating as percentage of the nominal kW. |
| Property- private | Get\_CutOutkWAC | Gets the cutout limit for the inverter generation threshold. |
| Property- private | Get\_CutInkWAC | Gets the cut in limit for the inverter charging threshold. |
| Method-public | SetNominalStorageOutput | Sets the power output for the storage according to its present features. |

### TUPFCObj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *TUPFCObj* | | | | |
| *Type-access* | | *Command* | *Description – (Specific class) otherwise generic* | |
| Implements the following properties/methods as in TLoadObj, *TIndMach012Obj, TPVSystemObj and TGeneratorOb*:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties 9. Get\_Variable 10. Set\_Variable 11. GetPropertyValue 12. NumVariables 13. GetAllVariables 14. VariableName | | | | |
| Property- private | GetinputCurr | | | Calculates the input current to absorb reactive power from UPFC. |
| Property- private | GetOutputCurr | | | Calculates the output current for the UPFC device. |
| Property- private | CalcUPFCPowers | | | Calculates the equivalent power output for the UPFC device. |
| Property- private | CalcUPFCLosses | | | Calculates the Active power losses at the input of the device by using the Load powers, the approach is based in the data provided. |
| Method- public | UploadCurrents | | | Uploads the input/output currents when commanded by the controller. |
| Property- public | CheckStatus | | | Checks if the UPFC control needs an update, returns true if so. |

### TVCCSObj

| *TVCCSObj* | | | | |
| --- | --- | --- | --- | --- |
| *Type-access* | | *Command* | *Description – (Specific class) otherwise generic* | |
| Implements the following properties/methods as in TLoadObj, *TIndMach012Obj, TPVSystemObj and TGeneratorOb*:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties 9. Get\_Variable 10. Set\_Variable 11. GetPropertyValue 12. NumVariables 13. GetAllVariables 14. VariableName 15. InitStateVars 16. IntegrateStates | | | | |
| Method- private | InitPhasorStates | | | Initializes the history terms for HW model source convention. |
| Method- private | IntegratePhasorStates | | | Integrates the phasor states for the next simulation iteration. |
| Method- private | ShutoffInjections | | | Support for DYNAMICMODE, NB: in phasor mode, use load convention for OpenDSS-X, stops injecting if the terminal opens. |
| Method- private | UpdateSequenceVoltage | | | Updates the register containing the sequence voltages at the terminal. |

### TVSConverterObj

|  |  |  |
| --- | --- | --- |
| *TVSConverterObj* | | |
| *Type-access* | *Command* | *Description – (Specific class) otherwise generic* |
| Implements the following properties/methods as in TLoadObj, *TIndMach012Obj, TPVSystemObj and TGeneratorOb*:   1. RecalcElementData 2. CalcYPrim 3. MakePosSequence 4. InjCurrents 5. GetInjCurrents 6. GetCurrents 7. InitPropertyValues 8. DumpProperties 9. Get\_Variable 10. Set\_Variable 11. GetPropertyValue | | |

## Power Delivery Elements

Power Delivery Elements (PDE) are elements derived from TDSSClass and NamedObject classes. They derive into other subclasses to differentiate them from PCE, the inheritance trees for the classes and objects within PDE are shown in Figure 4 and Figure 5 respectively.

Chart

Description automatically generated with low confidence

Figure . PDE class inheritance tree

### TPDClass

|  |  |  |
| --- | --- | --- |
| *TPDClass* | | |
| *Type-access* | *Command* | *Description* |
| Property-protected | ClassEdit | Continue parsing with contents of Parser. |
| Method-protected | ClassMakeLike | Same as in TPCClass. |
| Method-protected | CountProperties | Same as in TPCClass. |
| Method-protected | DefineProperties | Same as in TPCClass. |

### TLine

| *TLine* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Method-private | DoRmatrix | Calculates the R matrix for the line. |
| Method-private | DoXmatrix | Calculates the X matrix for the line. |
| Method-private | DoCmatrix | Calculates the C matrix for the line. |
| Method-protected | DefineProperties | Add Properties of this class to propName. |
| Property-protected | MakeLike | Same as in TLoadObj. |
| Property-protected | Edit | Same as in 3.1.4. |
| Property-protected | Init | Same as in 3.1.4. |
| Property-protected | NewObject | Same as in 3.1.4. |

### TAutoTrans

|  |  |  |  |
| --- | --- | --- | --- |
| *TAutoTrans* | | | |
| *Type-access* | | *Command* | *Description* |
| Implements the following properties/methods as in TLine:   1. DefineProperties 2. MakeLike 3. Edit 4. Init 5. NewObject | | | |
| Method-private | SetActiveWinding | | Sets the active winding using the given integer. |
| Method-private | InterpretAutoConnection | | Interprets the connection for the active winding. |
| Method-private | InterpretAllConns | | Interprets the connection for all windings. |
| Method-private | InterpretAllBuses | | Assigns buses to all windings. |
| Method-private | InterpretAllTaps | | Assigns active tap position to all windings. |
| Method-private | InterpretAllkVRatings | | Assigns kV ratings to all windings. |
| Method-private | InterpretAllkVARatings | | Assigns kVA ratings to all windings. |
| Method-private | InterpretAllRs | | Assigns Percent ac resistance to all windings. |
| Property-private | TrapZero | | Trap for catching 0 to avoid numeric overflows during calculations. |
| Property-private | InterpretLeadLag | | Routine expecting all winding bus connections expressed in one array of strings. |

### TCapacitor

|  |  |  |  |
| --- | --- | --- | --- |
| *TCapacitor* | | | |
| *Type-access* | | *Command* | *Description* |
| Implements the following properties/methods as in TLine:   1. DefineProperties 2. MakeLike 3. Edit 4. Init 5. NewObject | | | |
| Method-private | DoCmatrix | | Calculates C matrix for the capacitor. |
| Method-private | InterpretConnection | | Interprets the connection at the capacitor’s terminal(s). |
| Method-private | CapSetBus1 | | Sets the name and connection for cap’s bus1. |

### TFault

| *TFault* | | | |
| --- | --- | --- | --- |
| *Type-access* | | *Command* | *Description* |
| Implements the following properties/methods as in TLine:   1. DefineProperties 2. MakeLike 3. Edit 4. Init 5. NewObject | | | |
| Method-private | DoGmatrix | | Calculates Conductance matrix for the capacitor. |
| Method-private | FltSetBus1 | | Sets the name and connection for fault’s bus1. |

### TGICTransformer

|  |  |  |  |
| --- | --- | --- | --- |
| *TGICTransformer* | | | |
| *Type-access* | | *Command* | *Description* |
| Implements the following properties/methods as in TLine:   1. DefineProperties 2. MakeLike 3. Edit 4. Init 5. NewObject | | | |
| Method-private | GICTransSetBusH | | Sets the name and connection for fault’s bus1. |
| Method-private | GICTransSetBusX | | Sets the name and connection for fault’s bus3. |

### TReactor

|  |  |  |  |
| --- | --- | --- | --- |
| *TReactor* | | | |
| *Type-access* | | *Command* | *Description* |
| Implements the following properties/methods as in TLine:   1. DefineProperties 2. MakeLike 3. Edit 4. Init 5. NewObject | | | |
| Method-private | Domatrix | | Calculates admittance matrix for the element. |
| Method-private | InterpretConnection | | Interprets the connection at the element’s terminal(s). |
| Method-private | ReactorSetbus1 | | Sets the name and connection for element’s bus1. |

### TTransf

| *TTransf* | | | |
| --- | --- | --- | --- |
| *Type-access* | | *Command* | *Description* |
| Implements the following properties/methods as in TLine:   1. DefineProperties 2. MakeLike 3. Edit 4. Init 5. NewObject | | | |
| Method-private | SetActiveWinding | | Sets the active winding using the given integer. |
| Method-private | InterpretAutoConnection | | Interprets the connection for the active winding. |
| Method-private | InterpretAllConns | | Interprets the connection for all windings. |
| Method-private | InterpretAllBuses | | Assigns buses to all windings. |
| Method-private | InterpretAllTaps | | Assigns active tap position to all windings. |
| Method-private | InterpretAllkVRatings | | Assigns kV ratings to all windings. |
| Method-private | InterpretAllkVARatings | | Assigns kVA ratings to all windings. |
| Method-private | InterpretAllRs | | Assigns Percent ac resistance to all windings. |
| Property-private | TrapZero | | Trap for catching 0 to avoid numeric overflows during calculations. |
| Property-private | InterpretLeadLag | | Routine expecting all winding bus connections expressed in one array of strings. |

A picture containing chart

Description automatically generated

Figure . PDE object inheritance tree

### TPDElement

| *TPDElement* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Property-private | Get\_ExcessKVANorm | Returns the kVAs exceeding the normal kVA rating(s) of the element. |
| Property-private | Get\_ExcessKVAEmerg | Returns the kVAs exceeding the emergency kVA rating(s) of the element. |
| Method-public | InitPropertyValues | Same as in 3.1.6. |
| Method-public | GetCurrents | Same as in 3.1.6. |
| Method-public | CalcFltRate | Calculates failure rates for section and buses. |
| Method-public | AccumFltRate | Calculates the accumulated failure rates for section and buses. |
| Method-public | CalcNum\_Int | Calculates the number of Interruptions in forward sweep. |
| Method-public | CalcCustInterrupts | Calculates the number of Interruptions per customer in forward sweep. |
| Method-public | ZeroReliabilityAccums | Zero out reliability accumulators. |
| Property-public | ExcesskVANorm | PA Get\_ExcesskVANorm. |
| Property-public | ExcesskVAEmerg | PA Get\_ExcesskVAEmerg. |

### TLineObj

| *TLineObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Method-private | FMakeZFromGeometry | Make new Z, Zinv, Yc, etc. from line geometry reference. |
| Method-private | KillGeometrySpecified | Indicate No Line Geometry specification if this is called. |
| Method-private | FMakeZFromSpacing | make new Z, Zinv, Yc, etc. from linespacing reference. |
| Method-private | KillSpacingSpecified | Indicate No Line spacing specification if this is called. |
| Method-private | ClearYPrim | Clears the Y primitive memory spaces. |
| Method-private | ResetLengthUnits | Resets the length units to the default. |
| Method-private | UpdatePDProperties | Updates inherited properties. |
| Property-private | NumConductorData | Defines the number of conductors for the element. |
| Property-private | FetchConductorData | Gets the conductor data from the conductor data reference. |
| Method-private | ReallocZandYcMatrices | Reallocate Z and Y matrices in memory (possible expansion). |
| Method-private | DoLongLine | Long Line Correction for 1=phase. |
| Method-private | ConvertZinvToPosSeqR | For GIC analysis, primarily. Uses only real part of Z. |
| Method-public | GetLosses | Overrides the more specific class specification. Calculates the losses for the area. |
| Method-public | GetSeqLosses | Overrides the more specific class specification. Calculates the losses for the area using sequential components. |
| Method-public | RecalcElementData | See 3.1.9. |
| Method-public | CalcYPrim | See 3.1.9. |
| Method-public | MakePosSequence | See 3.1.9. |
| Property-public | MergeWith | Merge this line with another line and disable the other line. |
| Method-public | UpdateControlElements | Updates the elements controlled/assigned to the controllers in the model. |
| Property-public | GetPropertyValue | See 3.1.9. |
| Method-public | InitPropertyValues | See 3.1.9. |
| Method-public | DumpProperties | See 3.1.9. |
| Method-public | FetchLineCode | Loads the line code given in the argument into the active line. |
| Method-public | FetchGeometryCode | Loads the line geometry given in the argument into the active line. |
| Method-public | FetchLineSpacing | Loads the line spacing given in the argument into the active line. |
| Method-public | FetchWireList | Loads the wire data given in the argument into the active line. |
| Method-public | FetchCNCableList | Loads the CNCable data given in the argument into the active line. |
| Method-public | FetchTSCableList | Loads the TSCable data given in the argument into the active line. |
| Method-public | CalcFltRate | Overrides the generic class declaration. Calculates failure rates for section and buses. |

