REFERENCE GUIDE

The Distribution System Simulator™ (DSS)

Roger C. Dugan
Sr. Consultant
Electric Power Research Institute, Inc.
September, 2008

License

Table of Contents

Figures

Summary

The Open Distribution System Simulator (DSS) is a comprehensive electrical system simulation tool for electric utility distribution systems. It is implemented as a COM DLL and designed to be driven from a variety of existing software platforms.

It basically supports all rms steady-state analyses commonly performed for distribution systems. In addition, it supports many new types of analyses that are designed to meet future needs, many of which are being dictated by the deregulation of US utilities and the formation of distribution companies worldwide. Many of the features are driven by distributed generation analysis needs. The DSS is designed to be indefinitely expandable so that it can be more easily modified to meet future needs.

Through the COM interface, the user is capable of performing all the functions of the simulator, including definition of the model data. Thus, the DSS is entirely independent of any database or text file circuit definition. It can be driven entirely from a MS Office tool through VBA, for example, or from any other 3rd party analysis program that can handle COM.

One way to think of the DSS is as an object-oriented database of power system circuit data that can perform.

The COM interface contains a text-based command interface as well as numerous COM methods and properties for accessing many of the parameters and functions of the simulator's models. Through the command line interface, users can prepare scripts to do several functions in sequence. The input may be redirected to a text file to accomplish the same effect as macros and also provide some database-like characteristics.

EXAMPLE: USAGE IN DISTCO *Suite*™

As an illustration of how the DSS software component might be used in an actual application, we will briefly consider the architecture of one of the applications in which EPRI uses it.

Figure 1 shows a block diagram of the Distco *Suite*™ architecture. The user interface in the Distco *Suite* system consists of a number of Microsoft Excel workbooks to aid with distribution planning.

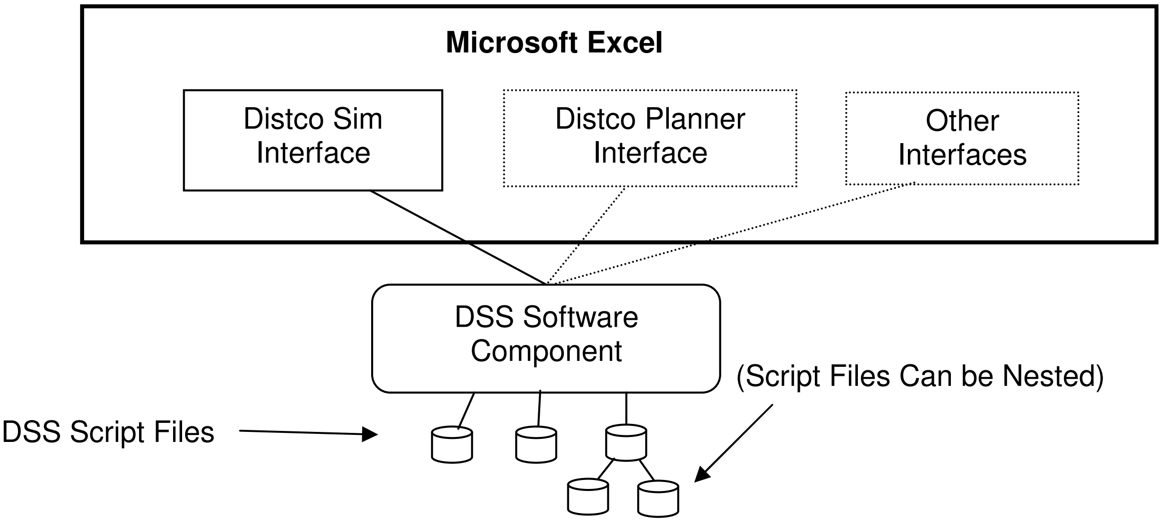


Figure 1: DistcoSuite Architecture

The Distco *Sim*™ interface is a workbook that provides a user interface for the Open Distribution System Simulator (DSS), which is a circuit simulation engine that provides no significant user interface itself. This workbook is used to drive the DSS and to present results from various circuit analysis tasks. Users many add custom worksheets to this workbook. One use is to prepare circuit models for planning analysis by other applications such as the Distco Planner application. Another use is to perform simulations for engineering purposes. Basically, each worksheet in the workbook can be considered to be a user interface form that one might have in another programming language such as VB or C++. In Excel, the DSS is driven through VBA "macros".

The DSS is implemented as a COM in-process server. It provides the basic electrical system simulation capability, but has only a simple user interface (available through the “panel” command or the ShowPanel method of the main DSS interface).

The fundamental way of controlling the DSS is through a text-based command interface. Thus, the entire operation of the DSS is scriptable. Scripts may be supplied through the COM interface or they may be placed in text files so that the DSS may read them more quickly. The files may be *nested.* That is, one script file may *redirect* the DSS input stream to another file. This feature is often used to provide circuit model descriptions.

With this kind of application, the user may develop other worksheets as needed In order to do this, it is necessary to write Visual Basic for Applications (VBA) code for transferring data to and from the DSS COM object. This reference describes the entire COM interface and the DSS command language.

Simulation Capabilities

The present version of the DSS is capable of performing the following analyses/simulations:

POWER FLOW

The DSS is designed to perform a basic distribution-style power flow in which the bulk power system is the dominant source of energy. It solves networked (meshed) systems as well as radial systems. It is intended to be used for distribution companies that may also have a transmission or subtransmission systems. Therefore, it can also be used to solve small- to medium-sized networks with a transmission-style power flow.

The circuit model employed can either a full multi-phase model or a positive-sequence model. The default is a full multi-phase model. For capacity studies, the latter is sufficient and will execute much faster. Due to the complex multi-phase models that may be created with myriad unbalances, the user is presently required to create positive-sequence models outside the DSS by defining a single-phase model of the circuit. However, by setting the proper flag, all power reports will report 3-phase quantities.

The power flow executes in numerous solution modes including the standard single snapshot mode, Daily mode, Dutycycle Mode, Monte Carlo mode, and several modes where the load varies as a function of time. (See the Help command for "Set Mode" for the up-to-date listing of solution modes). The time can be any arbitrary time period. Commonly, for planning purposes it will be a 24-hour day, a month, or a year. Users may also write external macros or programs to drive the load models in some other manner.

When a power flow is completed, the losses, voltages, flows, and other information are available for the total system, each component, and certain defined areas. For each instant in time, the losses are reported as kW losses, for example. Energy meter models may be used to integrate the power over a time interval,.

The power flow can be computed for both radial distribution (MV) circuits and network (meshed) systems. While the accuracy of some algorithms, such as the calculation of expected unserved energy, may depend on part of the circuit model being radial, the power flow solution is general. It works best on systems that have at least one stiff source.

The two basic power flow solution types are

1. Iterative power flow
2. Direct solution

For the iterative power flow, loads and distributed generators are treated as injection sources. In the Direct solution, they are included as admittances in the system admittance matrix, which is then solved directly without iterating. Either of these two types of solutions may be used for any of the several solution modes by setting the global LoadModel property to “Admittance” or “Powerflow” (can be abbreviated A or P).

There are two interative power flow algorithms currently employed:

1. "Normal" current injection mode
2. "Newton" mode.

The Normal mode is faster, but the Newton mode is more robust for circuit that are difficult to solve. The default is Normal.

Typically, power flow calculations will use an iterative solution and fault studies will use a direct solution by default.

FAULT STUDIES

The DSS will perform fault (short-circuit) studies in several ways:

- A conventional fault study for all buses ("Set Mode=Faultstudy"), reporting current and voltages on all phases for all types of faults: All-phase fault, SLG fault -- each phase, L-L and L-L-G faults. Since transformers will be represented in actual winding configuration, this is an excellent circuit model debugging tool as well as a tool for setting relays and sizing fuses.

- A single snapshot fault. User places a fault on the system at a particular bus, defining the type of fault and the value of the fault resistance. A fault is a circuit element just like any other and can be manipulated the same way.

- Applying faults randomly. (Monte Carlo fault study mode -- solution mode= "MF"). User defines Fault objects at locations where faults are desired. This is useful for such analyses as examining what voltages are observed at a DG site for various faults on the utility system.

LOAD PARAMETRIC VARIATION

The capabilities for doing parametric evaluation are provided for a variety of variables. Certain variables will be allowed to vary according to a function (e.g., load growth) or vary randomly for Monte Carlo and statistical studies. See the Set Mode command documentation for descriptions of the Monte Carlo solution modes.

DSS Control Panel

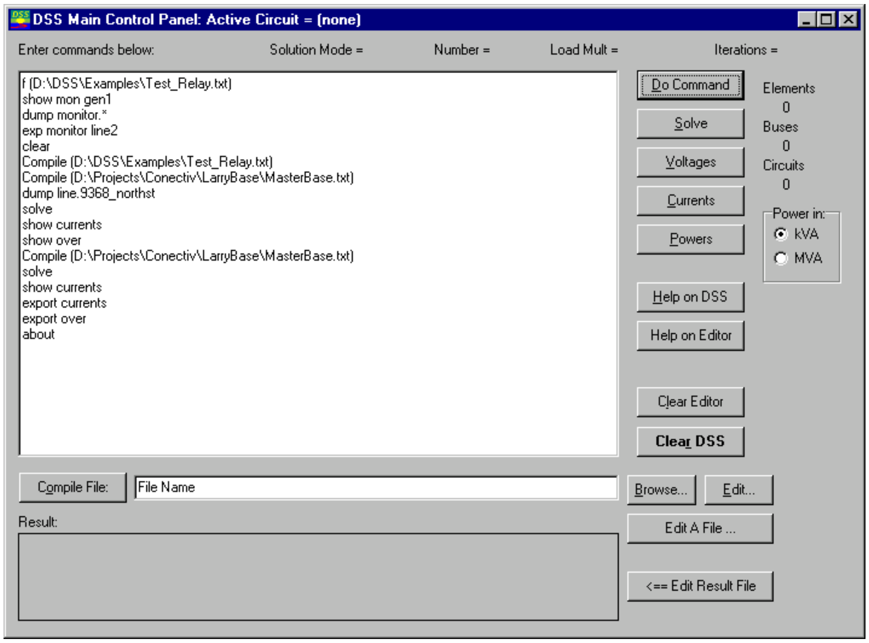
There is not an extensive, detailed user interface defined for the DSS as you might find with commercial shrink-wrapped programs. There is a "Control Panel" which is quite useful for various manual tasks, but access is generally through the COM interface or scripts.

Communication to the DSS is fundamentally accomplished through text strings passed to the DSS command interpreter. However, there are many functions provided in the COM interface to directly execute common commands (like "solve") and set properties of circuit elements without going through the command language.

Simulation results can be returned as arrays of values in the COM interface or in text of CSV files. Only a few standard text file reports will be provided by the base DSS software component (see Show and Export commands). The intent is for users to add customized reports through Excel worksheets or whatever application they use to control the DSS in special ways.

The DSS control panel a rudimentary user interface consisting of a single-screen Control Panel. The panel may also be invoked from the DSS COM library using the Panel or Show Panel commands or the ShowPanel method in the DSS COM interface.

(Caveat: When invoked from a MS Excel application, Excel may capture some of the key strokes, so it doesn't work as smoothly as it does in the standalone application.)



```
f (D:\DSS\Examples\Test_Relay.txt)
show mon gen1
dump monitor.*
exp monitor line2
clear
Compile (D:\DSS\Examples\Test_Relay.txt)
Compile (D:\Projects\Conectiv\LarryBase\MasterBase.txt)
dump line.9368_northst
solve
show currents
show over
Compile (D:\Projects\Conectiv\LarryBase\MasterBase.txt)
solve
show currents
export currents
export over
about
```

Figure 2: Control panel for the DSS

Export Menu Commands

The voltage and current exports were updated to be more consistent and useful when you load them into Excel or whatever.

- The output tables now stretch out horizontally instead of vertically. This is also done to fit in better with modifications to the Plot command
- Note: Current exports are keyed on device name
- Note: Voltage exports are keyed on bus name
- Residual values were added (sum of all conductors or nodes at a branch or bus). This is also true for the seq currents and seq voltage exports because the residuals can be different than the zero-sequence values.

Here are the export options:

- Export Voltages [Filename] (EXP_VOLTAGES.CSV)
- Export SeqVoltages [Filename] (EXP_SEQVOLTAGES.CSV)
- Export Currents [Filename] (EXP_CURRENTS.CSV)
- Export Overloads [Filename] EXP_OVERLOADS.CSV)
- Export Unserved [UEonly] [Filename] EXP_UNSERVED.CSV)
- Export SeqCurrents [Filename] (EXP_SEQCURRENTS.CSV)
- Export Powers [MVA] [Filename](EXP_POWERS.CSV)
- Export Faultstudy [Filename] (EXP_FAULTS.CSV)
- Export Generators [Filename | /m] (EXP_GENMETERS.CSV)
- Export Loads [Filename] (EXP_LOADS.CSV)
- Export Meters [Filename | /m] (EXP_METERS.CSV)
- Export Monitors monitorname (file name is assigned)

The circuit name is prepended onto the file names above.