### TAutoTransObj

| *TAutoTransObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TLineObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence | | |
| Property-private | Get\_PresentTap | Returns the present tap for the active winding. |
| Method-private | Set\_PresentTap | Sets the present tap for the active winding. |
| Property-private | Get\_MinTap | Returns the minimum tap for the active winding. |
| Property-private | Get\_MaxTap | Returns the maximum tap for the active winding. |
| Property-private | Get\_TapIncrement | Returns the tap increment for the active winding. |
| Property-private | Get\_BaseVoltage | Returns the base voltage (LN) for the active winding. |
| Property-private | Get\_BasekVLL | Returns the base voltage (LL) for the active winding. |
| Property-private | Get\_NumTaps | Returns the number of taps for the active winding. |
| Property-private | Get\_WdgResistance | Returns the resistance for the active winding. |
| Property-private | Get\_WdgConnection | Returns the connection type for the active winding. |
| Property-private | Get\_WdgkVA | Returns the nominal kVA for the active winding. |
| Property-private | Get\_Xsc | Returns the reactance for the active winding. |
| Property-private | Get\_WdgYPPM | Returns the parts per million of transformer winding VA rating connected to ground to protect against accidentally floating a winding without a reference. Applies for the active winding. |
| Method-private | CalcY\_Terminal | Calculates the Y primitive at the active winding. |
| Method-private | GICBuildYTerminal | Calculates the Y primitive at the active winding at the given frequency. |
| Method-private | BuildYPrimComponent | Routine used to add every element of Y\_terminal into Yprim somewhere. |
| Method-private | FetchXfmrCode | Imports the features of the XfmrCode assigned into the active element. |
| Property-private | GeTAutoWindingCurrentsResult | Returns string mag, angle for the active winding. |
| Method-private | SetBusAuto | For winding 2 set all nodes on second end of winding to same as 1st value so all neutral ends of common winding get connected to same neutral node. |
| Method-protected | SetTermRef | Sets an array which maps the two conductors of each winding to the phase and neutral conductors of the AutoTrans according to the winding connection. |
| Method-public | SetNumWindings | Sets the number of windings for the element, creates the windings objects needed. |
| Method-public | SetNodeRef | Overrides the more generic class implementation. fixup noderefs for series winding of AutoTrans. Warning \*\*\*\* Magic happens here Redefine 2nd node of Series winding to same as first node of 2nd winding (common winding). |
| Method-public | GetCurrents | Get present current values of terminal. |
| Property-public | RotatePhases | For Delta connections or Line-Line voltages. |
| Property-public | PresentTap | PA Get\_PresentTap and Set\_PresentTap. |
| Property-public | Mintap | PA Get\_MinTap. |
| Property-public | Maxtap | PA Get\_MaxTap. |
| Property-public | TapIncrement | PA Get\_TapIncrement. |
| Property-public | BaseVoltage | PA Get\_BaseVoltage. |
| Property-public | BasekVLL | PA Get\_BasekVLL. |

### TCapacitorObj

| *TCapacitorObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TLineObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. MakePosSequence | | |
| Property-private | get\_States | Returns the present states for the capacitor bank. |
| Method-private | Set\_States | Sets the present states for the capacitor bank. |
| Method-private | set\_LastStepInService | Forces the last step in service to be a certain value. |
| Method-private | ProcessHarmonicSpec | Interprets and loads the assigned harmonic spectrum into the capacitor. |
| Method-private | ProcessStatesSpec | Interprets and loads the assigned states array for the cap bank. |
| Method-private | MakeYprimWork | Call this routine only if step is energized. |
| Method-private | set\_NumSteps | Sets the number of steps for the cap bank (1=kvar, 2=Cuf, 3=Cmatrix). |
| Property-public | AddStep | Connects the next capacitor from the bank. |
| Property-public | SubtractStep | Disconnects the next capacitor from the bank. |
| Property-public | AvailableSteps | Returns the number of available steps for connection. |
| Method-public | FindLastStepInService | Finds the last cap in service within the cap bank. |
| Property-public | NumSteps | PA FNumSteps (variable) and set\_NumSteps. |
| Property-public | States | PA get\_States and set\_States. |
| Property-public | Totalkvar | PA FTotalkvar (variable). |
| Property-public | NomKV | PA kvrating (variable). |
| Property-public | LastStepInService | PA FLastStepInService (variable) and set\_LastStepInService. |
| Property-public | NumTerminals | PA NumTerm (variable). |

### TFaultObj

| *TFaultObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TLineObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. MakePosSequence | | |
| Property-private | FaultStillGoing | Returns a Boolean indicating if the fault is still active. |
| Method-public | CheckStatus | Checks and sets the fault status depending on the simulation mode and fault features. |
| Method-public | Reset | Resets (clear) the fault. |

### TGICTransformerObj

|  |  |  |
| --- | --- | --- |
| *TGICTransformerObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TLineObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. MakePosSequence | | |
| Method-public | WriteVarOutputRecord | Adds a record to the output file based on present GIC. |

### TReactorObj

|  |  |  |
| --- | --- | --- |
| *TReactorObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TLineObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. MakePosSequence | | |

### TTransfObj

| *TTransfObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TLineObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence | | |
| Property-private | Get\_PresentTap | Returns the present tap for the active winding. |
| Method-private | Set\_PresentTap | Sets the present tap for the active winding. |
| Property-private | Get\_MinTap | Returns the minimum tap for the active winding. |
| Property-private | Get\_MaxTap | Returns the maximum tap for the active winding. |
| Property-private | Get\_TapIncrement | Returns the tap increment for the active winding. |
| Property-private | Get\_BaseVoltage | Returns the base voltage (LN) for the active winding. |
| Property-private | Get\_BasekVLL | Returns the base voltage (LL) for the active winding. |
| Property-private | Get\_NumTaps | Returns the number of taps for the active winding. |
| Property-private | Get\_WdgResistance | Returns the resistance for the active winding. |
| Property-private | Get\_WdgConnection | Returns the connection type for the active winding. |
| Property-private | Get\_WdgkVA | Returns the nominal kVA for the active winding. |
| Property-private | Get\_Xsc | Returns the reactance for the active winding. |
| Property-private | Get\_WdgYPPM | Returns the parts per million of transformer winding VA rating connected to ground to protect against accidentally floating a winding without a reference. Applies for the active winding. |
| Method-private | CalcY\_Terminal | Calculates the Y primitive at the active winding. |
| Method-private | GICBuildYTerminal | Calculates the Y primitive at the active winding at the given frequency. |
| Method-private | BuildYPrimComponent | Routine used to add every element of Y\_terminal into Yprim somewhere. |
| Method-private | AddNeutralToY | Add neutral admittance at the given frequency. |
| Method-private | FetchXfmrCode | Imports the features of the XfmrCode assigned into the active element. |
| Method-protected | SetTermRef | Sets an array which maps the two conductors of each winding to the phase and neutral conductors of the AutoTrans according to the winding connection. |
| Method-public | SetNumWindings | Sets the number of windings for the element, creates the windings objects needed. |
| Property-public | GetWindingCurrentsResult | Returns all winding currents in string. |
| Method-public | GetWindingVoltages | Voltages across indicated winding. |
| Method-public | GetAllWindingCurrents | Returns all Winding currents in complex array. |
| Property-public | PresentTap | PA Get\_PresentTap and Set\_PresentTap. |
| Property-public | Mintap | PA Get\_MinTap. |
| Property-public | Maxtap | PA Get\_MaxTap. |
| Property-public | TapIncrement | PA Get\_TapIncrement. |
| Property-public | BaseVoltage | PA Get\_BaseVoltage. |
| Property-public | BasekVLL | PA Get\_BasekVLL. |

## Controls

Power Delivery Elements (PDE) are elements derived from TDSSClass and NamedObject classes. They derive into other subclasses to differentiate them from PCE and PDE, the inheritance trees for the classes and objects within PDE are shown in Figure 6 and Figure 7Figure 3 respectively.

Diagram

Description automatically generated

Figure . Control class inheritance tree

### TControlClass

|  |  |  |
| --- | --- | --- |
| *TControlClass* | | |
| *Type-access* | *Command* | *Description* |
| Property-protected | ClassEdit | Continue parsing with contents of Parser. |
| Method-protected | ClassMakeLike | Same as in TPCClass. |
| Method-protected | CountProperties | Same as in TPCClass. |
| Method-protected | DefineProperties | Same as in TPCClass. |

### TCapControl

|  |  |  |
| --- | --- | --- |
| *TCapControl* | | |
| *Type-access* | *Command* | *Description* |
| Method-protected | DefineProperties | See 3.2.1. |
| Property-protected | MakeLike | See 3.2.2. |
| Property-public | Edit | See 3.1.4. |
| Property-public | NewObject | See 3.1.4. |

### TESPVLControl

|  |  |  |
| --- | --- | --- |
| *TESPVLControl* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | |

### TExpControl

|  |  |  |  |
| --- | --- | --- | --- |
| *TExpControl* | | | |
| *Type-access* | | *Command* | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | | |
| Method-public | UpdateAll | | Called at end of main power flow solution loop, updates all the ExpControl devices in the circuit. |

### TGenDispatcher

|  |  |  |
| --- | --- | --- |
| *TGenDispatcher* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | |

### TInvControl

|  |  |  |  |
| --- | --- | --- | --- |
| *TInvControl* | | | |
| *Type-access* | | *Command* | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | | |
| Property-public | GetXYCurve | | Imports the features of the XY curve used for indicating the control curve. |
| Method-public | UpdateAll | | Called at end of main power flow solution loop, updates all the inv control devices in the model. |

### TRecloser

|  |  |  |
| --- | --- | --- |
| *TRecloser* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | |

### TRegControl

|  |  |  |
| --- | --- | --- |
| *TRegControl* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | |

### TRelay

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *TRelay* | | | | |
| *Type-access* | *Command* | | | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | | | |
| Property-public | | GetTccCurve | Imports the features of the TCC curve describing the protection’s behavior. | |

### TStorageController

|  |  |  |
| --- | --- | --- |
| *TStorageController* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | |

### TSwtControl

| *TSwtControl* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | |

### TUPFCControl

|  |  |  |
| --- | --- | --- |
| *TUPFCControl* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | |

### TFuse

|  |  |  |
| --- | --- | --- |
| *TFuse* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControl:   1. DefineProperties 2. MakeLike 3. Edit 4. NewObject | | |

Diagram

Description automatically generated with medium confidence

Figure . Control object inheritance tree

### TControlElem

|  |  |  |
| --- | --- | --- |
| *TControlElem* | | |
| *Type-access* | *Command* | *Description* |
| Method-protected | Set\_ControlledElement | Stores a pointer to target circuit element. |
| Method-protected | RemoveSelfFrom  ControlelementList | Remove this control from the control element list of the designated element. |
| Method-protected | Set\_MonitoredElement | Stores a pointer to the monitored element. |
| Method-public | Sample | Sample control quantities and set action times in Control Queue. |
| Method-public | DoPendingAction | Implements the routine for doing the action that is pending from last sampling. |
| Method-public | Reset | Implements the reference for resetting the control, it is expected to be overridden by more specific classes. |
| Property-public | ControlledElement | PA FControlledElement (variable) and Set\_ControlledElement. |
| Property-public | MonitoredElement | PA FMonitoredElement (variable) and Set\_MonitoredElement. |

### TCapControlObj

|  |  |  |
| --- | --- | --- |
| *TCapControlObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TLineObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence | | |
| Property-private | Get\_Capacitor | Returns the pointer to the controlled capacitor. |
| Property-private | NormalizeToTOD | Normalize time to a floating-point number representing time of day if Hour > 24. Resulting time should be 0:00+ to 24:00 inclusive. |
| Method-private | Set\_PendingChange | Sets the flag for indicating that there is a pending change to be performed. |
| Property-private | Get\_PendingChange | Returns the value of the flag indicating if there is a pending change. |
| Method-private | GetControlVoltage | Gets the Voltage used for voltage control based on specified options. |
| Method-private | GetControlCurrent | Gets the current to control on based on type of control specified. |
| Method-private | GetBusVoltages | Gets the voltages at the bus specified. |
| Method-public | Sample | Samples control quantities and set action times in Control Queue. |
| Method-public | DoPendingAction | Do the action that is pending from last sample. |
| Method-public | Reset | Resets the controller to the initial defined state. |
| Property-public | This\_Capacitor | PA Get\_Capacitor. |
| Property-public | PendingChange | PA Get\_PendingChange and Set\_PendingChange. |

### TESPVLControlObj

| *TESPVLControlObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset | | |
| Property-private | MakeLocalControlList | Search through the entire circuit for enabled generators and add them to the list of controlled elements. |

### TExpControlObj

|  |  |  |
| --- | --- | --- |
| *TExpControlObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset 12. Set\_PendingChange 13. Get\_PendingChange | | |
| Property-private | ReturnElementsList | Returns the list of elements controlled by this controller, the list is delivered as string. |
| Method-private | UpdateExpControl | Updates the state variables at the present simulation time/mode. |
| Method-public | GetInjCurrents | See 3.1.9. |
| Method-public | GetCurrents | See 3.1.9. |
| Property-public | MakePVSystemList | Allocates memory and creates the list of PV systems to be controlled. |
| Property-public | PendingChange | PA Get\_PendingChange and Set\_PendingChange. |
| Property-public | DERNameList | PA FDERNameList (variable). |
| Property-public | VregTau | PA FVregTau (variable). |
| Property-public | QVSlope | PA FSlope (variable). |
| Property-public | VregMin | PA FVregMin (variable). |
| Property-public | VregMax | PA FVregMax (variable). |
| Property-public | QMaxLead | PA FQMaxLead (variable). |
| Property-public | QMaxLag | PA FQMaxLag (variable). |
| Property-public | TResponse | PA FTResponse (variable). |

### TGenDispatcherObj

| *TGenDispatcherObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset 12. Set\_PendingChange 13. Get\_PendingChange | | |
| Property-public | MakeGenList | Allocates memory for creating a list of generators to be controlled by this controller. |

### TInvControlObj

| *TInvControlObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj and TExpControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset 12. Set\_PendingChange 13. Get\_PendingChange 14. GetCurrents 15. GetInjCurrents | | |
| Property-private | InterpretAvgVWindowLen | Interprets and assigns the Volt-watt control window length for the controller operation. |
| Property-private | InterpretDRCAvgVWindowLen | Interprets and assigns the DRC control window length for the controller operation. |
| Property-private | ReturnElementsList | Returns the list of controlled elements as string. |
| Method-private | UpdateInvControl | Updates the inv controls state by adding the present power flow solution voltage to the rolling average window. |
| Method-private | UpdateDERParameters | Updates the controlled devices according to the controller state. |
| Method-private | CalcVoltWatt\_watts | Calculates power output in VW mode. |
| Method-private | CalcQVVcurve\_desiredpu | Calculates reactive power output in VV mode. |
| Method-private | CalcQWPcurve\_desiredpu | Calculates reactive power output in WP mode. |
| Method-private | CalcQWVcurve\_desiredpu | Calculates reactive power output in WV mode. |
| Method-private | CalcQDRC\_desiredpu | Calculates reactive power output in DRC mode. |
| Method-private | CalcQAVR\_desiredpu | Calculates reactive power output in AVR mode. |
| Method-private | Check\_Qlimits | Checks if the present values are within allowed limits. |
| Method-private | Check\_Qlimits\_WV | Checks if the present values are within allowed limits (WV mode). |
| Method-private | Calc\_PQ\_WV | Calculates active and reactive powers in WV mode. |
| Method-private | Calc\_QHeadRoom | Calculates the present headroom for reactive power. |
| Method-private | CalcVoltVar\_vars | Calculates the reactive power for VV mode. |
| Method-private | CalcAVR\_vars | Calculates the reactive power for AVR mode. |
| Method-private | CalcWATTPF\_vars | Calculates the reactive power for WATTPF mode. |
| Method-private | CalcWATTVAR\_vars | Calculates the reactive power for WATTVAR mode. |
| Method-private | CalcDRC\_vars | Calculates the reactive power for DRC mode. |
| Method-private | CalcVVDRC\_vars | Calculates the reactive power for VVDRC mode. |
| Method-private | CalcLPF | Applies the LPF: Returns value is in kvar for VARS. Return value is in puPmpp for WATTS. |
| Method-private | CalcRF | Applies the Rise/Fall limiting function. |
| Method-private | Calc\_PBase | Calculates the active power base. |
| Method-private | Check\_Plimits | Calculates power limits. |
| Method-private | CalcPVWcurve\_limitpu | Calculates PVW curve limits in pu. |
| Method-private | GetmonVoltage | Gets the voltage at the point of connection of the DER. |
| Method-private | Change\_deltaQ\_factor | Updates delta Q factor. |
| Method-private | Change\_deltaP\_factor | Updates delta P factor. |
| Property-public | MakeDERList | Allocates memory for storing the list of DER to be controlled by this controller. |
| Property-public | PendingChange | PA Get\_PendingChange and Set\_PendingChange. |
| Property-public | Mode | PA ControlMode (variable). |
| Property-public | CombiMode | PA CombiControlMode (variable). |
| Property-public | DERNameList | PA FDERNameList (variable). |
| Property-public | vvc\_curve1 | PA Fvvc\_curvename (variable). |
| Property-public | hysteresis\_offset | PA Fvvc\_curveOffset (variable). |
| Property-public | voltage\_curvex\_ref | PA FVoltage\_CurveX\_ref (variable). |
| Property-public | avgwindowlen | PA FRollAvgWindowLength (variable). |
| Property-public | voltwatt\_curve | PA Fvoltwatt\_curvename (variable). |
| Property-public | voltwattCH\_curve | PA FvoltwattCH\_curvename (variable). |
| Property-public | DbVMin | PA FDbVMin (variable). |
| Property-public | DbVMax | PA FDbVMax (variable). |
| Property-public | ArGraLowV | PA FArGraLowV (variable). |
| Property-public | DynReacavgwindowlen | PA FDRCRollAvgWindowLength (variable). |
| Property-public | DeltaQ\_factor | PA FDeltaQ\_factor (variable). |
| Property-public | VoltageChangeTolerance | PA FVoltageChangeTolerance (variable). |
| Property-public | VarChangeTolerance | PA FVarChangeTolerance (variable). |
| Property-public | VoltwattYAxis | PA FVoltwattYAxis (variable). |
| Property-public | LPFTau | PA FLPFTau (variable). |
| Property-public | RiseFallLimit | PA FRiseFallLimit (variable). |
| Property-public | DeltaP\_factor | PA FDeltaP\_factor (variable). |
| Property-public | RefReactivePower | PA FReacPower\_ref (variable). |
| Property-public | ActivePChangeTolerance | PA FActivePChangeTolerance (variable). |
| Property-public | monVoltageCalc | PA FMonBusesPhase (variable). |
| Property-public | monBus | PA FMonBusesNameList (variable). |
| Property-public | monBusVbase | PA FMonBusesVbase (variable). |
| Property-public | v\_setpoint | PA Fv\_setpoint (variable). |

### TRecloserObj

|  |  |  |
| --- | --- | --- |
| *TRecloserObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj and TExpControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset 12. Set\_PendingChange 13. Get\_PendingChange 14. GetCurrents 15. GetInjCurrents | | |
| Method-private | InterpretRecloserState | Interprets the recloser state (open/closed/trip). |
| Property-private | get\_State | Returns the control state. |
| Method-private | set\_State | Sets the control state. |
| Property-private | get\_NormalState | Returns the control normal state. |
| Method-private | set\_NormalState | Sets the control normal state. |
| Property-public | PresentState | PA get\_State and set\_State. |
| Property-public | NormalState | PA get\_NormalState and set\_NormalState. |

### TRegControlObj

|  |  |  |
| --- | --- | --- |
| *TRegControlObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj and TExpControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset 12. Set\_PendingChange 13. Get\_PendingChange 14. GetCurrents 15. GetInjCurrents | | |
| Property-private | Get\_Transformer | Returns a pointer to the controlled transformer. |
| Property-private | Get\_Winding | Returns the index of the active winding. |
| Property-private | Get\_MinTap | Returns the minimum tap value. |
| Property-private | Get\_MaxTap | Returns the maximum tap value. |
| Property-private | Get\_TapIncrement | Returns the tap increment value. |
| Property-private | Get\_NumTaps | Returns the number of taps for the active winding. |
| Property-private | Get\_TapNum | Returns the present tap number. |
| Method-private | RegWriteTraceRecord | Adds a new line to the trace record using the present values. |
| Method-private | RegWriteDebugRecord | Adds a new line to the debug record using the present values. |
| Method-private | set\_PendingTapChange | Sets the flags indicating that there is a tap change pending for being applied. |
| Method-private | AtLeastOneTap | Called in STATIC mode. Changes 70% of the way but at least one tap, subject to maximum allowable tap change. |
| Property-private | ComputeTimeDelay | Computes time delay for the next tap change (if any). |
| Property-private | GetControlVoltage | Gets the control voltage according to the controller’s configuration. |
| Property-private | Set\_TapNum | Sets the present tap for the active winding. |
| Method-public | SaveWrite | Overrides standard SaveWrite. Regcontrol structure not conducive to standard means of saving. |
| Property-public | Transformer | PA Get\_Transformer. |
| Property-public | TrWinding | pA Get\_Winding. |
| Property-public | PendingTapChange | pA FPendingTapChange (variable) and set\_PendingTapChange. |

### TRelayObj

|  |  |  |
| --- | --- | --- |
| *TRelayObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj and TExpControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset 12. Set\_PendingChange 13. Get\_PendingChange 14. GetCurrents 15. GetInjCurrents | | |
| Method-private | InterpretRelayState | Interprets the relay state (open/closed/trip). |
| Property-private | get\_State | Returns the control state. |
| Method-private | set\_State | Sets the control state. |
| Property-private | get\_NormalState | Returns the control normal state. |
| Method-private | set\_NormalState | Sets the control normal state. |
| Method-private | InterpretRelayType | Interprets the relay type (current, voltage, etc.) |
| Method-private | OvercurrentLogic | Implements the overcurrent relay control logic. |
| Method-private | VoltageLogic | Implements the voltage relay control logic. |
| Method-private | RevPowerLogic | Implements the reverse power relay control logic. |
| Method-private | NegSeq46Logic | Implements the negative sequence 46 power relay control logic. |
| Method-private | NegSeq47Logic | Implements the negative sequence 47 power relay control logic. |
| Method-private | GenericLogic | Implements the generic relay control logic. |
| Method-private | DistanceLogic | Implements the distance relay control logic. |
| Method-private | TD21Logic | Implements the TD21 relay control logic. |
| Method-private | DirectionalOvercurrentLogic | Implements the directional overcurrent relay control logic. |
| Property-public | PresentState | PA get\_State and set\_State. |
| Property-public | NormalState | PA get\_NormalState and set\_NormalState. |

### TStorageControllerObj

| *TStorageControllerObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj, TStorageObj and TExpControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset 12. Set\_PendingChange 13. Get\_PendingChange 14. GetCurrents 15. GetInjCurrents 16. CalcYearlyMult 17. CalcDailyMult 18. CalcDutyMult | | |
| Method-private | SetAllFleetValues | Sets the dispatch values for all the storage devices in the fleet. |
| Method-private | SetFleetkWRate | Sets the kW values for all the storage devices in the fleet. |
| Method-private | SetFleetChargeRate | Sets the charging rate for all the storage devices in the fleet. |
| Method-private | SetFleetToCharge | Commands the fleet to enter in charging mode. |
| Method-private | SetFleetToDisCharge | Commands the fleet to enter in discharging mode. |
| Method-private | SetFleetToIdle | Commands the fleet to enter in Idle mode. |
| Method-private | SetFleetToExternal | Commands the fleet to enter in external mode. |
| Method-private | SetFleetDesiredState | Commands the fleet to enter in desired mode. |
| Property-private | InterpretMode | Interprets the mode embedded in the given string. |
| Property-private | GetModeString | Returns the present mode for the fleet. |
| Property-private | GetkWTotal | Returns the total kW. |
| Property-private | GetkWhTotal | Returns the total kWh. |
| Property-private | GetkWhActual | Returns the present kWh. |
| Property-private | GetkWActual | Returns the present kW. |
| Property-private | GetkWActual | Returns the present kW. |
| Property-private | ReturnSeasonTarget | Returns the seasonal target. |
| Property-private | ReturnElementsList | Returns the list of controlled elements as string. |
| Property-private | ReturnWeightsList | Returns the weigh list as string. |
| Property-private | MakeFleetList | Allocates the list of storage devices controlled by this control. |
| Method-private | DoLoadFollowMode | Implements the control routine for follow control mode. |
| Method-private | DoLoadShapeMode | Implements the control routine for load shape control mode. |
| Method-private | DoTimeMode | Implements the control routine for time control mode. |
| Method-private | DoScheduleMode | Implements the control routine for schedule control mode. |
| Method-private | DoPeakShaveModeLow | Implements the control routine for peak shave control mode. |
| Method-private | PushTimeOntoControlQueue | Push present time onto control queue to force re solve at new dispatch value. |
| Property-private | NormalizeToTOD | Normalize time to a floating-point number representing time of day If Hour > 24 time should be 0 to 23.999999. |
| Property-private | GetControlPower | Returns the total power of the fleet. |
| Property-private | GetControlCurrent | Returns the control current. |
| Property-private | Get\_FleetkW | Returns the fleet kW. |
| Property-private | Get\_FleetkWh | Returns the fleet kWh. |
| Property-private | Get\_FleetkWhRating | Returns the fleet kWh rating. |
| Property-private | Get\_FleetReservekWh | Returns the fleet kWh reserve. |
| Property-private | Get\_DynamicTarget | Returns the target considering the seasonal ratings. |
| Property-public | FleetkW | PA Get\_FleetkW. |
| Property-public | FleetkWh | PA Get\_FleetkWh. |
| Property-public | FleetkWhRating | PA Get\_FleetkWhRating. |
| Property-public | FleetReservekWh | PA Get\_FleetReservekWh. |