At the conclusion of an Export operation (as well as any other DSS operation that writes a file) the name of the output file appears in the Result window. You can quickly display/edit the file by:

1. Edit | Result File menu item
2. ctrl-R
3. There is a new button on the tool bar for this purpose

Plot Menu Commands

The existing “circuit plot” creates displays on which the thicknesses of LINE elements are varied according to:

- Power
- Current
- Voltage
- Losses
- Capacity (remaining)

This plot is a unicolor plot (specified by color C1). The following example will display the circuit with the line thickness propotional to POWER relative to a max scale of 2000 kW:

```
plot circuit Power Max=2000 dots=n labels=n subs=n C1=$00FF0000
```

The existing “general plot” creates displays on which the color of bus dots are varied according to an arbitrary quantity entered through a CSV file . The following gives a nice yellow-red variation of data field 1 (the bus name is assumed to be in the first column of the csv file) with the min displayed as color C1 (yellow) and the max=0.003 being C2 (red). Values in between are various shades of orange.

```
plot General 1 Max=.003 dots=n labels=n subs=y object=filename.csv  C1=$0080FFFF C2=$000000FF
```

Example results: This plot clearly shows the extent of harmonic resonance involving the residual current (likely, the zero sequence, too) on the circuit.

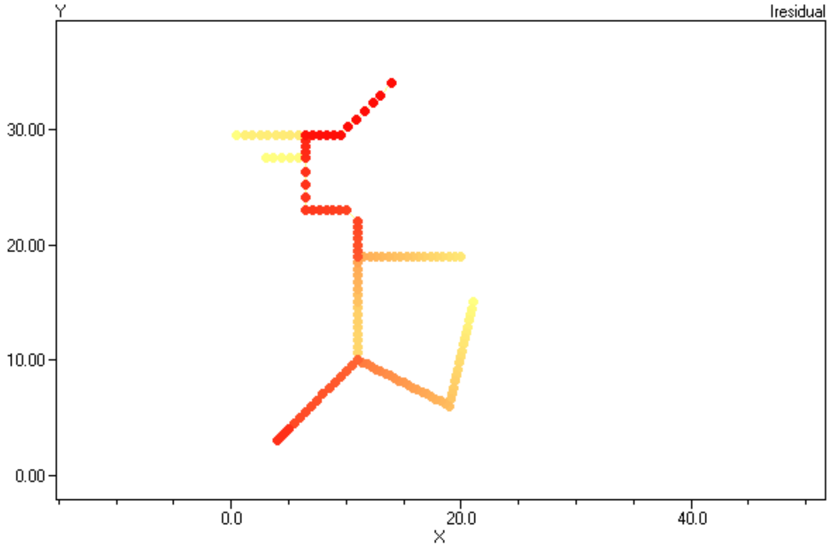


Figure 3: Color-shaded plot of harmonic resonance

There is also an option on the plot circuit command that will permit you to draw the circuit with lines proportional to some arbitrary quantity imported from a CSV file. The first column in the file should be the LINE name specified either by complete specification “line.elementname” or simply “elementname”. If the command can find a line by that name in the present circuit, it will plot it.

The usual thing you would do is to export a file that is keyed on branch name, such as

Export currents, or

Export seqcurrents

Then you would issue the plot command, for example:

```
Plot circuit quantity=1 Max=.001  dots=n  labels=n Object=feeder_exp_currents.CSV
```

The DSS differentiates this from the standard circuit plot by:

1. A numeric value for “quantity” (representing the field number) instead of “Power”, etc

2. A non-null specification for “Object” (the specified file must exist for the command to proceed).

The Plot > Circuit Plots > General Line Data menu will guide you through the creation of this command with menus and list prompts (turn the recorder on). The first line in the CSV file is assumed to contain the field names, which is common for CSV files.

For example, the same plot as above with line thicknesses instead of colored dots.

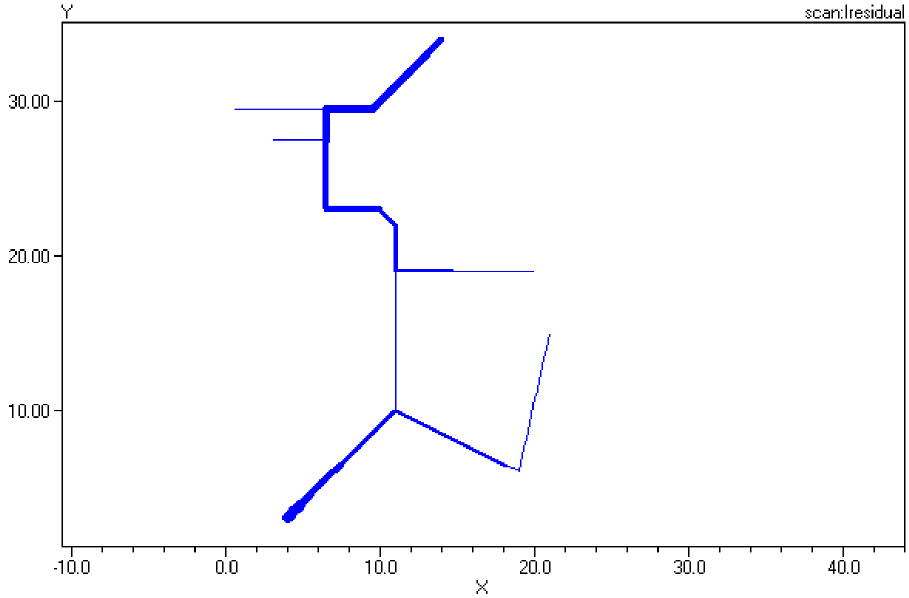


Figure 4: Thickness-weighted plot of harmonic resonance

Overall Circuit Model Concept

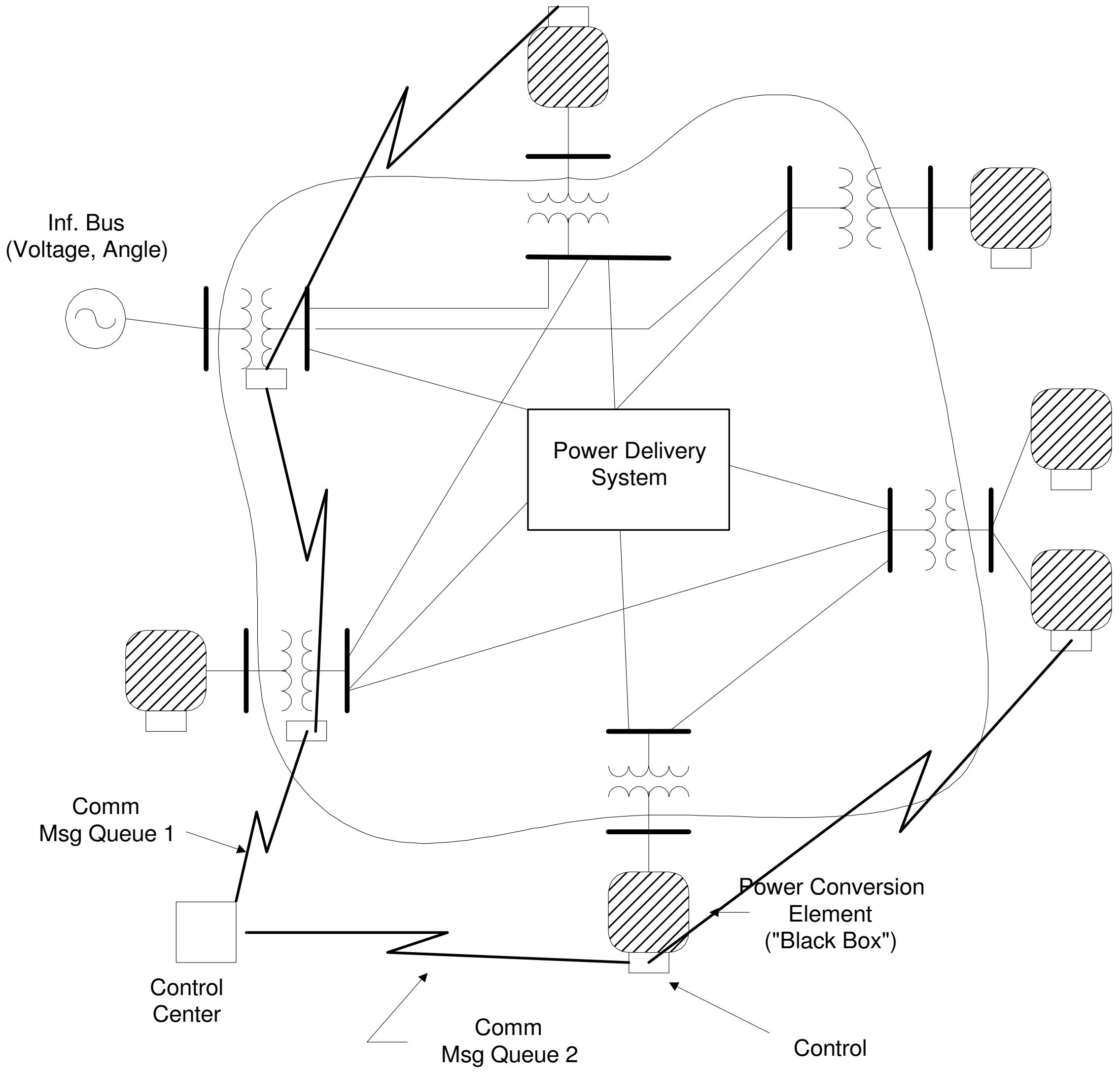


Figure 5: Electrical Circuit with Communications Network

The DSS consists of a model of the electrical system in the rms steady state, overlaid with a communications network that interconnects controls on power delivery elements and on power conversion elements.

[The communications message queues are not completely developed in this version -- one of the control queues is functional and used by the controls that are implemented.]

Bus and Terminal Models

Bus Definition

A bus is a circuit element having [1..N] *nodes*. Buses are the connection point for all other circuit elements.

The main electrical property of a Bus is voltage. Each node has a voltage with respect to the zero voltage reference (remote ground). There is a nodal admittance equation written for every *node* (i.e., the current is summed at each node).

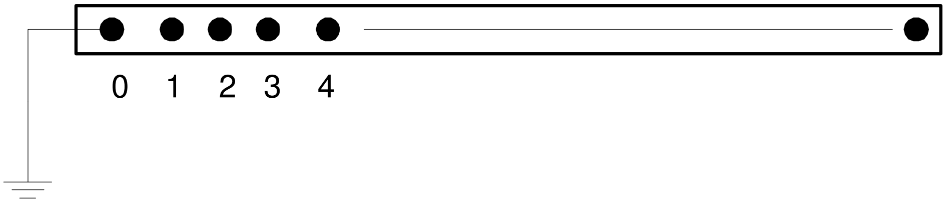


Figure 6: Bus Definition

Terminal Definition

Each electrical element in the power system has one or more terminals. Each terminal has one or more conductors. Each conductor contains a disconnect switch and a TCC (fuse) curve*[Fuse has been disabled and is being redesigned; a Relay object can be used if needed to control the switches]*. The conductors are numbered [1,2,3,…].

If the terminal is connected to an N-phase device, the first N conductors are assumed to correspond to the phases, in order. The remaining conductors may be neutrals or whatever.

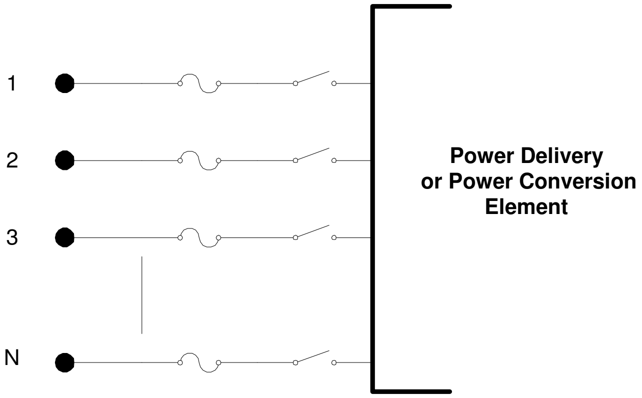


Figure 7: Terminal Definition

The DSS bus is a connecting place with 1 or more nodes for connecting individual phases and other conductors from the terminals of both power delivery elements and power conversion elements.

Buses are named with arbitrarily long null-terminated strings (no embedded blanks).

Node 0 of each bus is implicitly connected to the voltage reference (i.e., the node's voltage is always zero and is never explicitly included in the Y matrix).

BUS NAMING

Buses are named by an alphanumeric string of ASCII characters. The names may be numbers, but are always treated as strings. Internally, buses will be numbered (actually, each node is numbered), but will be referenced through the COM or command interface by name.

Names may not contain blanks, tabs, or other “white space” or control characters.

Names may be any length and are passed to the DSS through the COM interface as standard null-terminated strings. This is the common way for representing strings in the Windows environment. Thus, it is a simple matter to drive the DSS from most programming languages and, particularly, from Visual Basic.

TERMINAL REFERENCES

Terminals are not named separately from the device. Each device will have a name and a defined number of multiphase terminals. Terminals will be referenced explicitly by number [1, 2, 3 …] or by inference, in the sequence in which they appear. Internally, they will be sequenced by position in a list.

PHASES AND OTHER CONDUCTORS

Each terminal has one or more **phases,** or normal power-carrying conductors and, optionally, a number of other conductors to represent neutral, or grounding, conductors or conductors for any other purpose. By convention, if a device is declared as having N phases, the first N conductors of each terminal are assumed to be the phase conductors, in the same sequence on each terminal. Remaining conductors at each terminal refer to “neutral” or “ground” or "earth" conductors.

All terminals of a device are defined to have the same number of conductors. For most devices, this causes no ambiguity, but for transformers with both delta and wye (star) winding connections, there will be an extra terminal at the delta connection. The neutral is explicit to allow connection of neutral impedances to the wye-connected winding. The extra conductor for the delta connection is simply connected to ground (voltage reference) and the admittances are all set to zero. Thus, the conductor effectively does not appear in the problem; it is ignored. However, you may see it appear in reports that explicitly list all the voltages and currents in a circuit.

The terminal conductors and bus nodes may be combined to form any practical connection.