### TSwtControlObj

| *TSwtControlObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj, TRelayObj and TExpControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset 12. Set\_PendingChange 13. Get\_PendingChange 14. GetCurrents 15. GetInjCurrents 16. Set\_NormalState | | |
| Method-private | InterpretSwitchAction | Interprets the switch action (open/closed/trip). |
| Method-private | set\_Flocked | Locks the switch control. |
| Method-private | Set\_LastAction | Sets the last pending action. |
| Method-private | Set\_PresentState | Changes the present state of the switch control. |
| Property-public | NormalState | PA FNormalState (variable) and Set\_NormalState. |
| Property-public | PresentState | PA FPresentState (variable) and Set\_PresentState. |
| Property-public | IsLocked | PA FLocked (variable). |
| Property-public | Locked | PA Flocked (variable) and set\_Flocked. |
| Property-public | CurrentAction | PA ActionCommand (variable) and Set\_LastAction. |

### TUPFCControlObj

|  |  |  |
| --- | --- | --- |
| *TUPFCControlObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj, TRelayObj and TExpControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset 12. Set\_PendingChange 13. Get\_PendingChange 14. GetCurrents 15. GetInjCurrents | | |
| Property-public | MakeUPFCList | Allocates the list of UPFC devices controlled by this controller. |
| Property-public | UPFCList | PA FUPFCList (variable) and FUPFCList (variable). |
| Property-public | UPFCListSize | PA FListSize (variable) and FListSize (variable). |

### TFuseObj

|  |  |  |
| --- | --- | --- |
| *TSwtControlObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCapControlObj, TRelayObj and TExpControlObj:   1. RecalcElementData 2. CalcYPrim 3. GetLosses 4. GetPropertyValue 5. InitPropertyValues 6. DumpProperties 7. SaveWrite 8. MakePosSequence 9. Sample 10. DoPendingAction 11. Reset 12. Set\_PendingChange 13. Get\_PendingChange 14. GetCurrents 15. GetInjCurrents 16. get\_States 17. set\_States 18. get\_NormalStates 19. Set\_NormalState | | |
| Property-public | States | PA get\_States and set\_States. |
| Property-public | NormalStates | PA get\_NormalStates (variable) and set\_NormalStates. |

## General-purpose objects (libraries)

General-purpose elements are objects modeling default values for PD and PC elements. These objects can be used as libraries for reducing the length of the declarations for PD and PC elements. The inheritance tree for the existing general-purpose elements is shown in Figure 8 (classes) and Figure 9 (objects).

Chart

Description automatically generated

Figure . general-purpose classes inheritance tree

### TConductorData

|  |  |  |
| --- | --- | --- |
| *TConductorData* | | |
| *Type-access* | *Command* | *Description* |
| Method-protected | CountProperties | Add number of intrinsic properties to the number of properties of the caller object. |
| Method-protected | DefineProperties | See 3.3.1. |
| Property-protected | ClassEdit | See 3.3.1. |
| Method-protected | ClassMakeLike | See 3.3.1. |

### TCableData

|  |  |  |
| --- | --- | --- |
| *TCableData* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TConductorData:   1. CountProperties 2. DefineProperties 3. ClassEdit 4. ClassMakeLike | | |

### TCNData

|  |  |  |
| --- | --- | --- |
| *TCNData* | | |
| *Type-access* | *Command* | *Description* |
| Property-private | Get\_Code | Returns active line code string. |
| Method-private | Set\_Code | Sets the active CNData. |
| Method-protected | DefineProperties | See 3.3.2. |
| Property-protected | MakeLike | See 3.3.2. |
| Property-public | Edit | See 3.3.2. |
| Property-public | Init | See 3.3.2. |
| Property-public | NewObject | See 3.3.2. |
| Property-public | Code | PA Get\_Code and Set\_Code. |

### TTSData

|  |  |  |
| --- | --- | --- |
| *TTSData* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCNData:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | |

### TWireData

|  |  |  |
| --- | --- | --- |
| *TWireData* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TCNData:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | |

### TGrowthShape

| *TGrowthShape* | | | | |
| --- | --- | --- | --- | --- |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TCNData:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | | | |
| Method-private | DoCSVFile | | | Exports the growth shape in a CSV file. |
| Method-private | DoSngFile | | | Exports the growth shape in a SNG file. |
| Method-private | DoDblFile | | | Exports the growth shape in a DBL file. |

### TLoadShape

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *TLoadShape* | | | | |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TCNData:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | | | |
| Method-private | DoCSVFile | | | Exports the shape in a CSV file. |
| Method-private | DoSngFile | | | Exports the shape in a SNG file. |
| Method-private | DoDblFile | | | Exports the shape in a DBL file. |
| Method-private | Do2ColCSVFile | | | Exports the shape in a CSV file. In this case, exports 2 columns representing 2 different load shapes (normally P and Q). |
| Method-public | Find | | | Find an obj of this class by name. |
| Method-public | TOPExport | | | Implements the routine for generating a file compatible with TOP. |
| Method-public | CreateMMF | | | Creates the Memory mapping for the file specified, Destination is used to indicate the destination (0 = P, 1 = Q). |

### TPriceShape

| *TPriceShape* | | | | |
| --- | --- | --- | --- | --- |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TCNData:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | | | |
| Method-private | DoCSVFile | | | Exports the shape in a CSV file. |
| Method-private | DoSngFile | | | Exports the shape in a SNG file. |
| Method-private | DoDblFile | | | Exports the shape in a DBL file. |
| Method-public | Find | | | Find an obj of this class by name. |
| Method-public | TOPExport | | | Implements the routine for generating a file compatible with TOP. |

### TTShape

|  |  |  |
| --- | --- | --- |
| *TTShape* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TPriceShape:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code 9. DoCSVFile 10. DoSngFile 11. DoDblFile 12. Find 13. TOPExport | | |

### TXYcurve

| *TXYcurve* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TPriceShape:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code 9. DoCSVFile 10. DoSngFile 11. DoDblFile 12. Find | | |

### TTCC\_Curve

|  |  |  |
| --- | --- | --- |
| *TTCC\_Curve* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TPriceShape:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | |

### TXfmrCode

|  |  |  |  |
| --- | --- | --- | --- |
| *TXfmrCode* | | | |
| *Type-access* | *Command* | *Description* | |
| Implements the following properties/methods as in TPriceShape:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | | |
| Method-private | SetActiveWinding | | Sets the variable indicating the active winding. |
| Method-private | InterpretWindings | | Interprets the given string to assign a value to the property specified. |

### TSpectrum

|  |  |  |
| --- | --- | --- |
| *TSpectrum* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TPriceShape:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code 9. DoCSVFile | | |

### TLineSpacing

|  |  |  |  |
| --- | --- | --- | --- |
| *TLineSpacing* | | | |
| *Type-access* | | *Command* | *Description* |
| Implements the following properties/methods as in TPriceShape:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | | |
| Method-private | InterpretArray | | Interprets array given as string for assigning the spacing specified in the argument. |

### TLineGeometry

|  |  |  |
| --- | --- | --- |
| *TLineGeometry* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TPriceShape:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | |

### TLineCode

|  |  |  |  |
| --- | --- | --- | --- |
| *TLineCode* | | | |
| *Type-access* | | *Command* | *Description* |
| Implements the following properties/methods as in TPriceShape:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | | |
| Method-private | SetZ1Z0 | | Sets symmetrical component impedances and a flag to indicate they were changed. |
| Method-private | SetUnits | | Decode the units’ specification. |
| Method-private | DoMatrix | | Sets impedances as matrices. |

### TDynamicExp

|  |  |  |
| --- | --- | --- |
| *TDynamicExp* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TPriceShape:   1. Get\_Code 2. Set\_Code 3. DefineProperties 4. MakeLike 5. Edit 6. Init 7. NewObject 8. Code | | |

Diagram

Description automatically generated

Figure . General-purpose objects inheritance tree

### TConductorDataObj

| *TConductorDataObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Property-public | Rdc | PA FRDC (variable). |
| Property-public | Rac | PA FR60 (variable). |
| Property-public | GMR | PA FGMR60 (variable). |
| Property-public | CapRadius | PA Fcapradius60 (variable). |
| Property-public | Radius | PA FRadius (variable). |
| Property-public | ResUnits | PA FresistanceUnits (variable). |
| Property-public | RadiusUnits | PA FradiusUnits (variable). |
| Property-public | GMRUnits | PA FGMRUnits (variable). |
| Method-public | InitPropertyValues | Overrides the general class declaration. Initializes the property values for this element. |
| Method-public | DumpProperties | See 3.2.10. |
| Property-public | GetPropertyValue | See 3.2.10. |
| Property-public | GetNumProperties | See 3.2.10. |

### TCableDataObj

|  |  |  |
| --- | --- | --- |
| *TCableDataObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TConductorDataObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue 4. GetNumProperties | | |
| Property-public | EpsR | PA FEpsR (variable). |
| Property-public | DiaIns | PA FDiaIns (variable). |
| Property-public | DiaCable | PA FDiaCable (variable). |
| Property-public | InsLayer | PA FInsLayer (variable). |

### TCNDataObj

|  |  |  |
| --- | --- | --- |
| *TCNDataObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TConductorDataObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue 4. GetNumProperties | | |
| Property-public | NStrand | PA FkStrand (variable). |
| Property-public | DiaStrand | PA FDiaStrand (variable). |
| Property-public | GmrStrand | PA FGmrStrand (variable). |
| Property-public | RStrand | PA FRStrand (variable). |

### TTSDataObj

|  |  |  |
| --- | --- | --- |
| *TTSDataObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TConductorDataObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue 4. GetNumProperties | | |
| Property-public | DiaShield | PA FDiaShield (variable). |
| Property-public | TapeLayer | PA FTapeLayer (variable). |
| Property-public | TapeLap | PA FTapeLap (variable). |

### TWireDataObj

|  |  |  |
| --- | --- | --- |
| *TWireDataObj* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TConductorDataObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue 4. GetNumProperties | | |

### TGrowthShapeObj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *TGrowthShapeObj* | | | | |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TConductorDataObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue 4. GetNumProperties | | | | |
| Method-private | ReCalcYearMult | | | Fill up the YearMult array with total yearly multiplier from base year. |
| Property-public | GetMult | | | Get multiplier for Specified Year. |