Bus Nodes: A bus may have any number of *nodes* (places to connect device terminal conductors). The nodes may be arbitrarily numbered, except that the first N are reserved for the N phases. Thus, if a bus has 3-phase devices connected to it, connections would be expected to nodes 1, 2, and 3. So the DSS would use these voltages to compute the sequence voltages, for example. Phase 1 would nominally represent the same phase throughout the circuit, although that would not be mandatory. It is up to the user to maintain a consistent definition. If only the default connections are used, the consistency is maintained automatically.

Any other nodes would simply be points of connection with no special meaning. Each Bus object keeps track of the allocation and designation of its nodes.

Node 0 of a bus is always the voltage reference (a.k.a, ground, or earth). That is, it always has a voltage of exactly zero volts.

SPECIFYING CONNECTIONS

The user can define how terminals are to be connected to buses in three ways, not just one way:

1. Generically connect a circuit element's *terminal* to a *bus* without specifying node-to-conductor connections. This is the default connection. Normal phase sequence is assumed. Phase 1 of the terminal is connected to Node 1 of the Bus, and so on. Neutrals default to ground (Node 0).

2. Explicitly specify the first phase of the device is connected to node *j* of the bus. The remaining phases are connected in normal 3-phase sequence (1-2-3 rotation). Neutral conductors default to ground (Node 0).

3. Explicitly specify the connection for all phases of each terminal. Using this mode, neutrals (star points) may be left floating. Any arbitrary connection maybe achieved. The syntax is:

   BUSNAME.*i.j.k,* where i, j, k refer to the nodes of a bus.

   This is interpreted as the first conductor of the terminal is connected to node i of the bus designated by BUSNAME; the 2nd conductor is connected to node j, etc.

   The default node convention for a terminal-to-bus connection specification in which the nodes are not explicitly designated is:

   BUSNAME.1.0.0.0.0.0.0. … {Single-phase terminal)
   BUSNAME.1.2.0.0.0.0.0. … {two-phase terminal)
   BUSNAME.1.2.3.0.0.0.0 … {3-phase terminal}

   If the desired connection is anything else, it must be explicitly specified. Note: a bus object "learns" its definition from the terminal specifications. Extra nodes are created on the fly as needed. They are simply designations of places to connect conductors from terminals. For a 3-phase wye-connected capacitor with a neutral reactor, specify the connections as follows:

BUSNAME.1.2.3.4 {for 3-phase wye-connected capacitor}
BUSNAME.4 {for 1-phase neutral reactor (2nd terminal defaults to node 0)}

Power Delivery Elements

Power delivery elements usually consist of two or more multiphase terminals. Their basic function is to transport energy from one point to another. On the power system, the most common power delivery elements are lines and transformers. Thus, they generally have more than one terminal (capacitors and reactors can be an exception when shunt-connected rather than series-connected). Power delivery elements are standard electrical elements generally completely defined in the rms steady state by their *impedances*

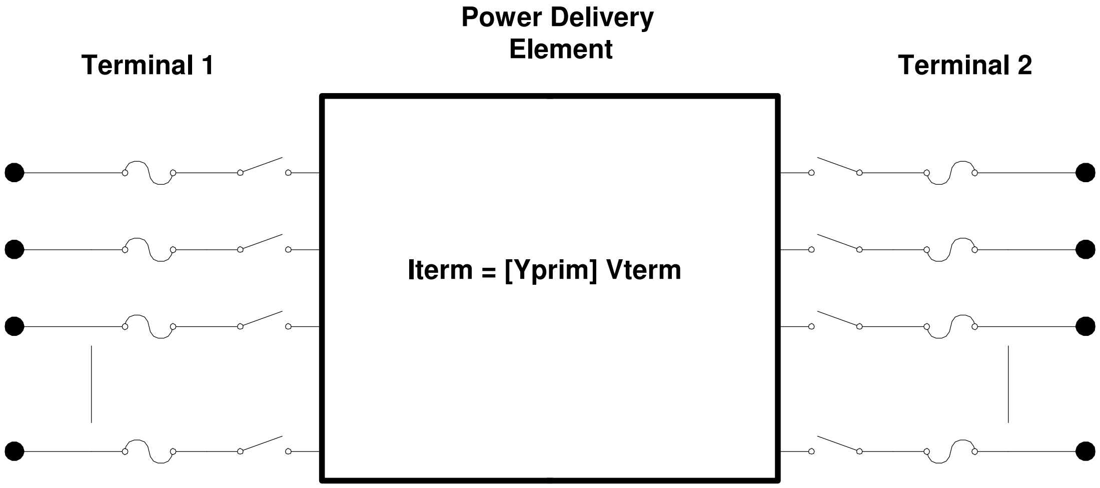


Figure 8: Power Delivery Element Definition

Power Conversion Elements

Power conversion elements convert power from electrical form to some other form, or vice-versa. Some may temporarily store energy and then give it back, as is the case for reactive elements. Most will have only one connection to the power system and, therefore, only one multiphase terminal. The description of the mechanical or thermal side of the power conversion is contained within the "Black box" model. The description may be a simple impedance or a complicated set of differential equations yielding a current injection equation of the form:

$$I_{Term}(t) = \mathbf{F}(V_{Term}, [State], t)$$

The function **F** will vary according to the type of simulation being performed. The power conversion element must also be capable of reporting the partials matrix when necessary:

$$\frac{\partial F}{\partial V}$$

In simple cases, this will simply be the primitive **y** (admittance) matrix; that is, the y matrix for this element alone.

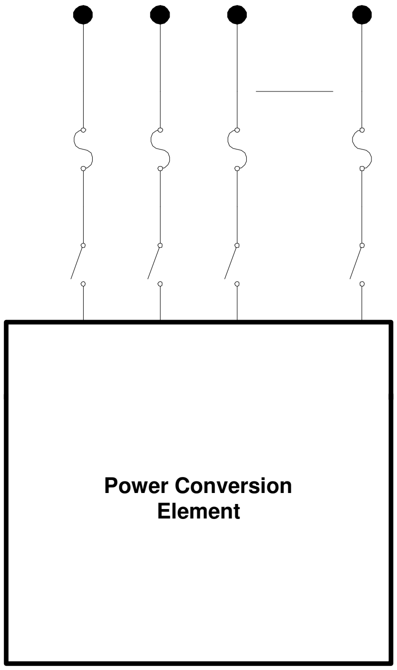


Figure 9: Power Conversion Element Definition

This concept may easily be extended to multi-terminal devices, which would allow the representation of complex series elements such as fault current limiters.

DSS Command Language Syntax

The DSS is designed such that all functions can be carried out through a text-based command language. The text streams may come either through the COM interface, or they may come from a standard text file to which the command interpreter may be temporarily diverted (Compile or Redirect commands).

This makes the DSS an easily accessible tool for users who simply want to key in a small circuit and do a quick study. However, it also makes the DSS more easily adapted by various commercial vendors who have a great deal invested in their own database and would not want to conform to another. Text files are the most common means of transferring data from one source into another.

Always refer to the **Help** command for the latest commands and property names that are recognized by the DSS.

COMMAND SYNTAX

The command language is of the form:

Command parm1, parm2 parm3 parm 4 ….

Parameters (parm1, etc) may be separated by commas (,) or white space (blank, tab). If a parameter includes a delimiter, enclose it in either double quotes ("), single quotes(') or parentheses (…).

PARAMETERS

Parameters may be *positional* or *named* (tagged). If named, an "=" sign is expected.

1. *Name=value* (this is the named form)
2. *Value* (value alone in positional form)

For example, the following two commands are equivalent.

```
New   Object="Line.First Line"   Bus1=b1240   Bus2=32   LineCode=336ACSR, …

New   (Line.First Line),  b1240   32   336ACSR, …
```

The first example uses named parameters, which are shown in the default order. The second example simply gives the values of the parameters and the parser assumes that they are in the default order. Note that the name of the object contains a blank, which is a delimiter character. Therefore, it is enclosed in quotes or parentheses.

You may mix named parameters and positional parameters. Using a named parameter repositions the parser's positional pointer so that subsequent parameters need not be named. The order of parameters is always given in the DSS help command window.

Some commands are interpreted at more than one lexical levels inside the DSS. In this example, the main DSS command interpreter interprets the **New** command and essentially passes the remainder of the string to the Executive for adding new circuit elements. It determines the type of element to add and passes the remainder on to the module that handles the instantiation and definitions of Line objects. Only the Line model code needs to know how to interpret the parameters it receives.

For the **New** command, the first two parameters are always required and positional:

1. The New command itself, and
2. The name of the object to add.

For circuit elements, the next one or two parameters are normally the bus connections, which are processed and stored with the circuit element model. Then the definition of the object being created continues, using an editing function expressly devoted to that class of circuit element.

ARRAY PARAMETERS

Array parameters are sequences of numbers. Of necessity, delimiters must be present to separate the numbers. To define an array, simply enclose the sequence in quotes (single or double) or parenthesis as you would for any parameter value containing a delimiter. For example, for a 3-winding transformer you could define arrays such as:

```
Kvs = (115, 6.6, 22)

Kvas=”20000  16000 16000”
```

MATRIX PARAMETERS

Matrix parameters are entered by extending the Array syntax: Simply place a bar (|) between rows, for example:

```
Xmatrix=”1.2  .3  .3 | .3  1.2  3 | .3  .3  1.2”    (3x3 matrix)
```

Symmetrical matrices like this example may also be entered in lower triangle form to be more concise.

```
Xmatrix=( 1.2  | .3 1.2  | .3  .3  1.2 )    (3x3 matrix – lower triangle)
```

STRING LENGTH

The strings may be as long as can be reasonably passed through the COM interface. They must not be split on separate “lines” since there is no concept of a line of text in the standard DSS COM interface; only separate commands.

When a command results in the instantiation of a circuit, or control element, the element will be instantiated with reasonable values. Only those parameters that need to be changed need be included in the command string. Also, a new element need not be defined in one command. It may be edited as many times as desired with subsequent commands.

When an element is created or selected by one command, it becomes the *Active* element. Thereafter, generic property edit commands are passed directly to the active element. In that respect, the command language will mirror the basic COM interface.

All changes are persistent. That is, a parameter changed with one command remains as it was defined until changed by a subsequent command.

DSS Command Reference

Nearly all DSS commands and parameter names may be abbreviated. This is for convenience when typing commands in directly. However, there is not necessarily any speed benefit to abbreviating for machine-generated text. The command names and parameter names are hashed. Therefore, they can generally be found faster if they are completely spelled out. (The algorithm first takes the string and searches the hash list first. Then it does a linear search looking for abbreviations, which can be slow.) Thus, commands and parameter names should be spelled out completely when placed in script files. Abbreviate only when manually typing commands to the DSS.

Specifying Objects

Any object in the DSS, whether a circuit element or a general DSS object, can be referenced by its complete name:

```
Object=Classname.objname
```

For example,

```
object=vsource.source.
```

In nearly all circumstances, the 'object=' may be omitted as it can for any other command line parameter. The object name is almost always expected immediately following the command verb.

If the 'classname' prefix is omitted (i.e., no dot in the object name), the previously used class (active DSS class) is assumed. For example,

```
New line.firstline . . .

New secondline . . .
```

The second command will attempt to create a new line object called 'secondline'.

If there is any chance the active class has been reset, use the fully qualified name. Alternatively, use the Set command to establish the active class:

```
Set class=Line

New secondline . . .
```

COMMAND DEFINITIONS

The following documents the command definitions as of this writing. Newer builds of the DSS may have additional properties and commands. Execute the Help command while running the DSS to view the present commands available in your version.

Help

Brings up a help screen with a "tree" view of all the commands and property names currently accepted by the DSS. (This is might be more up-to-date than the printed manuals if new features have been added.) Clicking on the commands will display a brief description of how to use the particular command or property.

//   (comment)

The appearance of "//" in the command position indicates that this statement is a comment line. It is ignored by the DSS. If you wish to place an in-line comment at the end of a command line, use the **"!"** character. The parser ignores all characters following the ! character.

// This is a comment line

New line.line4 linecode=336acsr length=2.0   **!** this is an in-line comment

New [Object] [Edit String]

Adds an element described on the remainder of the line to the active circuit. The first parameter (Object=…) is required for the New command. Of course, Object= may be omitted

The remainder of the command line is processed by the editing function of the specified element type. All circuit objects are instantiated with a reasonable set of values so that they can likely be included in the circuit and solved without modification. Therefore, the Edit String need only include definitions for parameter values that change.

Examples:

```
New Object=Line.Lin2    ! Min required

New Line.Lin2               ! Same, sans object= …

!Line from Bs1 to Bs2
New Line.Lin2 Bs1  Bs2  R1=.01 X1=.5 Length=1.3
```

The Edit String does not have to be complete at the time of issuing the New command. The object instantiated may be edited at any later time by invoking the Edit command.

Immediately after issuing the New command, the instantiated object remains the active object and the Edit command does not have to given to select the object.

You can simply issue the More command, or one of its abbreviations (~), and continue send editing instructions. Actually, the DSS command interpreter defaults to editing mode and you simply need only issue the command

Property=value …(and other editing statements)

The "=" is required. When the DSS parser sees this, it will assume you wish to continue editing and are not issuing a separate DSS Command. To avoid ambiguity, you may specify the element completely:

Class.ElementName.Property = Value

More than one property may be set on the same command, just as if you had issued the New or Edit commands.

Note that all DSS objects have a **Like** parameter. When another element of the same class is very similar to a new one being created, use the Like parameter to start the definition then change only the parameters that differ. Issue the Like=nnnn parameter first. Command lines are parsed from left to right with the later ones taking precedence. Throughout the DSS, the design goal is that a property remains in its last state until subsequently changed.

```
New Line.Lin3 like=Lin2 Length=1.7
```

Edit [Object] [Edit String]

Edits the object specified. The Type and Name fields are required and must designate a valid object (previously instantiated by a New command) in the problem. Otherwise, nothing is done and an error is posted.

The object named is passed the edit string to process.

More | M | ~ | [Edit String]

The More command continues editing the active object last selected by a New or Edit command. It simply passes the edit string to the object's editing function (actually, the function simply takes control of the parser after it is determined that this is a More command).

The More command may be abbreviated with simply **M** or ~.

Examples:

```
!Line from Bs1 to Bs2
New Line.Lin2 Bs1  Bs2
More R1=.01 X1=.5 Length=1.3

!Line from Bs1 to Bs2
New Line.Lin2 Bs1  Bs2  R1=.01 X1=.5 Length=1.3
M   C1=3.4  R1 = .02     ! define C1 and re-define R1
```

```
!Line from Bs1 to Bs2
New Object=Line.Lin2
~ Bus1=Bs1  Bus2=Bs2.2.3.1    ! Transposition line
~ R1=.01 X1=.5
~ Length=1.3
```

? [Object Property Name]

The “?” command allows you to query the present value of any published property of a DSS circuit element. For example,

? Monitor.mon1.mode

is a common command used to get the mode for the Monitor named Mon1.

The value is returned in the "Result" property of the DSS Text interface (See COM interface).

[Object Property Name] = value

This syntax permits the setting of any published property value of a DSS circuit element. Simply specify the complete property name, “=”, and a value. For example,

Monitor.mon1.mode=48

Sets the monitor mode queried in the previous example. The DSS command interpreter defaults to the Edit command. If it does not recognize the command it looks for the "=" and attempts to edit a property as specified. Thus, in this example, this method simply invokes the Monitor object’s editor and sets the value. If there is more text on the string, the editor continues editing. For example,

Line.line1.R1=.05 .12 .1 .4

Will set the R1, X1, R0, X0 properties of Line.line1 in sequence. This is a convenient syntax to use to change parameters in circuit elements that have already been defined.

Disable [Object]

Disables object in active circuit. All objects are **Enabled** when first defined. Use this command if you wish to temporarily remove an object from the active circuit, for a contingency case, for example. If this results in isolating a portion of the circuit, the voltages for those buses will be computed to be zero. (Also see Open, Close commands.)

Enable [Object]

Cancels a previous **Disable** command. All objects are automatically Enabled when first defined. Therefore, the use of this command is unnecessary until an object has been first disabled. (Also see Open, Close commands.)

Open [Object] [Term] [Cond]

Opens a specified terminal conductor switch. All conductors in the terminals of all circuit elements have an inherent switch. This command can be used to open one or more conductors in a specified terminal. If the 'Cond=' field is 0 or omitted, all *phase* conductors are opened. Any other conductors there might be are unaffected. Otherwise, open one conductor at a time (one per command). For example:

```
Open object=line.linxx  term=1      ! opens all phase conductors of terminal 1 of
linxx

Open line.linxx 2 3     ! opens 3rd conductor of 2nd terminal of linxx line object

Open load.LD3  1 4     ! opens neutral conductor of wye-connected 3-phase load
```

No action is taken if either the terminal or conductor specifications are invalid.

Note, this action disconnects the terminal from the node to which it is normally connected. The node remains in the problem. If it becomes isolated, a tiny conductance is attached to it and the voltage computes to zero. But you don't have to worry about it breaking the DSS.

Close [Object] [Term] [Cond]

Opposite of Open command.

Compile | Redirect [fileName]

The Compile and Redirect commands simply redirect the command interpreter to take input directly from a text file rather than from the Command property of the COM Text interface (the default method of communicating with the DSS). In DSS convention, use Compile when defining a new circuit and Redirect within script files to signify that the output is redirected to another file temporarily for the purpose of nesting script files. Use exactly the same syntax that you would supply to the Command property. Do not span lines; there is no line concept with the DSS commands. Each command must be on its own line in the file. Use the More command or its abbreviation "~" to continue editing a new line.

Set [prop1=value1] [prop2=value2]

Sets various global variables and options having to do with solution modes, user interface issues, and the like. Works like Edit command except that you don't specify object type and name. Properties that can be set with this command are:

Class (or **Type)=** sets class (type) for the Active DSS Object. This becomes the Active DSS Class.

Object (or **Name)=** sets the name of the Active DSS Object. Use the complete object specification (*classname.objname*), or simply the *objname,* to designate the active object which will be the target of the next command (such as the More command). If 'classname' is omitted, you can set the class by using the Class= field.

Hour= sets the hour to be used for the start time of the solution of the active circuit. (See also Time)

Sec= sets the seconds from the hour for the start time for the solution of the active circuit. (See also Time)

Year= sets the Year to be used for the next solution of the active circuit. Used to determine the growth multiplier for each load. Each load may have a unique growth curve (defined as a Growthshape object).

Freq= sets the frequency for the next solution of the active circuit.

Stepsize (or **h**)= sets the time step size (in sec) for the solution of the active circuit. Normally, specified for dynamic solution *(Not Yet Implemented!)* but is also, used for duty-cycle load following solutions. Yearly simulations go hour-by-hour and daily simulations follow the smallest increment of the daily load curves. (Note: defaults to 0.001 sec! This is nice for dynamics, but not for duty cycle)

LDcurve= name of the Loadshape object to use for the global circuit Load-Duration curve. Used in solution modes LD1 and LD2 (see below). Must be set before executing those modes. Simply define the load-duration curve as a loadshape object.

LoadModel= {"POWERFLOW" | "ADMITTANCE"} Sets the load model. If POWERFLOW (abbreviated P), loads do not appear in the System Y matrix. For iterative solution types (Mode ≠ Direct) loads (actually all PC Elements) are current injection sources. If ADMITTANCE, all PC elements appear in the System Y matrix and solution mode should be set to Direct (below) because there will be no injection currents.

LoadMult = global load multiplier to be applied to all "variable" loads in the circuit for the next solution. Loads designated as "fixed" are not affected. Note that not all solution modes use this multiplier, but many do, including all snapshot modes. See Mode below. The default LoadMult value is 1.0. Remember that it remains at the last value to which it was set. Solution modes such as Monte Carlo and Load-Duration modes will alter this multiplier. Its value is usually posted on DSS control panels. Loads defined with "status=fixed" are not affected by load multipliers. (The default for loads is "status=variable".)

Mode= specify the solution mode for the active circuit. Mode can be one of (unique abbreviation will suffice, as with nearly all DSS commands):

> **Snap**: Solve a single snapshot power flow for the present conditions. Loads are modified only by the global load multiplier (LoadMult) and the growth factor for the present year (Year).
>
> **Daily**: Do a series of solutions following the daily load curves. The Stepsize defaults to 3600 sec (1 hr). Set the starting hour and the number of solutions (e.g., 24) you wish to execute. Monitors are reset at the beginning of the solution. The peak of the daily load curve is determined by the global load multiplier (LoadMult) and the growth factor for the present year (Year).
>
> **Direct:** Solve a single snapshot solution using an admittance model of all loads. This is non-iterative; just a direct solution using the currently specified voltage and current sources.
>
> **Dutycycle**: Follow the duty cycle curves with the time increment specified. Perform the solution for the number of times specified by the Number parameter (see below).
>
> **FaultStudy:** Do a full fault study solution, determining the Thevenin equivalents for each bus in the active circuit. Prepares all the data required to produce fault study report under the Show Fault command.
>
> **Yearly**: Do a solution following the yearly load curves. The solution is repeated as many times as the specified by the Number= option. Each load then follows its yearly load curve. Load is determined solely by the yearly load curve and the growth multiplier. The time step is always 1 hour. Meters and Monitors are reset at the beginning of solution and sampled after each solution. If the yearly load curve is not specified, the daily curve is used and simply repeated if the number of solutions exceeds 24 hrs. This mode is nominally designed to support 8760-hr simulations of load, but can be used for any simulation that uses an hourly time step and needs monitors or meters.
>
> **LD1** (Load-Duration Mode 1): Solves for the joint union of a load-duration curve (defined as a Loadshape object) and the Daily load shape. Nominally performs a Daily solution (24-hr) for each point on the Load-duration (L-D) curve. Thus, the time axis of the L-D curve represents *days* at that peak load value. L-D curves begin at zero (0) time. Thus, a yearly L-D curve would be defined for 0..365 days. A monthly L-D curve should be defined for 0..31 days. Energy meters and monitors are reset at the beginning of the solution. At the conclusion, the energy meter

values represent the total of all solutions. If the L-D curve represent one year, then the energy will be for the entire year. This mode is intended for those applications requiring a single energy number for an entire year, month, or other time period. Loads are modified by growth curves as well, so set the year before proceeding. Also, set the L-D curve (see Ldcurve option).

LD2 (Load-Duration Mode 2): Similar to LD1 mode except that it performs the Load-duration solution for only a selected hour on the daily load shape. Set the desired hour before executing the Solve command. The meters and monitors are reset at the beginning of the solution. At the conclusion, the energy meters have only the values for that hour for the year, or month, or whatever time period the L-D curve represents. The solver simply solves for each point on the L-D curve, multiplying the load at the selected hour by the L-D curve value. This mode is used by the Distco *Planner™* application to generate a 3-D plot of energy vs. month and hour of the day.

M1 (Monte Carlo Mode 1): Perform a number of solutions allowing the loads to vary randomly. Executes number of cases specified by the Number option (see below). At each solution, each load is modified by a random multiplier -- a different one for each load. In multiphase loads, all phases are modified simultaneously so that the load remains balanced. The random variation may be uniform or gaussian as specified by the global Random option (see below). If uniform, the load multipliers are between 0 and 1. If gaussian, the multipliers are based on the mean and standard deviation of the Yearly load shape specified for the load. Be sure one is specified for each load.

M2 (Monte Carlo Mode 2): This mode is designed to execute a number of Daily simulations with the global peak load multiplier (LoadMult) varying randomly. Set time step size (h) and Number of solutions to run. "h" defaults to 3600 sec (1 Hr). Number of solutions refers to the number of *DAYS*. For Random = Gaussian, set the global %Mean and %Stddev variables, e.g. "Set %Mean=65 %Stddev=9". For Random=Uniform, it is not necessary to specify %Mean, since the global load multiplier is varied from 0 to 1. For each day, the global peak load multiplier is generated and then a 24 hour Daily solution is performed at the specified time step size.

M3 (Monte Carlo Mode 3): This mode is similar to the LD2 mode except that the global load multiplier is varied randomly rather than following a load-duration curve. Set the Hour of the day first (either Set Time=… or Set Hour=…). Meters and monitors are reset at the beginning of the solution. Energy at the conclusion of the solution represents the total of all random solutions. For example, one might use this mode to estimate the total annual energy at a given hour by running only 50 or 100 solutions at each hour (rather than 365) and ratioing up for the full year.

MF (Monte Carlo Fault mode). One of the faults defined in the active circuit is selected and its resistance value randomized. All other Faults are disabled. Executes number of cases specified by the Number parameter.

Peakdays: Do daily solutions (24-hr) only for those days in which the peak exceeds a specified value.

Random= specify the mode of random variation for Monte Carlo studies: May abbreviate value to "g" or "u" where **g** = gaussian (using mean and std deviation for load shape); **u** = uniform (varies between 0 and 1 randomly). Anything else:randomization disabled.

Number= specify the number of time steps or solutions to run or the number of Monte Carlo cases to run.

Time= Specify the solution start time as an array : time="hour, sec" or time = (hour, sec): e.g., time = (23, 370) designate 6 minutes, 10 sec past the 23rd hour.

Circuit= Set the active circuit (by name).

Editor= Set the command string required to start up the editor preferred by the user. Defaults to Notepad. This is used to display certain reports from the DSS. Note: on many Windows systems, Wordpad.exe is not on the path. However, Write frequently is and will start up Wordpad. Therefore, you may say "Editor=Write" if you wish to use Wordpad. Otherwise, you must locate Wordpad.exe and use the complete path name.

Tolerance= sets the solution tolerance. Default is 0.0001.

Maxiter= sets the maximum allowable iterations for the power flow solutions. Default is 15.

Vmin= Minimum permissible per unit voltage for normal conditions. Default is 0.95.

Vmax= Maximum permissible per unit voltage for normal conditions. Default is 1.05.

VminEmerg= Minimum permissible per unit voltage for emergency (contingency) conditions. Default is 0.90.

VmaxEmerg= Maximum permissible per unit voltage for emergency (contingency) conditions. Default is 1.08.

%mean= Percent mean for global load multiplier (LoadMult) when gaussian random variation is used. Default is 65%.

%stddev= Percent standard deviation for global load multiplier (LoadMult) when gaussian random variation is used. Default is 9%.

Ldcurve = Name of LoadShape object to be used as a load-duration curve. Must be defined prior to setting this value.

%Growth = Default annual growth rate, in percent. Used for loads which have no growth curves specified. Default is 2.5%.

GenkW = Size of generator, kW, to automatically add to system. Default is 1000.0

GenPF = Power factor of generator to assume for automatic addition. Default is 1.0.

Capkvar = Size of capacitor, kVAR, to automatically add to system. Default is 600.0.

Addtype = {Generator | Capacitor} Default is Generator. Type of device for AutoAdd Mode.

AllowDuplicates = {YES/TRUE | NO/FALSE} Default is No. Flag to indicate if it is OK to have devices of same name in the same class. If No, then a New command is treated as an Edit command but adds an element if it doesn't exist already. If Yes, then a New command will always result in a device being added.