### TLoadShapeObj

| *TLoadShapeObj* | | | | |
| --- | --- | --- | --- | --- |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TConductorDataObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue | | | | |
| Property-private | Get\_Interval | | | Returns the present time interval. |
| Method-private | Set\_NumPoints | | | Updates the number of points for the shape. |
| Method-private | SaveToDblFile | | | Saves the shape into a DBL file. |
| Method-private | SaveToSngFile | | | Saves the shape into a SNG file. |
| Method-private | CalcMeanandStdDev | | | Calculates the mean and standard deviation for the shape. |
| Method-private | Get\_Mean | | | Returns the latest mean calculated. |
| Method-private | Get\_StdDev | | | Returns the latest standard deviation calculated. |
| Method-private | Set\_Mean | | | Updates the mean using the given value. |
| Method-private | Set\_StdDev | | | Normalize the curve presently in memory. |
| Method-private | SetMaxPandQ | | | Obtains the maximum P and Q from the curves already loaded in memory. |
| Property-public | GetMult | | | Get multiplier at specified time. |
| Property-public | Mult | | | Get multiplier by index. |
| Property-public | Hour | | | Get hour corresponding to point index. |
| Method-public | Normalize | | | Normalizes the waveform in memory. |
| Method-public | LoadMMFView | | | Loads the current view of the MMF into memory for further use. |
| Method-public | LoadFileFeatures | | | Loads the mapped file features into local variables for further use. |
| Property-public | NumPoints | | | PA FNumPoints (variable) and Set\_NumPoints. |
| Property-public | PresentInterval | | | PA Get\_Interval. |
| Property-public | Mean | | | PA Get\_Mean and Set\_Mean. |
| Property-public | StdDev | | | PA Get\_StdDev and Set\_StdDev. |

### TPriceShapeObj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *TPriceShapeObj* | | | | |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TLoadShapeObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue 4. Get\_Interval 5. Set\_NumPoints 6. SaveToDblFile 7. SaveToSngFile 8. CalcMeanandStdDev 9. Get\_Mean 10. Get\_StdDev 11. Set\_Mean 12. Set\_StdDev 13. NumPoints 14. PresentInterval 15. Mean 16. StdDev | | | | |
| Property-public | GetPrice | | | Get Prices at specified time, hr. |
| Property-public | Price | | | Get Prices by index. |
| Property-public | Hour | | | Get hour corresponding to point index. |

### TTShapeObj

| *TTShapeObj* | | | | |
| --- | --- | --- | --- | --- |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TLoadShapeObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue 4. Get\_Interval 5. Set\_NumPoints 6. SaveToDblFile 7. SaveToSngFile 8. CalcMeanandStdDev 9. Get\_Mean 10. Get\_StdDev 11. Set\_Mean 12. Set\_StdDev 13. NumPoints 14. PresentInterval 15. Mean 16. StdDev | | | | |
| Property-public | GetTemperature | | | Get Temperatures at specified time, hr. |
| Property-public | Temperature | | | Get Temperatures by index. |
| Property-public | Hour | | | Get hour corresponding to point index. |

### TXYcurveObj

| *TXYcurveObj* | | | | |
| --- | --- | --- | --- | --- |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TLoadShapeObj and TLineObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue 4. Set\_NumPoints 5. SaveWrite 6. NumPoints | | | | |
| Property-private | InterpolatePoints | | | Returns a point after interpolating the given coordinates. |
| Property-private | Get\_YValue | | | Get Y Value by index. |
| Property-private | Get\_XValue | | | Get X Value corresponding to point index. |
| Method-private | Set\_XValue | | | Sets X Value corresponding to point index. |
| Method-private | Set\_YValue | | | Sets Y Value corresponding to point index. |
| Property-private | Get\_X | | | Get X present value. |
| Property-private | Get\_Y | | | Get Y present value. |
| Method-private | Set\_X | | | Sets present X value. |
| Method-private | Set\_Y | | | Sets present Y value. |
| Property-public | GetYValue | | | Get Y value at specified X Value. |
| Property-public | GetXValue | | | Get X value at specified Y Value. |
| Property-public | GetCoefficients | | | This function returns the coefficients of the line interpolated line for the given X (a\*X + b). If no points exist in the curve (or just a single point), the result is (a = 0, b = 0). If Xvalue is outside the range of defined X values, the curve is extrapolated from the Ends (a = 0, b = extrapolated value). |
| Property-public | XValue\_pt | | | PA Get\_XValue and Set\_XValue. |
| Property-public | YValue\_pt | | | PA Get\_YValue and Set\_YValue. |
| Property-public | X | | | PA Get\_X and Set\_X. |
| Property-public | Y | | | PA Get\_Y and Set\_Y. |

### TTCC\_CurveObj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *TTCC\_CurveObj* | | | | |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TLoadShapeObj and TLineObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue 4. NumPoints | | | | |
| Property-public | GetTCCTime | | | Return operating time for a particular time value. |
| Property-public | GetUVTime | | | Return operating time for undervoltage relay. |
| Property-public | GetOVTime | | | Return operating time for overvoltage relay. |
| Property-public | Value | | | Get C\_Value by index. |
| Property-public | Time | | | Get time value (sec) corresponding to point index. |

### TXfmrCodeObj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *TXfmrCodeObj* | | | | |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TLoadShapeObj and TLineObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue | | | | |
| Property-public | SetNumWindings | | | Reallocates memory for hosting the given number of windings. |
| Property-public | PullFromTransformer | | | Imports the features of the given transformer. |

### TSpectrumObj

| *TSpectrumObj* | | | | |
| --- | --- | --- | --- | --- |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TLoadShapeObj and TLineObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue | | | | |
| Property-public | GetMult | | | Search List for harmonic (nearest 0.01 harmonic) and returns multiplier. |

### TLineSpacingObj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *TLineSpacingObj* | | | | |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TLoadShapeObj and TLineObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue | | | | |
| Method-private | set\_Nwires | | | Reallocates memory for hosting the given number of wires. |
| Property-private | Get\_FX | | | Gets X for the given index. |
| Property-private | Get\_FY | | | Gets Y for the given index. |
| Property-public | Xcoord | | | PA Get\_FX. |
| Property-public | Ycoord | | | PA Get\_FY. |
| Property-public | NWires | | | PA FNConds (variable) and set\_Nwires. |
| Property-public | NPhases | | | PA FNPhases (variable). |
| Property-public | Units | | | PA FUnits (variable). |

### TLineGeometryObj

| *TLineGeometryObj* | | | | |
| --- | --- | --- | --- | --- |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TLoadShapeObj and TLineObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue 4. SaveWrite | | | | |
| Method-private | ChangeLineConstantsType | | | Reallocates memory for hosting the given number of phases. |
| Method-private | set\_Nconds | | | Sets the number of conductors for this line geometry. |
| Method-private | set\_Nphases | | | Sets the number of phases for this line geometry. |
| Method-private | set\_ActiveCond | | | Sets the given conductor number to be the active conductor. |
| Method-private | Get\_YCmatrix | | | Returns the Y matrix for this element. |
| Method-private | Get\_Zmatrix | | | Returns the Z matrix for this element. |
| Method-private | Get\_RhoEarth | | | Gets the rho earth value for the line model. |
| Method-private | Set\_RhoEarth | | | Sets the rho earth value for the line model. |
| Property-private | get\_Nconds | | | Returns the number of conductors for this line model. |
| Method-private | UpdateLineGeometryData | | | Call this before using the line data. Implements the routine for importing the wire/cable data values linked to this model. |
| Property-private | Get\_FX | | | Gets X for the given index. |
| Property-private | Get\_FY | | | Gets Y for the given index. |
| Property-private | Get\_Funits | | | Returns the units for the active conductor. |
| Property-private | Get\_ConductorName | | | Returns the name for the active conductor. |
| Property-private | Get\_ConductorData | | | Returns the conductor data object for the active conductor. |
| Property-private | Get\_PhaseChoice | | | Returns the phase choice for the active conductor. |
| Property-public | Xcoord | | | PA Get\_FX. |
| Property-public | Ycoord | | | PA Get\_FY. |
| Property-public | Nwires | | | PA FNConds (variable) and set\_Nwires. |
| Property-public | Nphases | | | PA FNPhases (variable) and set\_Nphases. |
| Property-public | Units | | | PA Get\_FUnits . |
| Method-public | LoadSpacingAndWires | | | Called from a Line object that has its own Spacing and Wires input automatically sets reduce=y if the spacing has more wires than phases. |
| Property-public | Nconds | | | PA get\_Nconds and set\_Nconds. |
| Property-public | ActiveCond | | | PA FActiveCond (variable) and set\_ActiveCond. |
| Property-public | Zmatrix | | | PA Get\_Zmatrix. |
| Property-public | YCmatrix | | | PA Get\_YCmatrix. |
| Property-public | RhoEarth | | | PA Get\_RhoEarth and Set\_RhoEarth. |
| Property-public | ConductorName | | | PA Get\_ConductorName. |
| Property-public | ConductorData | | | PA Get\_ConductorData. |
| Property-public | NWires | | | PA FNConds (variable). |
| Property-public | PhaseChoice | | | PA Get\_PhaseChoice. |

### TLineCodeObj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *TLineCodeObj* | | | | |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TLoadShapeObj and TLineObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue | | | | |
| Method-private | Set\_NPhases | | | Sets the number of phases for this line model. |
| Method-private | DoKronReduction | | | Applies the Kron reduction to the model. |
| Property-private | get\_Rmatrix | | | Returns the R matrix as a string. |
| Property-private | get\_Xmatrix | | | Returns the X matrix as a string. |
| Property-private | get\_CMatrix | | | Returns the C matrix as a string. |
| Property-public | NumPhases | | | PA FNPhases (variable) and Set\_Nphases. |
| Property-public | CalcMatricesFromZ1Z0 | | | Calculates the sequence impedances matrices for this model. |

### TDynamicExpObj

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *TDynamicExpObj* | | | | |
| *Type-access* | | *Command* | *Description* | |
| Implements the following properties/methods as in TLoadShapeObj and TLineObj:   1. InitPropertyValues 2. DumpProperties 3. GetPropertyValue | | | | |
| Property-private | InterpretDiffEq | | | Builds the list of commands required for operating the equation declared, this  automation is intended to accelerate the calculation in run time. |
| Property-private | Get\_Closer\_Op | | | Returns the index of the closest operator found in the given string starting from left to right. |
| Property-private | Get\_Var\_Idx | | | Returns the index of the variable if it exists in the state variable list, otherwise, it returns 50001 if the string entered is a numeric constant (DBL) or -1 if the string entered is neither a numeric constant nor state variable. |
| Property-private | Check\_If\_CalcValue | | | Checks if the given string is a value calculated by the element using the eq model. |
| Property-private | Get\_Out\_Idx | | | Gets the index for the given variable if it is an output. |
| Property-private | SolveEq | | | Implements the solver for the local equation. |
| Property-private | IsInitVal | | | Returns true if the given code is for an initialization value. |
| Property-private | Get\_DynamicEqVal | | | Returns the value of the state variable using the given indexes. |
| Property-private | Get\_VarName | | | Returns the name of the state variable corresponding to the given index. |
| Property-private | NumVars | | | PA FNumVars (variable). |

## Meters

Meters are objects crated for recording values across the circuit model during the simulation. These values can be exported at the end of the simulation or can be used by controls to collect data that will be used for performing control actions. Some meters have additional functionalities that affect not only the monitored element but the whole circuit. The inheritance tree for the existing general-purpose elements is shown in Figure 8 (classes) and Figure 9 (objects).

A picture containing chart

Description automatically generated

Figure . Meter classe inheritance tree.