ZoneLock = {YES/TRUE | NO/FALSE} Default is No. if No, then meter zones are recomputed each time there is a change in the circuit. If Yes, then meter zones are not recomputed unless they have not yet been computed. Meter zones are normally recomputed on Solve command following a circuit change.

Ueweight = Weighting factor for UE/EEN in AutoAdd functions. Defaults to 1.0. Autoadd mode minimizes

(Lossweight * Losses + UEweight * UE).

If you wish to ignore UE, set to 0. This applies only when there are EnergyMeter objects. Otherwise, AutoAdd mode minimizes total system losses.

LossWeight = Weighting factor for Losses in AutoAdd functions. Defaults to 1.0. Autoadd mode minimizes

(Lossweight * Losses + UEweight * UE).

If you wish to ignore Losses, set to 0. This applies only when there are EnergyMeter objects. Otherwise, AutoAdd mode minimizes total system losses.

Ueregs = Which EnergyMeter register(s) to use for UE in AutoAdd Mode. May be one or more registers. if more than one, register values are summed together. Array of integer values > 0. Defaults to 11 (for Load EEN).

For a list of EnergyMeter register numbers, do the "Show Meters" command after defining a circuit.

LossRegs = Which EnergyMeter register(s) to use for Losses in AutoAdd Mode. May be one or more registers. if more than one, register values are summed together. Array of integer values > 0. Defaults to 13 (for Zone kWh Losses).

For a list of EnergyMeter register numbers, do the "Show Meters" command after defining a circuit.

VoltageBases = Define legal voltage bases for this circuit. Enter an array of the legal voltage bases, for example:

```
set voltagebases=".208, .480, 12.47, 24.9, 34.5, 115.0, 230.0"
```

When the CALCVOLTAGEBASES command is issued, a snapshot solution is performed with no load injections and the bus base voltage is set to the nearest legal voltage base. The defaults are as shown in the example above.

The DSS does not used per unit values in its solution. You only need to set the voltage bases if you wish to see per unit values on the reports or if you intend to use the AutoAdd feature.

Algorithm = {Normal | Newton} Solution algorithm type. Normal is a fixed point current-injection iteration that is a little quicker (about twice as fast) than the Newton iteration. Normal is adequate for most systems. Newton is more robust for circuits that are difficult to solve.

Trapezoidal = {YES/TRUE | NO/FALSE} Default is "No". Specifies whether to use trapezoidal integration for accumulating energy meter registers. Applies to EnergyMeter and Generator objects. Default method simply multiplies the present value of the registers times the width of the interval. Trapezoidal is more accurate when there are sharp changes in a load shape or unequal intervals. Trapezoidal is automatically used for some load-duration curve simulations where the interval size varies considerably. Keep in mind that for Trapezoidal, you have to solve one more point than the number of intervals. That is, to do a Daily simulation on a 24-hr load shape, you would set Number=25 to force a solution at the first point again to establish the last (24th) interval.

AutoBusList = Array of bus names to include in AutoAdd searches. Or, you can specify a text file holding the names, one to a line, by using the syntax (file=filename) instead of the actual array elements. Default is null, which results in the program using either the buses in the EnergyMeter object zones or, if no EnergyMeters, all the buses, which can make for lengthy solution times.

Examples:

```
Set autobuslist=(bus1, bus2, bus3, ... )

Set autobuslist=(file=buslist.txt)
```

ControlMode = {OFF | STATIC |EVENT | TIME} Default is "STATIC". Control mode for the solution. Set to OFF to prevent controls from changing.

STATIC = Time does not advance. Control actions are executed in order of shortest time to act until all actions are cleared from the control queue. Use this mode for power flow solutions which may require several regulator tap changes per solution.

EVENT = solution is event driven. Only the control actions nearest in time are executed and the time is advanced automatically to the time of the event.

TIME = solution is time driven. Control actions are executed when the time for the pending action is reached or surpassed.

Controls may reset and may choose not to act when it comes their time.

Use TIME mode when modeling a control externally to the DSS and a solution mode such as DAILY or DUTYCYCLE that advances time, or set the time (hour and sec) explicitly from the external program.

TraceControl = {YES/TRUE | NO/FALSE} Set to YES to trace the actions taken in the control queue. Creates a file named TRACE_CONTROLQUEUE.CSV in the default directory. The names of all circuit elements taking an action are logged.

GenMult = Global multiplier for the kW output of every generator in the circuit. Default is 1.0. Applies to Snapshot, Daily, and DutyCycle solution modes. Ignored if generator is designated as Status=Fixed.

DefaultDaily = Default daily load shape name. Default value is "default", which is a 24-hour curve defined when the DSS is started.

DefaultYearly = Default yearly load shape name. Default value is "default", which is a 24-hour curve defined when the DSS is started. If no other curve is defined, this curve is simply repeated when in Yearly simulation mode.

AllocationFactors = Sets all allocation factors for all loads in the active circuit to the value given. Useful for making an initial guess or forcing a particular allocation of load. The allocation factors may be set automatically by the energy meter elements by placing energy meters on the circuit, defining the PEAKCURRENT property, and issuing the ALLOCATELOADS command.

CktModel = {Multiphase | Positive} Default = Multiphase. Designates whether circuit model is to interpreted as a normal multi-phase model or a positive-sequence only model. If Positive sequence, all power quantities are mulitiplied by 3 in reports and through any interface that reports a power quantity.

PriceSignal = Sets the price signal ($/MWh) for the circuit. Initial value is 25.

PriceCurve = Sets the curve to use to obtain for price signal. Default is none (null string). If none, price signal either remains constant or is set by an external process. Curve is defined as a loadshape (not normalized) and should correspond to the type of analysis being performed (daily, yearly, load-duration, etc.).

Terminal = Set the active terminal of the active circuit element. May also be done with Select command.

There are new Set command values added periodically. Check the Help on the DSS while it is running.

Get [prop1] [prop2] etc.

Basically, the opposite of the SET command. Returns DSS property values set using the Set command. Result is returned in the Result property of the Text interface.

VBA Example:

DSSText.Command = "Get mode"

Answer = DSSText.Result

Multiple properties may be requested on one get. The results are appended and the individual values separated by commas. Array values are returned separated by commas.

See help on Set command for property names.

Reset {Meters | Monitors }

{Monitors | Meters | (no argument) } Resets all Monitors or Energymeters as specified. If no argument specified, resets meters and monitors.

Show <Quantity>

Writes a text file report of the specified quantity for the most recent solution and opens a viewer (the default Editor -- e.g., Notepad or Wordpad) to display the file. Defaults to Show Voltages

Quantity can be one of:

Voltages - Shows the voltages at each node, grouped by bus.

Currents - Shows the currents into each device terminal.

Monitor <monitor name> - Shows a text (CSV) file with the voltages and currents presently stored in the specified monitor.

Powers - Shows power flow into each device terminal.

Faults - Shows results of Faultstudy mode solution: all-phase, one-phase, and adjacent 2-phase fault currents at each bus.

Elements - Shows all the elements in the active circuit.

Buses Shows all buses in the active circuit.

Panel - Same as Panel Command. Opens the internal DSS control panel.

Meter - shows the present values in the energy meter registers in the active circuit.

Generators - Each generator has its own energy meter. Shows the present values in each generator energy meter register in the active circuit.

Export <Quantity> [Filename or switch]

Writes a text file (.CSV) of the specified quantity for the most recent solution Defaults to Export Voltages. The purpose of this command is to produce a file that is readily readable by other programs such as those written in VB, spreadsheet programs, or database programs.

The first record is a header record providing a name for the field. The remaining records are for data. For example, the voltage export looks like this:

```
Bus, Node Ref., Node, Magnitude, Angle, p.u., Base kV
sourcebus   ,   1,    1,  6.6395E+0004,    0.0,    1.000,   115.00
sourcebus   ,   2,    2,  6.6395E+0004, -120.0,    1.000,   115.00
sourcebus   ,   3,    3,  6.6395E+0004,  120.0,    1.000,   115.00
subbus      ,   4,    1,  7.1996E+0003,   30.0,    1.000,    12.47
subbus      ,   5,    2,  7.1996E+0003,  -90.0,    1.000,    12.47
subbus      ,   6,    3,  7.1996E+0003,  150.0,    1.000,    12.47
```

This format is common for many spreadsheets and databases, although databases may require field types and sizes for direct import. The columns are aligned for better readability.

Valid syntax for the command can be one of the following statement prototypes in bold. If the Filename is omitted, the file name defaults to the name shown in italics in parantheses.

Export Voltages [Filename] ***(EXP_VOLTAGES.CSV)***. Exports voltages for every bus and active node in the circuit. (Magnitude and angle format).

Export SeqVoltages [Filename] ***(EXP_SEQVOLTAGES.CSV)*** Exports the sequence voltage magnitudes and the percent of negative- and zero-sequence to positive sequence.

Export Currents [Filename] ***(EXP_CURRENTS.CSV)*** Exports currents in magnitude and angle for each phase of each terminal of each device.

Export Overloads [Filename] ***EXP_OVERLOADS.CSV)*** Exports positive sequence current for each device and the percent of overload for each power delivery element that is overloaded.

Export SeqCurrents [Filename] ***(EXP_SEQCURRENTS.CSV)*** Exports the sequence currents for each terminal of each element of the circuit.

Export Powers [MVA] [Filename] ***(EXP_POWERS.CSV)*** Exports the powers for each terminal of each element of the circuit. If the MVA switch is specified, the result are specified in MVA. Otherwise, the results are in kVA units.

Export Faultstudy [Filename] ***(EXP_FAULTS.CSV)*** Exports a simple report of the 3-phase, 1-phase and max L-L fault at each bus.

Export Loads [Filename] ***(EXP_LOADS.CSV)*** Exports the follow data for each load object in the circuit: Connected KVA, Allocation Factor, Phases, kW, kvar, PF, Model.

Export Monitors monitorname ***(file name is assigned)*** Automatically creates a separate filename for each monitor. Exports the monitor record corresponding the monitor's mode. This will vary for different modes.

Export Meters [Filename | /multiple] ***(EXP_METERS.CSV)***
Export Generators [Filename | /multiple] ***(EXP_GENMETERS.CSV)***
EnergyMeter and Generator object exports are similar. Both export the time and the values of the energy registers in the two classes of objects. In contrast to the other Export options, each invocation of these export commands **appends** a record to the file. For Energymeter and Generator, specifying the switch "/multiple" (or /m) for the file name will cause a separate file to be written for each meter or generator. The default is for a single file containing all meter or generator elements.

Solve [set command options ...]

Executes the solution mode specified by the Set Mode = command. It may execute a single solution or hundreds of solutions. The Solution is a DSS object associated with each circuit. It has several properties that you must set to define which solution mode will be performed next. This command invokes the Solve method of the Solution object, which proceeds to execute the designated mode. You may also specify the Mode and Number options directly on the Solve command line if you wish: Solve Mode=M1 Number=1000, for example. Note that there is also a Solve method in the Solution interface in the DSS COM implementation. This is generally faster when driving the DSS from user-written code for a custom solution algorithm that must execute many solutions.

You may use the same options on the Solve command as with the Set command. For example:

```
Solve mode=daily   algorithm=newton
```

Dump <Circuit Element>

Writes a text file showing all the properties of the circuit object and displays it with the DSS text editor. You would use this command to check the definition of elements. It is also printed in a format that would allow it to be fed back into the DSS with no, or only minor, editing. If the “dump” command is used without an object reference, all elements in the active circuit are dumped to a file, which could be quite voluminous.

Clear

Clears all circuit definitions from the DSS.

About

Displays the "About" box. The Result string is set to the version string.

CalcVoltageBases

Calculates the voltage base for each bus based on voltage bases defined with "Set Voltagebases=..." array command. Performs a zero-current power flow considering only the series power-delivery elements of the system. The voltage base for the bus is then set to the nearest voltage base specified in the voltage base array. Alternatively, you may use the SetkVBase command to set the voltage base for each bus individually. The DSS does not need the voltage base for most calculations, but uses it for reporting. The exception is the AutoAdd solution mode where the program needs to specify the voltage rating of capacitors and generators it is adding to the system.

SetkVBase

Command to explicitly set the base voltage for a bus. Bus must be previously defined. Parameters in order are:

Bus = {bus name}

kVLL = (line-to-line base kV)

kVLN = (line-to-neutral base kV)

kV base is normally given in line-to-line kV (phase-phase). However, it may also be specified by line-to-neutral kV. The following exampes are equivalent:

```
setkvbase Bus=B9654 kVLL=13.2

setkvbase B9654 13.2

setkvbase B9654 kvln=7.62
```

BuildY

Forces rebuild of Y matrix upon next Solve command regardless of need. The usual reason for doing this would be to reset the matrix for another load level when using LoadModel=PowerFlow (the default) when the system is difficult to solve when the load is far from its base value. Works by invalidating the Y primitive matrices for all the Power Conversion elements.

Init

This command forces reinitialization of the solution for the next Solve command. To minimize iterations, most solutions start with the previous solution unless there has been a circuit change. However, if the previous solution is bad, it may be necessary to re-initialize. In most cases, a re-initialization results in a zero-load power flow solution with only the series power delivery elements considered.

Fileedit [filename]

Edit specified file in default text file editor (see Set Editor= option).

```
Fileedit EXP_METERS.CSV     (brings up the meters export file)
```

"FileEdit" may be abbreviated to a unique character string.

Voltages

Returns the voltages for the ACTIVE TERMINAL ONLY of the active circuit element in the Result string. For setting the active circuit element, see the Select command or the Set Terminal= property. Returned as magnitude and angle quantities, comma separated, one set per conductor of the terminal.