### TMeterClass

|  |  |  |
| --- | --- | --- |
| *TMeterClass* | | |
| *Type-access* | *Command* | *Description* |
| Method-protected | CountProperties | Add number of intrinsic properties to the number of properties of the caller object. |
| Method-protected | DefineProperties | See 3.3.1. |
| Property-protected | ClassEdit | See 3.3.1. |
| Method-protected | ClassMakeLike | See 3.3.1. |
| Method-public | ResetAll | Defines the space in the object for clearing all the meters in the specific class. It is expected to be overridden by a more specific class. |
| Method-public | SampleAll | Defines the space in the object for forcing all meters in the specific class to take a sample. It is expected to be overridden by a more specific class. |
| Method-public | SaveAll | Defines the space in the object for forcing all meters in the specific class to save their buffers to disk. It is expected to be overridden by a more specific class. |

### TEnergyMeter

| *TEnergyMeter* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TMeterClass and 3.1.4:   1. DefineProperties 2. MakeLike 3. Edit 4. Init 5. NewObject | | |
| Method-private | ProcessOptions | Parses the options for the energy meter. |
| Method-private | Set\_SaveDemandInterval | Sets the demand interval flag using the given value. |
| Method-private | Get\_SaveDemandInterval | Gets the demand interval flag. |
| Method-private | CreateMeterTotals | Allocates in memory for the meter totals registers. |
| Method-private | CreateFDI\_Totals | Allocates in memory for the FDI totals registers. |
| Method-private | ClearDI\_Totals | Clears the DI totals registers. |
| Method-private | WriteTotalsFile | Sum up all registers of all meters and write to Totals.CSV. |
| Method-private | OpenOverloadReportFile | Allocates memory for the overload report, initializes the memory space with headers. |
| Method-private | OpenVoltageReportFile | Allocates memory for the voltage violations report, initializes the memory space with headers. |
| Method-private | WriteOverloadReport | Scans the active circuit for overloaded PD elements and writes each to a file. This is called only if in Demand Interval (DI) mode and the file is open. |
| Method-private | WriteVoltageReport | For any bus with a defined voltage base, test for > Vmax or < Vmin and writes each to a file. This is called only if in Demand Interval (DI) mode and the file is open. |
| Method-private | InterpretRegisterMaskArray | Interprets the registry mask array using the given reference. |
| Method-private | Set\_DI\_Verbose | Sets the DI verbose flag. |
| Method-private | Get\_DI\_Verbose | Gets the DI verbose flag. |
| Method-protected | SetHasMeterFlag | Set the HasMeter Flag for all cktElement. |
| Method-public | ResetMeterZonesAll | Force all EnergyMeters in the circuit to reset their meter zones. |
| Method-public | ResetAll | Reset all meters in active circuit to zero. Overrides the more generic class. |
| Method-public | SampleAll | Forces all meters in active circuit to sample. Overrides the more generic class. |
| Method-public | SaveAll | Forces all meters in the circuit to take a sample. Overrides the more generic class. |
| Method-public | AppendAllDIFiles | Appends the DI files for all the energy meters. |
| Method-public | OpenAllDIFiles | Opens the DI files for all the energy meters. |
| Method-public | CloseAllDIFiles | Closes the DI files for all the energy meters. |
| Property-public | SaveDemandInterval | PA Get\_SaveDemandInterval and Set\_SaveDemandInterval. |
| Property-public | DI\_Verbose | PA Get\_DI\_Verbose and Set\_DI\_Verbose. |

### TDSSFMonitor

|  |  |  |
| --- | --- | --- |
| *TDSSFMonitor* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TMeterClass and 3.1.4:   1. DefineProperties 2. MakeLike 3. Edit 4. Init 5. NewObject | | |
| Method-public | ResetAll | Reset all meters in active circuit to zero. Overrides the more generic class. |
| Method-public | SampleAll | Forces all meters in active circuit to sample. Overrides the more generic class. |
| Method-public | SaveAll | Forces all meters in the circuit to take a sample. Overrides the more generic class. |
| Method-public | update\_sys\_ld\_info | Updates the latest simulation data for all Fmonitors. |
| Method-public | Calc\_P\_freq | Calculates frequency for each cluster. |
| Method-public | update\_atks | Updates the attack variables for all Fmonitors. |
| Method-public | update\_defense\_layer | Updates the defense layer for all Fmonitors. |

### TDSSMonitor

|  |  |  |
| --- | --- | --- |
| *TDSSMonitor* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TMeterClass, TEnergyMeter and 3.1.4:   1. DefineProperties 2. MakeLike 3. Edit 4. Init 5. NewObject 6. ResetAll 7. SampleAll 8. SaveAll | | |
| Method-public | SampleAllMode5 | Sample just Mode 5 monitors. |
| Method-public | PostProcessAll | Calculates frequency for each cluster. |
| Method-public | update\_atks | Forces all enabled monitors to postprocess recorded data. |
| Method-public | TOPExport | See 3.4.9. |

### TSensor

|  |  |  |
| --- | --- | --- |
| *TSensor* | | |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in TEnergyMeter:   1. DefineProperties 2. MakeLike 3. Edit 4. Init 5. NewObject 6. ResetAll 7. SampleAll 8. SaveAll | | |
| Method-public | SetHasSensorFlag | Sets the HasSensorObj Flag for all cktElement. |

Chart, box and whisker chart

Description automatically generated

Figure . Meter object inheritance tree.

### TMeterElement

|  |  |  |
| --- | --- | --- |
| *TMeterElement* | | |
| *Type-access* | *Command* | *Description* |
| Method-protected | TakeSample | Sample control quantities and set action times in Control Queue. |
| Method-protected | AllocateSensorArrays | Allocate memory for storing recorded data. |
| Method-protected | CalcAllocationFactors | Calculates the allocation factors. The Phase Allocation Factor is the amount that the load must change to match the measured peak. |

### TEnergyMeterObj

| *TEnergyMeterObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in 3.3.15:   1. MakePosSequence 2. RecalcElementData 3. CalcYPrim 4. GetCurrents 5. GetInjCurrents 6. GetPropertyValue 7. InitPropertyValues 8. DumpProperties | | |
| Method-private | Integrate | Performs the integration of the demand interval registers. |
| Method-private | SetDragHandRegister | Stores data in the shift registers for DI. |
| Property-private | Accumulate\_Load | Accumulate load in meter’s zone. |
| Method-private | Accumulate\_Gen | Accumulate generation in meter’s zone. |
| Method-private | CalcBusCoordinates | Calculates bus coordinates using interpolation. |
| Property-private | AddToVoltBaseList | Add to VoltBase list if not already there and return index. |
| Property-private | MakeDIFileName | Creates and returns the name for the DI report file. |
| Property-private | MakeVPhaseReportFileName | Creates and returns the name for the phase voltage report file. |
| Method-private | AssignVoltBaseRegisterNames | Assigns voltage base to register names (DI) |
| Method-private | TotalupDownstreamCustomers | Totalizes the number of customers in the zone. |
| Method-protected | OpenDemandIntervalFile | Opends DI file. |
| Method-protected | WriteDemandIntervalData | Writes in DI file. |
| Method-protected | CloseDemandIntervalFile | Closes in DI file. |
| Method-protected | AppendDemandIntervalFile | Appends to DI file. |
| Method-public | ResetRegisters | Resets the meter registers. |
| Method-public | TakeSample | Update registers from metered zone. Assumes one time has taken place since last sample. Overrides the more generic class. |
| Method-public | SaveRegisters | Saves the meter register in the meter file. |
| Method-public | MakeMeterZoneLists | This gets fired off whenever the bus lists are rebuilt. Must be updated whenever there is a change in the circuit. |
| Method-public | ZoneDump | Dumps the elements within the zone into the meter file assigned. |
| Method-public | InterpolateCoordinates | Completes the missing bus coordinates by interpolation. |
| Method-public | EnableFeeder | HasFeeder must be true before feederObj will be re-enabled. Not implemented. |
| Method-public | AllocateLoad | Allocates load across the feeder model using allocation factors. |
| Method-public | ReduceZone | Reduce Zone by eliminating buses and merging lines. |
| Method-public | SaveZone | Run down the zone and write each element into a file. |
| Method-public | GetPCEatZone | Gets all the PCE within the meter zone. |
| Method-public | CalcReliabilityIndices | Calculates the feeder’s reliability indices using backward forward sweep. |

### TFMonitorObj

| *TFMonitorObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in 3.3.15:   1. MakePosSequence 2. RecalcElementData 3. CalcYPrim 4. GetCurrents 5. GetInjCurrents 6. GetPropertyValue 7. InitPropertyValues 8. DumpProperties | | |
| Method-private | Set\_nodes\_for\_fm | Initiate the structure of this FMon. |
| Method-private | Set\_CommVector | Loop for no more than the expected number of windings; Ignore omitted values. |
| Method-private | Set\_CommVector\_Hide | Loop for no more than the expected number of windings; Ignore omitted values. |
| Method-private | Set\_CommVector\_NodeHide | Loop for no more than the expected number of windings; Ignore omitted values. |
| Method-private | Set\_volt\_lmt\_clstr | The first entry is the No. of iNode. |
| Method-private | Set\_CommDelayVector | Loop for no more than the expected number of windings; Ignore omitted values. |
| Method-private | ResetDelaySteps | Calculates the array for defining how many delays for communication. |
| Method-private | update\_attack | Update d\_i. |
| Method-private | update\_defense | Update z\_i. |
| Property-private | organise\_dfs\_node | Update z\_i. |
| Method-private | Set\_atk\_dfs | Calculate K\_i z. |
| Method-private | Set\_EquivalentGenerator | Sets the equivalent generator. |
| Method-private | Set\_ElemTable\_line | Terminal number load into data str. |
| Method-private | Init\_nodeFM | Init all info of this node. |
| Method-private | Get\_PDElem\_terminal\_voltage | Procedure push\_voltage |
| Method-private | Calc\_Alpha\_for\_PDNode | Must be called after self-gradient calc. Self-gradient calc is in 'Update \_pd\_node\_info’. |
| Method-private | update\_all\_nodes\_info | Updates info about all nodes. |
| Property-private | AvgPmax | Returns the average P max. |
| Property-private | AvgQmax | Returns the average Q max. |
| Method-private | Get\_PQ\_DI | Calculates the P and Q for DI. |
| Property-private | Calc\_Grdt\_for\_Alpha | Calculate the gradient for alpha i. |
| Property-private | Calc\_Grdt\_for\_Alpha\_vivj | Calculate the gradient for alpha i. |
| Property-private | Getgradient | Returns the gradient for positive sequence. |
| Property-private | Calc\_GP\_AlphaP | NodeNuminClstr: node number in cluster. |
| Property-private | Get\_power\_trans | AlphaP Gradient or Pref. |
| Property-private | Coef\_Phi | Returns a coefficient. |
| Method-public | ResetIt | Resets the monitor. |
| Method-public | Save | Saves present buffer to file. |
| Property-public | Calc\_sum\_dij\_Alphaj | Calculates alpha. |
| Property-public | Calc\_Alpha\_M2 | Calculates alpha. Only work for Generic5 nodefm. |
| Property-public | Calc\_Alpha\_L | Calculates alpha losses. |
| Property-public | Calc\_Alpha\_L\_vivj | Calculates alpha losses. |
| Property-public | Calc\_Alpha\_LnM2 | Calculates alpha losses. Only work for Generic5 nodefm. |
| Property-public | Calc\_AlphaP | Calculates alpha P sum. |
| Property-public | Get\_P\_mode | Returns P mode. |
| Property-public | Calc\_fm\_ul\_0 | Update voltages on all buses. |
| Property-public | Calc\_fm\_us\_0 | Update voltages on all buses. |
| Method-public | Agnt\_smpl | Sample data of this node at each t\_intvl\_smpl. |
| Method-public | Init\_delay\_array | Initializes the delay array. |
| Property-public | Calc\_ul\_P | For real power control-dynamic simulation. |
| Property-public | Calc\_Gradient\_ct\_P | For real power control-dynamic simulation curtailment. |
| Method-public | update\_node\_info\_each\_time\_  step | Updates all nodes in the cluster. |
| Method-public | update\_ld\_dly | Updates all nodes in this cluster with delay. |
| Method-public | Calc\_P\_freq\_fm | Calculates frequency for each cluster. |
| Property-public | Get\_FileName | Returns the file name for storing meter data. |

### *TMonitorObj*

| *TMonitorObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in 3.3.15:   1. MakePosSequence 2. RecalcElementData 3. CalcYPrim 4. GetCurrents 5. GetInjCurrents 6. GetPropertyValue 7. InitPropertyValues 8. DumpProperties | | |
| Method-private | AddDblsToBuffer | Adds floating-point numbers to the values buffer. |
| Method-private | AddDblToBuffer | Adds a single floating-point number to the values buffer. |
| Method-public | ResetIt | Resets the monitor. |
| Method-public | Save | Saves present buffer to file. |
| Method-public | PostProcess | Calculates Pst or other post-processing. |
| Method-public | Add2Header | Adds text to the monitor header. |
| Method-public | OpenMonitorStream | Allocates memory for storing data. |
| Method-public | ClearMonitorStream | Clears the monitor’s byte stream. |
| Method-public | CloseMonitorStream | Destroys the monitor’s byte stream. |
| Method-public | TranslateToCSV | Saves the monitor’s byte stream into a CSV file. |
| Property-public | Get\_FileName | Returns the file name for storing meter data. |