Currents

Returns the currents for each conductor of ALL terminals of the active circuit element in the Result string. (See Select command.) Returned as comma-separated magnitude and angle.

Powers

Returns the powers (complex) going into each conductors of ALL terminals of the active circuit element in the Result string. (See Select command.) Returned as comma-separated kW and kvar.

SeqVoltages

Returns the sequence voltages at all terminals of the active circuit element (see Select command) in Result string. Returned as comma-separated magnitude only values.Order of returned values: 0, 1, 2 (for each terminal).

SeqCurrents

Returns the sequence currents into all terminals of the active circuit element (see Select command) in Result string. Returned as comma-separated magnitude only values. Order of returned values: 0, 1, 2 (for each terminal).

SeqPowers

Returns the sequence powers into all terminals of the active circuit element (see Select command) in Result string. Returned as comma-separated kw, kvar pairs.Order of returned values: 0, 1, 2 (for each terminal).

Losses

Returns the **total** losses for the active circuit element (see Select command) in the Result string in kW, kvar.

PhaseLosses

Returns the losses for the active circuit element (see Select command) for each PHASE in the Result string in comma-separated kW, kvar pairs.

CktLosses

Returns the total losses for the active circuit in the Result string in kW, kvar.

AllocateLoads

Estimates the allocation factors for loads that are defined using the XFKVA property. Requires that energymeter objects be defined with the PEAKCURRENT property set. Loads that are not in the zone of an energymeter cannot be allocated. This command adjusts the allocation factors for the appropriate loads until the best match possible to the meter values is achieved. Loads are adjusted by phase. Therefore all single-phase loads on the same phase will end up with the same allocation factors.

If loads are not defined with the XKVA property, they are ignored by this command.

General DSS Object Property Descriptions

These are objects common to all circuits in the DSS (there may be several circuits loaded into memory at any one time). Any circuit can reference the data contained in the General DSS Objects.

The following describes each object class and the parameters that may be defined through the command interface.

Note that all DSS objects have a **Like** parameter. When another element of the same class is very similar to a new one being created, use the Like parameter to start the definition then change the parameters that differ. Issue the Like=nnnn parameter first. Command lines are parsed and executed from left to right.

LINECODE

Linecodes are objects that contain impedance characteristics for lines and cables. The term "line code" is an old term that simply refers to a code that was made up by programmers to describe a line construction. In most distribution analysis programs, one can describe a line by its linecode and its length. Linecodes were defined in a separate file. This collection of objects emulates the old linecode files, except that the concept is slightly more powerful.

Ultimately, the impedance of a line is described by its series impedance matrix and nodal capacitive admittance matrix. These matrices may be specified directly or they can be generated by specifying the symmetrical component data. Note that the impedances of lines may be specified directly and one does not need to use a line code, although the linecode will be more convenient most of the time. There may be hundreds of lines, but only a few different kinds of line constructions.

LineCode also performs a Kron reduction, reducing out the last conductor in the impedance matrices, which is assumed to be a neutral conductor. This applies only if the impedance is specified as a matrix. If the impedance is defined as symmetrical components, this function does not apply because symmetrical component values already assume the reduction.

By specifying the values of Rg, Xg, and rho, the DSS will take the base frequency impedance matrix values and adjust the earth return component for frequency. Skin effect in the conductors is not modified. To represent skin effect, you have to define the geometry.

This assumes the impedance matrix is constructed as follows:

$$Z = R + jX = \begin{bmatrix} Z_{11} + Zg & Z_{12} + Zg & Z_{13} + Zg \\ Z_{21} + Zg & Z_{22} + Zg & Z_{23} + Zg \\ Z_{31} + Zg & Z_{32} + Zg & Z_{33} + Zg \end{bmatrix}$$

Where, using the 1st term of Carson’s formulation,

$$Rg = \mu_0 \frac{\omega}{8} \quad \Omega / m$$

$$Xg = \mu_0 \frac{\omega}{2\pi} \ln\left(658.5\sqrt{\frac{\rho}{f}} \right) \quad \Omega / m$$

$$Xii = \mu_0 \frac{\omega}{2\pi} \ln\left(\frac{1}{GMRi} \right) \quad \Omega / m$$

$$Xij = \mu_0 \frac{\omega}{2\pi} \ln\left(\frac{1}{dij} \right) \quad \Omega / m$$

$$\mu_0 = 4\pi \times 10^{-7}$$

(All dimensional values in meters.)

The LineCode editing parameters, in order, are: (not case sensitive)

Nphases= Number of phases. Default = 3.

R1 = Positive-Sequence resistance, ohms per unit length.

X1 = Positive-Sequence reactance, ohms per unit length.

R0 = Zero-Sequence resistance, ohms per unit length.

X0 = Zero-Sequence reactance, ohms per unit length.

C1= Positive-Sequence capacitance, nanofarads per unit length

C0= Zero-Sequence capacitance, nanofarads per unit length

Units= {mi | km | kft | m | me} Length units. If not specified, it is assumed that the units correspond to the length being used in the Line models.

If any of the Symmetrical Component data is specified, the impedance matrices are determined from that data. You may also enter the matrices directly using the following three parameters. The order of matrices expected is the number of phases. The matrices may be entered in lower triangle form or full matrix. The result is always symmetrical. Matrices are specified by the syntax shown below. The "|" separates rows. Extraneous numbers on each row are ignored.

```
Rmatrix="11 | 21 22 | 31 32 33"
```

Rmatrix= Series resistance matrix, ohms per unit length..

Xmatrix= Series reactance matrix, ohms per unit length.

Cmatrix= Shunt nodal capacitance matrix, nanofarads per unit length.

BaseFreq= Base Frequency at which the impedance values are specified. Default = 60.0 Hz.

Normamps= Normal ampacity, amps.

Emergamps= Emergency ampacity, amps.

Faultrate= Number of faults per year per unit length. This is the default for this general line construction.

Pctperm= Percent of the fault that become permanent (requiring a line crew to repair and a sustained interruption).

Kron= Y/N. Default=N. Perform Kron reduction on the impedance matrix after it is formed, reducing order by 1. Do this only on initial definition after matrices are defined. Ignored for symmetrical components.

Rg= Carson earth return resistance per unit length used to compute impedance values at base frequency. For making better frequency adjustments. Default=0

Xg= Carson earth return reactance per unit length used to compute impedance values at base frequency. For making better frequency adjustments. Default=0

Rho= Earth resistivity used to compute earth correction factor. Default=100 meter ohms.

Like= Name of an existing LineCode object to build this like.

WIREDATA

This class of data defines the raw conductor data that is used to compute the impedance for a line geometry.

Note that you can use whatever units you want for any of the dimensional data – be sure to declare the units. Otherwise, the units are all assumed to match, which would be very rare for conductor data. Conductor data is usually supplied in a hodge-podge of units. Everything is converted to meters internally to the DSS

Rdc= dc Resistance, ohms per unit length (see Runits). Defaults to Rac if not specified.

Rac= Resistance at 60 Hz per unit length. Defaults to Rdc if not specified.

Runits= Length units for resistance: ohms per {mi|kft|km|m|Ft|in|cm } Default=none.

GMRac= GMR at 60 Hz. Defaults to .7788*radius if not specified.

GMRunits= Units for GMR: {mi|kft|km|m|Ft|in|cm } Default=none.

Radius= Outside radius of conductor. Defaults to GMR/0.7788 if not specified.

Radunits= Units for outside radius: {mi|kft|km|m|Ft|in|cm } Default=none.

Normamps= Normal ampacity, amperes. Defaults to Emergency amps/1.5 if not specified.

Emergamps= Emergency ampacity, amperes. Defaults to 1.5 * Normal Amps if not specified.

Diam= Diameter; Alternative method for entering radius.

Like= Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

LINEGEOMETRY

This class of data is used to define the positions of the conductors.

Nconds= Number of conductors in this geometry. Default is 3. Triggers memory allocations. Define first!

Nphases= Number of phases. Default =3; All other conductors are considered neutrals and might be reduced out.

Cond= Set this to number of the conductor you wish to define. Default is 1.

Wire= Code from WireData. MUST BE PREVIOUSLY DEFINED. no default.

X= x coordinate.

H= Height of conductor.

Units= Units for x and h: {mi|kft|km|m|Ft|in|cm } Initial default is "ft", but defaults to last unit defined

Normamps= Normal ampacity, amperes for the line. Defaults to first conductor if not specified.

Emergamps= Emergency ampacity, amperes. Defaults to first conductor if not specified.

Reduce= {Yes | No} Default = no. Reduce to Nphases (Kron Reduction). Reduce out neutrals.

Like= Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

LINE CONSTANT EXAMPLES

Define the wire data:

```
New Wiredata.ACSR336  GMR=0.0255000 DIAM=0.7410000 RAC=0.3060000
~  NormAmps=530.0000
~  Runits=mi radunits=in gmrunits=ft
New Wiredata.ACSR1/0  GMR=0.0044600 DIAM=0.3980000 RAC=1.120000
~   NormAmps=230.0000
~   Runits=mi radunits=in gmrunits=ft
```

Define the Geometry data:

```
New Linegeometry.HC2_336_1neut_0Mess  nconds=4 nphases=3
~ cond=1 Wire=acsr336  x=-1.2909 h=13.716 units=m
~ cond=2 Wire=acsr336  x=-0.502    h=13.716 !units=m
~ cond=3 Wire=acsr336  x=0.5737   h=13.716 !units=m
~ cond=4 Wire= ACSR1/0  x=0           h=14.648 ! units=m  ! neutral
```

Define a 300-ft line section:

```
New Line.Line1 Bus1=xxx   Bus2=yyy
~   Geometry= HC2_336_1neut_0Mess
~    Length=300   units=ft
```

Check out the line constants at 60 and 600 Hz in per km values. This command shows line constants for all defined geometries:

```
Show lineconstants freq=60 units=km
Show lineconstants freq=600 units=km
```

If the number of conductors = 3, this Show command will also give you the sequence impedances. If your geometry has more than 3 conductors and you want to see the sequence impedance, define

```
nconds = whatever
nphases = 3
Reduce=Yes
```

This will force the impedance matrices to be reduced to 3x3 and the Show LineConstants command will automatically give the sequence impedances. Note: make sure the phase conductors are defined first in the geometry definition.

No automatic assembly of bundled conductors is available yet. However, you can specifically define the position of each conductor in the geometry definition and connect them up explicitly in the DSS, for example, for a two conductor bundle:

```
New Line.2Bundled bus1=FromBus.1.1.2.2.3.3  ToBus.1.1.2.2.3.3
~ Geometry=2BundleGeometry  Length= etc. etc.
```

LOADSHAPE

A LoadShape object consists of a series of multipliers, nominally ranging from 0.0 to 1.0 that are applied to the base kW values of the load to represent variation of the load over some time period.

Load shapes are generally fixed interval, but may also be variable interval. For the latter, both the time, hr, and the multiplier must be specified.

All loadshapes, whether they be daily, yearly, or some arbitrary duty cycle, are maintained in this class. Each load simply refers to the appropriate shape by name.

The loadshape arrays may be entered directly in command line, or the load shapes may be stored in one of three different types of files from which the shapes are loaded into memory.

The command parameters for LoadShape objects are:

Npts= Number of points to expect when defining the curve

Interval= time interval of the data, Hr. Default=1.0. If the load shape has non-uniformly spaced points, specify the interval as 0.0.

Mult= Array of multiplier values. Looking for Npts values. To enter an array, simply enclose a series of numbers in double quotes "…", single quotes '…', or parentheses(..). Omitted values are assumed to be zero. Extra values are ignored. You may also use the syntax: **mult=(file=filename.ext)** in which the array values are entered one per line in the text file referenced.

Hour= Array of hour values corresponding to the multipliers. Not required if Interval>0. You may also use the syntax: **hour=(file=filename.ext)** in which the hour array values are entered one per line in the text file referenced. Again, this is not required for fixed interval load shape curves.

Mean= Mean of the multiplier array. The mean and standard deviation are always computed after an array of points are entered or normalized (see below). However, if you are doing only parametric load studies using the Monte Carlo solution mode, only the Mean and Std Deviation are required to define a loadshape. These two values may be defined directly rather than by supplying the curve. Of course, the multiplier points are not generated.

Stddev= Standard Deviation (see Mean, above).

The next three parameters instruct the LoadShape object to get its data from a file. Three different formats are allowed. If Interval>0 then only the multiplier is entered. For variable interval data, set Interval=0.0 and enter both the time (in hours) and multiplier, in that order for each interval.

Csvfile= Name of a CSV file containing load shape data, one interval to a line. For variable interval data enter one (hour, multiplier) point to a line with the values separated by a comma. Otherwise there is simply one value to a line.

Sngfile= Name of a binary file of single-precision floating point values containing the load shape data. The file is packed. For fixed interval data, the multipliers are packed in order. For variable interval data, start with the first hour point and alternate with the multiplier value.

Dblfile= Name of a binary file of double-precision floating point values containing the load shape data. The file is packed. For fixed interval data, the multipliers are packed in order. For variable interval data, start with the first hour point and alternate with the multiplier value.

Action= { Normalize} Many times the raw load shape data is in actual kW or some other unit. The load shapes normally will have a maximum value of 1.0. Specifying this parameter as "Action=N" *after* the load shape multiplier data are imported will force the normalization of the data in memory and recalculation of the mean and standard deviation.

Like= Name of an existing loadshape object to base this one on.

GrowthShape

A GrowthShape object is similar to a Loadshape object. However, it is intended to represent the growth in load year-by-year and the way the curve is specified is entirely different. You must enter the growth for the first year. Thereafter, only the years where there is a change must be entered. Otherwise it is assumed the growth stays the same.