### TSensorObj

| *TSensorObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in 3.3.15:   1. MakePosSequence 2. RecalcElementData 3. CalcYPrim 4. GetCurrents 5. GetInjCurrents 6. GetPropertyValue 7. InitPropertyValues 8. DumpProperties | | |
| Method-private | Set\_Conn | Interpret the Connection. |
| Method-private | Set\_Action | Interpret Action Property. Not implemented. |
| Method-private | ZeroSensorArrays | Resets all the sensor arrays (0). |
| Method-private | AllocateSensorObjArrays | Allocates the sensor arrays in memory. |
| Method-private | RecalcVbase | Recalculates the voltage base for the monitored buses. |
| Method-private | RotatePhases | For Delta connections or Line-Line voltages. |
| Property-private | LimitToPlusMinusOne | Returns +/- 1 accroding to the sign of the given integer. |
| Method-private | ClearSensor | Clears the sensor registers and flags. |
| Method-private | UpdateCurrentVector | Updates the current vector when P and Q are defined as the input vectors for the sensor. |
| Property-private | Get\_WLSCurrentError | Return the WLS Error for Currents. Get Square error and weight it. |
| Property-private | Get\_WLSVoltageError | Return the WLS Error for Voltages. Get Square error and weight it. |
| Method-public | TakeSample | Go add a sample to the buffer. Overrides the more generic class. |
| Method-public | ResetIt | Resets the monitor. |
| Method-public | Save | Saves present buffer to file. |
| Property-public | Conn | PA Fconn (variable) and Set\_Conn. |
| Property-public | Action | PA Set\_Action. |
| Property-public | WLSCurrentError | PA Get\_WLSCurrentError. |
| Property-public | WLSVoltageError | PA Get\_WLSVoltageError. |
| Property-public | BaseKV | PA kvbase (variable). |
| Property-public | DeltaDirection | PA FDeltaDirection (variable). |
| Property-public | SensorP | PA SensorKW (variable). |
| Property-public | SensorQ | PA SensorKVAR (variable). |

## Containers

Containers are objects that can contain other objects such as the circuit model, or the solution algorithms for the simulation. The inheritance tree for the container objects presented in this section are shown in Figure 11Figure 8 (classes) and Figure 12Figure 9 (objects).

Chart

Description automatically generated with medium confidence

Figure . Containers class inheritance tree

### TDSSSolution

| *TDSSSolution* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Method-protected | DefineProperties | See 3.3.2. |
| Property-protected | Edit | See 3.2.2. |
| Method-protected | Init | See 3.2.2. |
| Method-public | NewObject | See 3.1.4. |

The TDSSSolution class implements an additional class derived from TThread for allowing OpenDSS-X to use parallel processing. This is called TSolver, which implements the calls to the solution algorithms on a separate thread. TSolver is implemented as an actor, allowing the TDSSSolution and TDSSSolutionObj to send messages to TSolver commanding actions. The messages received are enqueued, guaranteeing that if several callers send commands to the actor “concurrently”, these will be executed by TSolver.

The messages of TSolver are handled through events. TSolver keeps waiting indefinitely until a new message arrives. Every time a new circuit is created, OpenDSS-X will create an actor (instance) of TSolver, this actor will have affinity to all the processor’s threads and real-time priority (MS Windows). TSolver gets destroyed after a “Clear” (for the active actor) or “ClearAll” (for all actors). OpenDSS-X is executed in a separate thread than all the other TSolver instances, facilitating communication between them and avoiding unnecessary queues. The job progress and status of the actors can be verified by TDSSSolutionObj using the global variables for such purpose (DSSGlobals.dss).

Chart, box and whisker chart

Description automatically generated

Figure . Containers object inheritance tree

### TSolutionObj

| *TSolutionObj* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Implements the following properties/methods as in 3.3.15:   1. InitPropertyValues 2. DumpProperties | | |
| Property-private | Converged | Returns a Boolean flag indicating if the latest simulation job converged. |
| Property-private | OK\_for\_Dynamics | Initializes the environment variables required for dynamics mode. |
| Property-private | OK\_for\_Harmonics | Initializes the environment variables required for harmonics mode. |
| Method-private | DoNewtonSolution | Implements a version of the solution algorithm using Newton-Raphson. |
| Method-private | DoNormalSolution | Implements the standard solution algorithm in OpenDSS-X based on floating-point solver. |
| Method-private | SetGeneratordQdV | Save the generator dispatch level and set on high enough to turn all generators on. |
| Method-private | SumAllCurrents | Sums terminal currents into system Currents Array. |
| Method-private | Set\_Frequency | Forces Rebuild of all Y Primitives and to rebuild of System Y using the given frequency. |
| Method-private | Set\_Mode | Sets the given simulation mode and initializes the environment variables to the default values for the mode. |
| Method-private | Set\_Year | Sets the simulation time variables using the given year. |
| Method-private | Set\_Total\_Time | Stores the given time into the total time variable. |
| Method-public | ZeroAuxCurrents | Sets the aux. currents array to zero. |
| Method-public | SolveZeroLoadSnapShot | Solves a power flow using the Y series matrix only (no shunt elements). |
| Method-public | DoPFLOWsolution | Executes the active power flow solution algorithm. |
| Method-public | Solve | Main Solution dispatch. |
| Method-public | SnapShotInit | Initializes the variables and environment for a snapshot simulation. |
| Property-public | SolveSnap | Solves the model once. |
| Property-public | SolveDirect | Solves for now once, direct solution (non-iterative). |
| Property-public | SolveYDirect | Like SolveDirect; used for initialization. |
| Property-public | SolveCircuit | SolveSnap sans control iteration. |
| Method-public | CheckControls | Snapshot checks with matrix rebuild, it takes place after a solution has been calculated. |
| Method-public | SampleControlDevices | Commands all control devices in the model for the active actor to take a sample. |
| Method-public | DoControlActions | Commands all control devices in the model for the active actor to perform all pending control actions in the control queue. |
| Method-public | Sample\_DoControlActions | Calls SampleControlDevices and DoControlActions routines. |
| Method-public | Check\_Fault\_Status | Checks the status of all the faults within the circuit model. |
| Property-public | SolveAD | Solves one of the A-Diakoptics stages locally. |
| Method-public | SetGeneratorDispRef | Sets the global generator dispatch reference. |
| Method-public | SetVoltageBases | Sets voltage bases using voltage at first node (phase) of a bus. |
| Method-public | SaveVoltages | Saves the present voltages into a txt file. |
| Method-public | UpdateVBus | Updates voltages for each bus from NodeV. |
| Method-public | RestoreNodeVfromVbus | Opposite of updatebus. |
| Property-public | VDiff | Calculates the difference between two node voltages. |
| Method-public | Get\_Yiibus | Updates voltages for each bus from NodeV. |
| Property-public | Get\_Yij | Gets Gij + j Bij. |
| Method-public | WriteConvergenceReport | Writes the convergence report to the hard drive. |
| Method-public | Update\_dblHour | Updates the time in the floating-point time register (DBL) in hours. |
| Method-public | Increment\_time | Increments the solution time according to the simulation time step. |
| Method-public | UpdateLoopTime | Update Loop time is called from end of time step cleanup. Timer is based on beginning of SolveSnap time. |
| Property-public | Mode | PA dynavars.SolutionMode (variable) and Set\_Mode. |
| Property-public | Frequency | PA FFrequency (variable) and Set\_Frequency. |
| Property-public | Year | PA FYear (variable) and Set\_Year. |
| Property-public | Time\_Solve | PA Solve\_Time\_Elapsed. |
| Property-public | Time\_TotalSolve | PA Total\_Solve\_Time\_Elapsed. |
| Property-public | Time\_Step | PA Step\_Time\_Elapsed. |
| Property-public | Total\_Time | PA Total\_Time\_Elapsed and Set\_Total\_Time. |
| Method-public | AddInAuxCurrents | Only AutoAdd Obj uses this. |
| Property-public | SolveSystem | Solves the circuit using KLUsolve. It needs several initializations before being called. |
| Method-public | GetPCInjCurr | Gets injected currents from all enabled PC devices. |
| Method-public | GetSourceInjCurrents | Gets injected currents from all enabled Source devices (VSource, ISource). |
| Method-public | ZeroInjCurr | Clears the injection currents vector (0 + j0) |
| Method-public | Upload2IncMatrix | Uploads the values to the incidence matrix. |
| Method-public | Calc\_Inc\_Matrix | Calculates the incidence matrix for the Circuit. |
| Method-public | Calc\_Inc\_Matrix\_Org | Calculates the incidence matrix hierarchically organized for the Circuit. |
| Property-public | get\_IncMatrix\_Row | Gets the index of the Row connected to the specified Column. |
| Property-public | get\_IncMatrix\_Col | Gets the index of the Column connected to the specified Row. |
| Property-public | CheckLocationIdx | Evaluates the area covered by the tearing point to see if there is a better one. |
| Property-public | get\_PDE\_Bus1\_Location | Gets the index of myPDE -> bus1 within the Inc matrix. |
| Method-public | AddLines2IncMatrix | Adds the Lines to the Incidence matrix arrays. |
| Method-public | AddXfmr2IncMatrix | Adds the Xfmrs to the Incidence matrix arrays. |
| Method-public | AddSeriesCap2IncMatrix | Adds capacitors in series to the Incidence matrix arrays. |
| Method-public | AddSeriesReac2IncMatrix | Adds Reactors in series to the Incidence matrix arrays. |
| Method-public | SendCmd2Actors | Sends a message to other actors different than 1. |
| Method-public | UploadV2Master | Uploads the local solution into the master's (actor 1) voltage array. |
| Method-public | UpdateISrc | Updates the local ISources using the data available at Ic for actor 1. |
| Property-public | VoltInActor1 | Returns the voltage indicated in NodeIdx in the context of the actor 1. |

### TDSSCircuit

| *TDSSCircuit* | | |
| --- | --- | --- |
| *Type-access* | *Command* | *Description* |
| Method-private | AddDeviceHandle | Adds a handle Index into CktElements. |
| Method-private | AddABus | Reallocates the bus array for adding a new element. |
| Method-private | AddANodeBus | Reallocates the MapNodeToBus array for adding a new element. |
| Property-private | AddBus | Adds a new bus to the bus list, reallocates memory if needed. |
| Method-private | Set\_ActiveCktElement | Sets the given ckt element active in the actor context. |
| Method-private | Set\_BusNameRedefined | Force Rebuilding of SystemY if bus def has changed so controls will know buses redefined. |
| Property-private | Get\_Losses | Gets the total Circuit losses. |
| Property-private | Set\_LoadMultiplier | Sets the load multiplier in the actor context. |
| Property-private | SaveBusInfo | Save existing bus definitions and names for info that needs to be restored. |
| Property-private | RestoreBusInfo | Restore kV bases, other values to buses still in the list. |
| Property-private | SaveMasterFile | Write Redirect for all populated DSS Classes Except Solution Class. |
| Property-private | SaveDSSObjects | Write Files for all populated DSS Classes Except Solution Class. |
| Property-private | SaveFeeders | Write out all energy meter zones to separate subdirectories. |
| Property-private | SaveBusCoords | Writes the circuit’s buscoords into the file assigned. |
| Property-private | SaveGISCoords | Writes the circuit’s GIS coordinates into the file assigned. |
| Property-private | SaveVoltageBases | Writes the circuit’s bus voltage bases into the file assigned. |
| Method-private | ReallocDeviceList | Reallocate the device list to improve the performance of searches. |
| Method-private | Set\_CaseName | Stores in memory the case name assigned by the user. |
| Property-private | Get\_Name | Gets the circuit name. |
| Method-public | AddCktElement | Adds last DSS object created to circuit. |
| Method-public | ClearBusMarkers | Clears all the bus marker objects in the model. |
| Method-public | TotalizeMeters | Totalize all energymeters in the problem. |
| Property-public | ComputeCapacity | Calculates the system total capacity. |
| Property-public | Save | Save the present circuit - Enabled devices only. |
| Method-public | ProcessBusDefs | Processes the bus definitions in the circuit. |
| Method-public | ReProcessBusDefs | Redo all Buslists, nodelists. Includes memory reallocation if needed. |
| Method-public | DoResetMeterZones | Do this only if meterzones unlocked. Normally, Zones will remain unlocked so that all changes to the circuit will result in rebuilding the lists. |
| Property-public | SetElementActive | Sets the given element active in the global context for the active actor. |
| Method-public | InvalidateAllPCElements | Forces rebuild of matrix on next solution. |
| Method-public | DebugDump | Writes the debug file in the active folder. |
| Property-public | GetTopology | Access to topology from the first source. |
| Method-public | FreeTopology | Frees all the elements within the topology object. |
| Property-public | GetBusAdjacentPDLists | Returns the PD list adjacent to the active point in the topology tree. |
| Property-public | GetBusAdjacentPCLists | Returns the PC list adjacent to the active point in the topology tree. |
| Property-public | Tear\_Circuit | Tears the circuit considering the number of Buses of the original Circuit. |
| Property-public | Create\_MeTIS\_graph | Generates the graph describing the model for MeTiS. |
| Property-public | Create\_MeTIS\_Zones | Executes MeTiS and loads the zones into memory for further use. |
| Method-public | AggregateProfiles | Aggregates profiles using the number of zones defined by the user. |
| Method-public | Disable\_All\_DER | Disables all DER present in the model. |
| Method-public | Save\_SubCircuits | Saves the subcircuits created in memory into the hard drive. |
| Property-public | getPCEatBus | Returns the list of all PCE connected to the bus name given at BusName. |
| Property-public | getPDEatBus | Returns the list of all PDE connected to the bus name given at BusName. |
| Property-public | ReportPCEatBus | Gets all PCE at given bus and returns the list as string. |
| Property-public | ReportPDEatBus | Gets all PDE at given bus and returns the list as string. |
| Property-public | get\_Line\_Bus | Delivers the name of the bus at the specific line and terminal. |
| Method-public | get\_longest\_path | This routine calculates the longest path within a linearized graph considering the zero level buses as the beginning of new path. |
| Property-public | Append2PathsArray | Appends a new path to the array and returns the index(1D). |
| Method-public | Normalize\_graph | This routine normalizes the Inc\_matrix levels. |
| Method-public | Get\_paths\_4\_Coverage | Calculates the paths inside the graph to guarantee the desired coverage when tearing the system. |
| Method-public | Format\_SubCircuits | Arrange the files of the subcircuits to make them independent. |
| Method-public | AppendIsources | Appends single phase ISources to each node of bus specified if the given linkBranch. These actions take place within the given file. |
| Property-public | Name | PA Get\_Name. |
| Property-public | CaseName | PA FCaseName (variable) and Set\_CaseName. |
| Property-public | ActiveCktElement | PA FActiveCktElement (variable) and Set\_ActiveCktElement. |
| Property-public | Losses | PA Get\_Losses. |
| Property-public | BusNameRedefined | PA FBusNameRedefined (variable) and Set\_BusNameRedefined. |
| Property-public | LoadMultiplier | PA FLoadMultiplier (variable) and Set\_LoadMultiplier. |

## Other objects

Other objects around OpenDSS-X are not descendants of the main classes in the program’s architecture (TNamedObject, TDSSClass), however, they play an important role during the simulation execution given the information they carryout. These types of objects are normally subclasses of containers or circuit elements that user them for storing data that is important during the simulation.

The elements are enumerated in the following list with a brief description of what they are used to:

| *Object name* | *Description* |
| --- | --- |
| TControlQueue | Stores the pending control actions uploaded by controls after sampling. Controls can push and pull actions from the control queue. |
| TConductor | It’s the basic structure for a conductor in OpenDSS-X. |
| TExecutive | Implements all the routines needed for executing a command string written in OpenDSS-X syntax. It is also responsible for creating the default items for OpenDSS-X. Executive is the first element to be created when trying to implement an OpenDSS-X global instance. |
| TParser | Implements the DSS parser that interprets strings in OpenDSS-X syntax and returns their equivalents as integer, floating-point numbers, arrays, and other structures. It also helps to identify classes and elements by name while processing a script. |
| TRPNCalc | Implements the Reverse Polish Notation solver used for solving mathematical expressions in OpenDSS-X. |
| TCktTreeNode | Implements the structure for storing and managing the circuit tree for topological operations and navigation. |
| TCommandList | Implements the structure for storing the command list for an element when gets created. |
| THashList | Implements the structure for creating hash lists. |
| TPointerList | Implements the structure for storing pointers. The pointer list gets populated while creating the circuit and allows to gain access to an element within the circuit through its pointer. |
| TcMatrix | Implements the structure for defining matrices within OpenDSS-X. It also implements linear algebra operations that can be applied to the active matrix or between matrices. |
| TAutoAdd | Unit for processing the AutoAdd Solution FUNCTIONs. |
| TdJSON | Implements a JSON serializer for Delphi provided by Thomas Erlang. |

# Support routines and global context

There are libraries created for supporting specific OpenDSS-X functions that are required at different parts of the program. These libraries are implemented as subroutines (not objects) and depending on their purpose, are implement in specific files. The list of files and a brief description to them is given as follows:

| *File name* | *Description* |
| --- | --- |
| Diakoptics.pas | Implements the routines needed for solving the circuit model using spatial parallelization (A-Diakoptics). |
| DSSCallBackRoutines.pas | Implements the routines for connecting different routines across OpenDSS-X. |
| DSSForms.pas | Depending on the implementation (EXE, DLL, COM, Cmd), implements all the forms and graphical objects for providing feedback to the user. |
| DSSGlobals.pas | Includes all the global variables and routines to be used in the global context. It also includes the initialization for the global variables. |
| ExportResults.pas | Includes all the routines for exporting results and information about the simulation. |
| KLUsolve.pas | Implements the calls and connections to the solver DLL (KLUSolve). |
| Notes.pas | Contains notes from the developers, not actively updated. |
| ShowResults.pas | Includes all the routines for exporting results and information about the simulation. It is like ExportResult.pas, however, in this case, once the reports are exported, they will show up using the default editor (normally Notepad). |
| SolutionAlgs.pas | Implements the solution algorithms required for implementing the different solution modes available in OpenDSS-X. |
| TOPExport.pas | Implements the routines for exporting simulation results compatible with TOP. |
| Utilities.pas | Implements multiple routines for extracting circuit data in specific formats, fixing strings, navigate the circuit model, perform specific calculations based on the simulation results, among others. |
| YMatrix.pas | Implements the routines for creating, initializing, storing, and freeing sparse matrices. |
| Arraydef.pas | Implements the structures and pointers used across the program for storing arrays of elements. |
| Mathutil.pas | Implements special mathematical routines that can be used for translating electrical quantities into other formats. |
| UCmatrix.pas | Implements the structure and operation that can be applied to complex matrices in OpenDSS-X. |
| Ucomplex.pas | Implements linear algebra, domain conversion, and mathematical operations with complex numbers. |

# The solution algorithm

The OpenDSS-X program is designed to perform a basic distribution-style power flow in which the bulk power system is the dominant source of energy. However, it differs from the traditional radial circuit solvers in that it solves networked (meshed) distribution systems as easily as radial systems. It is intended to be used for distribution companies that may also have transmission or sub-transmission systems. Therefore, it can also be used to solve small- to medium-sized networks with a transmission-style power flow.

The power flow solution problem in OpenDSS-X is modeled as a fixed floating-point solution method based on a nodal admittance Y Bus matrix representing the elements within the power system, such as lines, transformers, loads, and capacitors. The inputs to the algorithm are the currents injected by active elements, or power injection elements, which represent substations, distributed generators, and every source of energy that injects current into the power grid.

The currents injected are not compatible with the currents flowing across the model, but represent the power feeder the grid, normally coming from shunt connected devices. By applying the injected currents in the form of a vector to the Y Bus matrix, the algorithm will return the voltage for all the nodes of the system. The nodes of the system refer to the connection points for each conductor/phase of the elements connected to the grid, allowing OpenDSS-X to consider unbalanced conditions mimicking the behavior of distribution power systems in real life. Then, the voltages calculated are compared to the previous solution until the algorithm finds convergence or reaches the maximum number of solution iterations. This last condition cancels the solution algorithm and next steps, reporting that the solution for the present simulation step cannot be found.

At each solution, shunt connected devices will modify the injected currents vector to compensate for the variations in voltage given the nominal voltage of the element. The value of the compensation current depends on the model (loads have eight different models) and the features of the element in time, such as the variability in the demand (loads) or variable generation. This solution algorithm can be depicted as follows:

At this point, the solution for one simulation step was found. No control actions have taken place and the solution only represents the voltages at the nodes of the power system model for the present time instant.

## Control Actions

The control actions are the actions performed by the controllers deployed across the power system to force the power system variables within a band specified by the power system operator. These controls comprehend generators, regulators, capacitor controls, smart inverters, and other control devices. The control actions depend on the latest solution obtained to determine if a control action needs to happen in the present simulation step.

If control actions need to be applied, they take place and then a new solution is calculated including the modifications triggered by the control actions, which can modify the Y Bus admittance matrix or the injected currents vector, depending on the controller type.

This loop repeats until no new control actions are required or the maximum number of control iterations is reached, which in that case will cancel the simulation and report the issue. Reaching the maximum number of control iterations signals an oscillatory behavior in the controls that requires user intervention for better setup or to coordinate the controllers.

When the solution is found successfully, meaning that there are no more control actions pending in the simulation step and the maximum number of solution/control iterations was not reached, the next step is to sample all the monitors, energy meters, and other sampling devices that register the electrical values obtained for the user-specified elements within the model.

This is the last step in the solution algorithm for the present simulation step, enabling the simulator to continue the simulation for the next time interval. This algorithm is generalized in the diagram shown in Figure 14.

Other processes around the solution algorithm include debugging logs to facilitate the controls and model tuning when something goes wrong. These are user-configurable options. The algorithm in Figure 14 represents the solution implemented in OpenDSS-X for QSTS simulations. Other solution modes, such as harmonics, use a similar structure with slight modifications.

A picture containing diagram

Description automatically generated

Figure . Generalized solution algorithm in OpenDSS-X

In Figure 14, the solution algorithm contains two main loops and three processes: the solution and control loops and the solution, control actions, and meter sampling processes. The loops represent the parts of the algorithm adding to the computational time required to find a solution. In this case, these loops comprehend two out of three processes of the solution algorithm that may need multiple iterations to complete.

The number of solution and control iterations depends on the power system model features. The maximum number of iterations is user configurable, providing flexibility for solving models with different difficulty levels. The difficulty level is determined by the circuit model’s topology and the type of elements deployed in the model, such as load models, generators, energy storage (ES), PV solar panels, and other technologies that can be connected to the model.

Control actions can also be time consuming in the simulation step. As mentioned above, control actions may require the recalculation of parts of the Y Bus matrix, adding an extra overhead that can be significant in respect to previous processes if a substantial number of controllers require similar control actions. Other controllers do not modify the Y Bus matrix. Instead, they are modeled as current injection devices that modify the power system response by injecting currents at the controlled element’s connection point.

# References

CMake. (2022, April 3). *Running CMake*. Retrieved from CMake: https://cmake.org/runningcmake

Dugan, R. C., & Montenegro, D. (2020). *The Open Distribution System Simulator (OpenDSS) Reference Guide.* Electric Power Research Institute. Washington, DC: Electric Power Research Institute. Retrieved from https://sourceforge.net/p/electricdss/code/HEAD/tree/trunk/Distrib/Doc/OpenDSSManual.pdf

Electric Power Research Institute. (2022, April 2). *OpenDSS*. Retrieved from EPRI: https://www.epri.com/pages/sa/opendss

Fu, F. (2019). *OpenDSS Tutorial and Cases.* Iowa State University, Department of Electrical and Computer Engineering. Ames, Iowa: Iowa State University. Retrieved from https://www.coursehero.com/file/87693420/EE653-OpenDSS-Tutorial-and-Casespptx

Hewitt, C., Woods, J., & Spurr, J. (2015). Inconsistency Robustness. College Publications.

# Acronyms

DER Distributed Energy Resource

DSS Distribution System Simulator

GUI Graphical User Interface

IDE Interactive Development Environment

OS Operating System

UI User Interface

PA Provides access to

1. OpenDSS is an EPRI software tool (open source) that utilizes network solution algorithms that have been proven and extensively used by EPRI, academic researchers and the electricity utility industry. [↑](#footnote-ref-2)