Growth rate is specified by specifying the *multiplier* for the previous year's load. Thus, if the load grows 2.5% in 1999, the multiplier for that year will be specified as 1.025.

(If no growth shape is specified, the load growth defaults to the circuit's default growth rate -- see **Set %Growth** command).

The parameters are:

Npts= Number of points to expect when defining the curve.

Year= Array of year values corresponding to the multiplier values. Enter only those years in which the multiplier changes. Year data may be any integer sequence -- just so it's consistent for all growth curves. Setting the global solution variable Year=0 causes the growth factor to default to 1.0, effectively neglecting growth. This is what you would do for all base year analyses.

You may also use the syntax: **year=(file=filename.ext)** in which the array values are entered one per line in the text file referenced.

Mult= Array of Multiplier values corresponding to the year values. Enter multiplier by which the load will grow in this year.

Example:

```
New growthshape.name npts=3 year="1999 2003 2007" mult="1.10  1.05  1.025"
```

This defines a growthshape the a 10% growth rate for 1999-2002. Beginning in 2003, the growth tapers off to 5% and then to 2.5% for 2007 and beyond.

Normally, only a few points need be entered and the above parameters will be quite sufficient. However, provision has been made to enter the (year, multiplier) points from files just like the LoadShape objects. You may also use the syntax: **mult=(file=filename.ext)** in which the array values are entered one per line in the text file referenced.

Csvfile= Name of a csv file containing one (year, mult) point per line. Separate the year from the multiplier by a comma.

Sngfile= Name of a file of single-precision numbers containing (year, mult) points packed.

Dblfile= Name of a file of single-precision numbers containing (year, mult) points packed.

Like= Name of an existing GrowthShape object to base this one on.

TCC_CURVE

A TCC_Curve object is defined similarly to Loadshape and Growthshape objects in that they all are defined by curves consisting of arrays of points. Intended to model time-current characteristics for overcurrent relays, TCC_Curve objects are also used for other relay types requiring time curves. Both the time array and the C array must be entered.

The parameters are:

Npts= Number of points to expect when defining the curve.

C_Array= Array of current (or voltage or whatever) values corresponding to time values in T_Array (see T_Array).

T_Array= Array of time values in sec. Typical array syntax:

```
t_array = (1, 2, 3, 4, ...)
```

You may also substitute a file designation: **t_array = (file=filename).** The specified file has one value per line.

Like= Name of an existing GrowthShape object to base this one on.

DSS Circuit Element Object Descriptions

The following DSS objects are circuit elements. The DSS contains collections of each class that are treated as libraries of objects of each class. However, individual dircuit element objects are owned by a Circuit object. The following are descriptions of these types.

VSOURCE OBJECT

Voltage source. This is a special power conversion element. It is special because voltage sources must be identified to initialize the solution with all other injection sources set to zero.

A Vsource object is simply a multi-phase Thevenin equivalent with data specified as it would commonly be for a power system source equivalent: Line-line voltage (kV) and short circuit MVA.

The properties are, in order:

Bus1= Name of bus to which the source's one terminal is connected. Remember to specify the node order if the terminals are connected in some unusual manner.

basekv= base or rated Line-to-line kV.

pu= Actual per unit at which the source is operating. Assumed balanced for all phases.

Angle= Base angle, degrees, of the first phase.

Frequency= frequency of the source.

Phases= Number of phases. Default = 3.0.

Mvasc3= 3-phase short circuit MVA = $kVBase^2 / Z_{SC}$

Mvasc1- 1-phase short circuit MVA. There is some ambiguity concerning the meaning of this quantity For the DSS, it is defined as $kVBase^2 / Z_{1\text{-phase}}$ where

$$Z_{1\text{-phase}} = 1/3\ (2Z_1+Z_0)$$

Thus, unless a neutral reactor is used, it should be a number on the same order of magnitude as Mvasc3.

x1r1= Ratio of X1/R1. Default = 4.0.

x0r0= Ratio of X0/R0. Default = 3.0.

Isc3 = Alternate method of defining the source impedance. 3-phase short circuit current, amps. Default is 10000.

Isc1 = Alternate method of defining the source impedance. single-phase short circuit current, amps. Default is 10500.

R1 = Alternate method of defining the source impedance. Positive-sequence resistance, ohms. Default is 1.65.

X1 = Alternate method of defining the source impedance. Positive-sequence reactance, ohms. Default is 6.6.

R0 = Alternate method of defining the source impedance. Zero-sequence resistance, ohms. Default is 1.9.

X0 = Alternate method of defining the source impedance. Zero-sequence reactance, ohms. Default is 5.7.

BaseFreq = Base Frequency for impedance specifications. Default is 60 Hz.

like= Name of an existing Vsource object on which to base this one.

LINE OBJECT

Multi-phase, two-port line or cable. Pi model. Power delivery element described by its impedance. Impedances may be specified by symmetrical component values or by matrix values. Alternatively, you may simply refer to an existing LineCode object from which the impedance values will be copied. Then you need only specify the length.

You can define the line impedance at a base frequency directly in a Line object definition or you can import the impedance definition from a LineCode object. Both of these definitions of impedance are quite similar except that the LineCode object can perform Kron reduction. (See LineCode Object).

If the geometry property is specified all previous definitions are ignored. The DSS will compute the impedance matrices from the specified geometry each time the frequency changes.

Whichever definition is the most recent applies, as with nearly all DSS functions.

Note the units property; you can declare any length measurement in whatever units you please. Internally, everything is converted to meters. Just be sure to declare the units. Otherwise, they are assumed to be compatible with other data or irrelevant.

The properties, in order, are:

bus1= Name of bus for terminal 1. Node order definitions optional.

bus2= Name of bus for terminal 2.

Linecode= name of an existing LineCode object containing impedance definitions.

Length= Length multiplier to be applied to the impedance data.

Phases= No. of phases. Default = 3. A line has the same number of conductors per terminal as phases. Neutrals are not explicitly modeled unless declared as a phase and the impedance matrices adjusted accordingly. If you don't understand this, you probably shouldn't do it.

R1= positive-sequence resistance, ohms per unit length.

X1= positive-sequence reactance, ohms per unit length.

R0= zero-sequence resistance, ohms per unit length.

X0= zero -sequence reactance, ohms per unit length.

C1= positive-sequence capacitance, nanofarads per unit length.

C0= zero-sequence capacitance, nanofarads per unit length.

Normamps= Normal ampacity, amps.

Emergamps= Emergency ampacity, amps. Usually the one-hour rating.

Faultrate= Number of faults per year per unit length. This is the default for this general line construction.

Pctperm= Percent of the faults that become permanent (requiring a line crew to repair and a sustained interruption).

Repair= Hours to repair.

BaseFreq= Base Frequency at which the impedance values are specified. Default = 60.0 Hz.

Rmatrix= Series resistance matrix, ohms per unit length. See Command Language for syntax. Lower triangle form is acceptable.

Xmatrix= Series reactance matrix, ohms per unit length.

Cmatrix= Shunt nodal capacitance matrix, nanofarads per unit length.

Switch= {y/n | T/F} Default= no/false. Designates this line as a switch for graphics and algorithmic purposes. SIDE EFFECT: Sets R1=0.001 X1=0.0. You must reset if you want something different.

Rg= Carson earth return resistance per unit length used to compute impedance values at base frequency. For making better frequency adjustments. Default=0

Xg= Carson earth return reactance per unit length used to compute impedance values at base frequency. For making better frequency adjustments. Default=0

Rho= Earth resistivity used to compute earth correction factor. Overrides Line geometry definition if specified. Default=100 meter ohms.

Geometry= Geometry code for LineGeometry Object. Supercedes any previous definition of line impedance. Line constants are computed for each frequency change or rho change. CAUTION: may alter number of phases.

Units= Length Units = {none | mi|kft|km|m|Ft|in|cm } Default is None - assumes length units match impedance units.

Like= Name of an existing Line object to build this like.

LOAD OBJECT

A Load is a complicated Power Conversion element that is at the heart of many analyses. It is basically defined by its nominal kW and PF or its kW and kvar. Then it may be modified by a number of multipliers, including the global circuit load multiplier, yearly load shape, daily load shape, and a dutycycle load shape.

The default is for the load to be a current injection source. Thus, its primitive Y matrix contains only the impedance that might exist from the neutral of a wye-connected load to ground. However, if the load model is switched to Admittance from PowerFlow (see Set LoadModel command), the load is converted to an admittance and included in the system Y matrix. This would be the model used for fault studies where convergence might not be achieved because of low voltages.

Loads are assumed balanced for the number of phases specified. If you would like unbalanced loads, enter separate single-phase loads.

There are three legal ways to specify the base load:

1. kW, PF
2. kw, kvar
3. kVA, PF

If you sent these properties in the order shown, the definition should work. If you deviate from these procedures, the result may or may not be what you want. (To determine if it has accomplished the desired effect, execute the Dump command for the desired load(s) and observe the settings.)

The properties, in order, are:

bus1= Name of bus to which the load is connected. Include node definitions if the terminal conductors are connected abnormally. 3-phase Wye-connected loads have 4 conductors; Delta-connected have 3. Wye-connected loads, in general, have one more conductor than phases. 1-phase Delta has 2 conductors; 2-phase has 3. The remaining Delta, or line-line, connections have the same number of conductors as phases.

Phases= No. of phases this load.

Kv= Base voltage for load. For 2- or 3-phase loads, specified in phase-to-phase kV. For all other loads, the actual kV across the load branch. If wye (star) connected, then phase-to-neutral (L-N). If delta or phase-to-phase connected, they phase-to-phase (L-L) kV.

Kw= nominal kW for load. Total of all phases.

Pf= nominal Power Factor for load. Negative PF is leading. Specify either PF or kvar (see below). If both are specified, the last one specified takes precedence.

Model= Integer defining how the load will vary with voltage. Presently defined models are:

1: Normal load-flow type load: constant P and Q.
2: Constant impedance load
3: Constant P, Quadratic Q (somewhat like a motor)
4: Linear P, Quadratic Q (Mixed resistive, motor)
5: Rectifier load (Constant P, constant current)
6: Constant P; Q is fixed at nominal value
7: Constant P; Q is fixed impedance at nominal value

"Constant" values may be modified by loadshape multipliers. "Fixed" values are always the same -- at nominal, or base, value.

Yearly= Name of Yearly load shape.

Daily= Name of Daily load shape.

Duty= name of Duty cycle load shape. Defaults to Daily load shape if not defined.

Growth= Name of Growth Shape. Growth factor defaults to the circuit's default growth rate if not defined. (see Set %Growth command)

Conn= {wye | y | LN} for Wye (Line-Neutral) connection; {delta | LL} for Delta (Line-Line) connection. Default = wye.

Kvar= Base kvar. If this is specified, supercedes PF. (see PF)

Rneut= Neutral resistance, ohms. If entered as negative, non-zero number, neutral is assumed open, or ungrounded. Ignored for delta or line-line connected loads.

Xneut= Neutral reactance, ohms. Ignored for delta or line-line connected loads. Assumed to be in series with Rneut value.

Status= {fixed | variable}. Default is variable. If fixed, then the load is not modified by multipliers; it is fixed at its defined base value.

Class= Integer number segregating the load according to a particular class.

Vminpu = Default = 0.95. Minimum per unit voltage for which the MODEL is assumed to apply. Below this value, the load model reverts to a constant impedance model.

Vmaxpu = Default = 1.05. Maximum per unit voltage for which the MODEL is assumed to apply. Above this value, the load model reverts to a constant impedance model.

VminNorm = Minimum per unit voltage for load EEN evaluations, Normal limit. Default = 0, which defaults to system "vminnorm" property (see Set Command under Executive). If this property is specified, it ALWAYS overrides the system specification. This allows you to have different criteria for different loads. Set to zero to revert to the default system value.

VminEmerg = Minimum per unit voltage for load UE evaluations, Emergency limit. Default = 0, which defaults to system "vminemerg" property (see Set Command under Executive). If this property is specified, it ALWAYS overrides the system specification. This allows you to have different criteria for different loads. Set to zero to revert to the default system value.

XfkVA = Default = 0.0. Rated kVA of service transformer for allocating loads based on connected kVA at a bus. Side effect: kW, PF, and kvar are modified. See PeakCurrent property of EnergyMeter. See also AllocateLoads Command.

AllocationFactor = Default = 0.5. Allocation factor for allocating loads based on connected kVA at a bus. Side effect: kW, PF, and kvar are modified by multiplying this factor times the XFKVA (if > 0). See also AllocateLoads Command.

kVA= Definition of the Base load in kVA, total all phases. This is intended to be used in combination with the power factor (PF) to determine the actual load.

Basefreq= Base frequency for which this load is defined. Default is 60.0.

Like= Name of another Load object on which to base this one.

GENERATOR OBJECT

A Generator is a Power Conversion element similar to a Load object. Its rating is basically defined by its nominal kW and PF or its kW and kvar. Then it may be modified by a number of multipliers, including the global circuit load multiplier, yearly load shape, daily load shape, and a dutycycle load shape.

The generator is essentially a negative load that can be dispatched.

If the dispatch value (DispValue property) is 0, the generator always follows the appropriate dispatch curve, which is simply a Loadshape object. If DispValue>0 then the generator only comes on when the global circuit load multiplier exceeds DispValue. When the generator is on, it always follows the dispatch curve appropriate for the type of solution being performed.

If you want to model a generator that is fully on whenever it is dispatched on, simply designate "Status=Fixed". The default is "Status=Variable" (i.e., it follows a dispatch curve. You could also define a dispatch curve that is always 1.0.

Generators have their own energy meters that record:

1. Total kwh
2. Total kvarh
3. Max kW
4. Max kVA
5. Hours in operation
6. $ (Price signal * energy generated)

Generator meters reset with the circuit energy meters and take a sample with the circuit energy meters as well. The Energy meters also used trapezoidal integration so that they are compatible with Load-Duration simulations.

Generator power models are:

1. Constant P, Q (* dispatch curve, if appropriate).
2. Constant Z (For simple, approximate solution)
3. Constant P, |V| somwhat like a standard power flow
4. Constant P, fixed Q. P follows dispatch; Q is always the same.
5. Constant P, fixed reactance. P follows dispatch, Q is computed as if it were a fixed reactance.

Most of the time you will use #1 for planning studies.

The default is for the generator to be a current injection source. Thus, its primitive Y matrix contains only the impedance that might exist from the neutral of a wye-connected generator to ground. However, if the generator model is switched to Admittance from PowerFlow (see Set Mode command), the generator is converted to an admittance and included in the system Y matrix.

Generators are assumed balanced for the number of phases specified. If you would like unbalanced generators, enter separate single-phase generators.

The properties, in order, are:

bus1= Name of bus to which the generator is connected. Include node definitions if the terminal conductors are connected unusually. 3-phase Wye-connected generators have 4 conductors; Delta-connected have 3. Wye-connected generators, in general, have one more conductor than phases. 1-phase Delta has 2 conductors; 2-phase has 3. The remaining Delta, or line-line, connections have the same number of conductors as phases.

Phases= No. of phases this generator.

Kv= Base voltage for generator. For 2- or 3-phase generators, specified in phase-to-phase kV. For all other generators, the actual kV across the generator branch. If wye (star) connected, then phase-to-neutral (L-N). If delta or phase-to-phase connected, they phase-to-phase (L-L) kV.

Kw= nominal kW for generator. Total of all phases.

Pf= nominal Power Factor for generator. Negative PF is leading. Specify either PF or kvar (see below). If both are specified, the last one specified takes precedence.

Model= Integer defining how the generator will vary with voltage. Presently defined models are:

1: Normal load-flow type load: constant P and Q.
2: Constant impedance generator
3: Constant P, |V| like a conventional transmission power flow *[not implemented in this version]*

Yearly= Name of Yearly load shape.

Daily= Name of Daily load shape.

Duty= name of Duty cycle load shape. Defaults to Daily load shape if not defined.

Dispvalue= Dispatch value. If = 0.0 then Generator follows dispatch curves. If > 0 then Generator is ON only when the global load multiplier exceeds this value. Then the generator follows dispatch curves (see also Status)

Conn= {<u>w</u>ye | y | LN} for Wye (Line-Neutral) connection; {<u>d</u>elta | LL} for Delta (Line-

Line) connection. Default = wye.

Kvar= Base kvar. If this is specified, supercedes PF. (see PF)

Rneut= Neutral resistance, ohms. If entered as negative, non-zero number, neutral is assumed open, or ungrounded. Ignored for delta or line-line connected generators. Default is 0.

Xneut= Neutral reactance, ohms. Ignored for delta or line-line connected generators. Assumed to be in series with Rneut value.

Status= {fixed | variable}. If Fixed, then dispatch multipliers do not apply. The generator is alway at full power when it is ON. Default is Variable (follows curves).

Class= Integer number segregating the generator according to a particular class.

Maxkvar = Maximum kvar limit for Model = 3. Defaults to twice the specified load kvar. Always reset this if you change PF or kvar properties.

Minkvar = Minimum kvar limit for Model = 3. Enter a negative number if generator can absorb vars. Defaults to negative of Maxkvar. Always reset this if you change PF or kvar properties.

Pvfactor = Convergence deceleration factor for P-V generator model (Model=3). Default is 0.1. If the circuit converges easily, you may want to use a higher number such as 1.0. Use a lower number if solution diverges. Use Debugtrace=yes to create a file that will trace the convergence of a generator model.

Debugtrace = {Yes | No } Default is no. Turn this on to capture the progress of the generator model for each iteration. Creates a separate file for each generator named "GEN_name.CSV".

Vminpu = Default = 0.95. Minimum per unit voltage for which the Model is assumed to apply. Below this value, the generator model reverts to a constant impedance model.

Vmaxpu = Default = 1.05. Maximum per unit voltage for which the Model is assumed to apply. Above this value, the generator model reverts to a constant impedance model.

ForceON = {Yes | No} Forces generator ON despite requirements of other dispatch modes. Stays ON until this property is set to NO, or an internal algorithm cancels the forced ON state.

Basefreq= Base frequency for which this generator is defined. Default is 60.0.

Like= Name of another Generator object on which to base this one.

ENERGYMETER OBJECT

An EnergyMeter object is an intelligent meter connected to a terminal of a circuit element. It simulates the behavior of an actual energy meter. However, it has more capability because it can access values at other places in the circuit rather than simply at the location at which it is installed. It measures not only power and energy values at its location, but losses and overload values within a defined region of the circuit.

The operation of the object is simple. It has several *registers* that accumulate certain values. At the beginning of a study, the registers are cleared (reset) to zero. At the end of each subsequent solution, the meter is instructed to take a sample. Energy values are then integrated using the interval of time that has passed since the previous solution.

Registers

There are two types of registers:

1. Energy Accumulators (for energy values)
2. Maximum power values ("drag hand" registers).

The energy registers use trapezoidal integration, which allows to use somewhat arbitrary time step sizes between solutions with less integration error. This is important for using load duration curves approximated with straight lines, for example.

The present definitions of the registers are:

1. KWh at the meter location.
2. Kvarh at the meter location.
3. Maximum kW at the meter location.
4. Maximum kVA at the meter location.
5. KWh in the meter zone.
6. Kvarh in the meter zone.
7. Maximum kW in the meter zone.
8. Maximum kVA in the meter zone.
9. Overload kWh in the meter zone, normal ratings.
10. Overload kWh in the meter zone, emergency ratings.
11. Energy Exceeding Normal (EEN) in the loads in the meter zone.
12. Unserved Energy (UE) in the loads in the meter zone.

13. Losses (kWh) in power delivery elements in the meter zone.

14. Reactive losses (kvarh) in power delivery elements in the meter zone.

15. Maximum losses (kW) in power delivery elements in the meter zone.

16. Maximum reactive losses (kvar) in power delivery elements in the meter zone.

Zones

The EnergyMeter object uses the concept of a *zone.* This is an area of the circuit for which the meter is responsible. It can compute energies, losses, etc for any power delivery object and Load object in its zone (Generator objects have their own intrinsic meters).

A zone is a collection of circuit elements "downline" from the meter. This concept is nominally applicable to radial circuits, but also has some applicability to meshed circuits. The zones are automatically determined according to the following rules:

1. Start with the circuit element in which the meter is located. Ignore the terminal on which the meter is connected. This terminal is the start of the zone. Begin tracing with the other terminal(s).

2. Trace out the circuit, finding all other circuit elements (loads and power delivery elements) connected to the zone. Continue tracing out every branch of the circuit. Stop tracing a branch when:

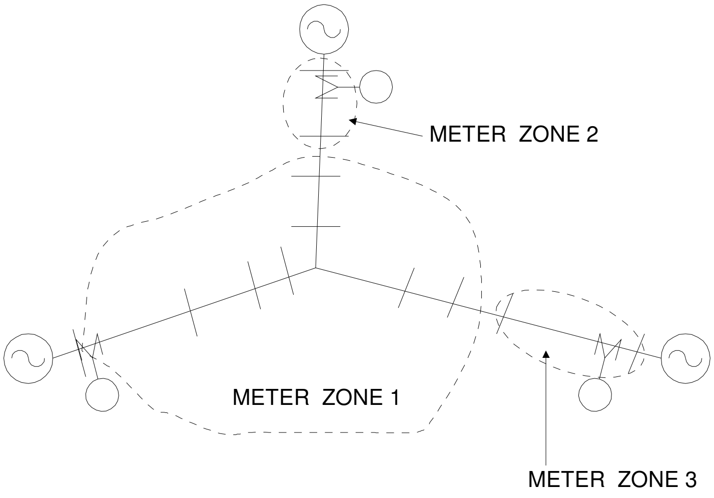
   - The end of the circuit branch is reached
   - A circuit element containing another EnergyMeter object is encountered
   - A OPEN terminal is encountered. (all phases in the terminal are open.)
   - A disabled device is encountered.
   - A circuit element already included in another zone is encountered.
   - There are no more circuit elements to consider.

Zones are automatically updated after a change in the circuit unless the ZONELOCK option (Set command) is set to true (Yes). Then zones remain fixed after initial determination.

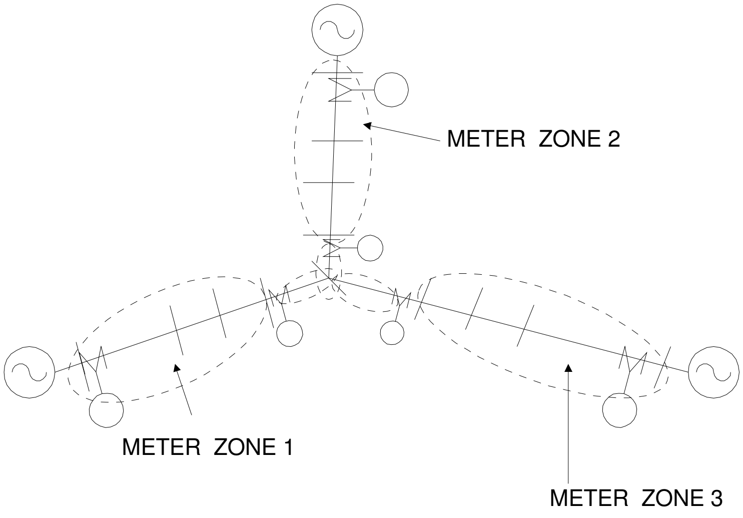
.Zones on Meshed Networks

While the concept of zones nominally applies to radial circuits, judicious placement of energy meters can make the concept useful for meshed networks as well. Keep in mind that there can be many EnergyMeter objects defined in the circuit. Their placement does not necessarily have to represent reality; they are for the reporting of power and energy quantities throughout the system.

The automatic algorithm for determining zones will determine zones consistently for meshed networks, although the zones themselves may not be radial. If there are several meters on the network that could be monitoring the same zone, the first one defined will have access to all the elements except the ones containing the other meters. The others will have only one element in their zone.



Default Zones for a Simple Network



Using additional meters to force the definition of meter zones.
(Note: Don't put any load at tie point or take care in processing meter information so that it isn't counted more than once. All three of the meters added would see the bus in this example.)

Sampling

The sampling algorithms are as follows:

1. Local Energy and Power Values: Simply compute the power into the terminal on which the meter is installed and integrate using the interval between the present solution and the previous solution. This operation uses the voltage and current computed from the present solution.

2. Losses in Zone: Accumulate the kW losses in each power delivery element in the zone.

3. Load in Zone: While sampling the losses in each power delivery element, accumulate the power in all loads connected to the downline bus(es) of the element.

4. Overload Energy in Zone: For each power delivery element in the zone, compute the amount of power exceeding the rating the element compared to both normal and emergency ratings. If an element is overloaded

5. EEN and UE in Zone: For each load in the zone marked as exceeding normal or unserved, compute the present power. Integrate to get energies.

EEN and UE Definitions

EEN refers to load energy considered unserved with because the power exceeds Normal ratings. UE refers to load energy considered unserved because the power exceeds Emergency, or maximum, ratings.

On radial systems (default), a load is marked as unserved with respect to either normal or emergency ratings if either:

1. The voltage at the load bus is below minimum ratings

2. The current in any power delivery element supplying the load exceeds the current ratings.

Either the entire load or just the portion above rating at the bus that is considered unserved is counted as unserved, depending on whether the Excess option or the Total option have been specified.

Properties

The properties, in order, are:

Element= Name of an existing circuit element to which the monitor is to be connected. Note that there may be more than one circuit element with the same name (not wise, but it is allowed). The monitor will be placed at the first one found in the list.

Terminal= No. of the terminal to which the monitor will be connected.

Action= Optional action to execute. One of

1. Clear = reset all registers to zero

2. Save = Saves (appends) the present register values to a file. File name is MTR_*metername*.CSV, where *metername* is the name of the energy meter.

3. Take = Takes a sample at the present solution.

Option = Options: Enter a string ARRAY of any combination of the following. Options processed left-to-right:

(**E**)xcess : (default) UE/EEN is estimate of only energy exceeding capacity

(**T**)otal : UE/EEN is total energy after capacity exceeded.

(**R**)adial : (default) Treats zone as a radial circuit

(**M**)esh : Treats zone as meshed network (not radial).

Example: option=(E, R)

In a meshed network, the overload registers represent the total of the power delivery element overloads and the load UE/EEN registers will contain only those loads that are "unserved", which are those with low voltages. In a radial circuit, the overload registers record the max overload (absolute magnitude, not percent) in the zone. Loads become unserved either with low voltage or if any line in their path to the source is overloaded.

KWNorm = Upper limit on kW load in the zone, *Normal* configuration. Default is 0.0 (ignored). If specified, overrides limits on individual lines for overload EEN. KW above this limit for the entire zone is considered EEN.

KWEmerg = Upper limit on kW load in the zone, *Emergency* configuration. Default is 0.0 (ignored). If specified, overrides limits on individual lines for overload UE. KW above this limit for the entire zone is considered UE.

Peakcurrent = ARRAY of current magnitudes representing the peak currents measured at this location for the load allocation function (for loads defined with **xfkva=**). Default is (400, 400, 400). Enter one current for each phase.

Like= Name of another EnergyMeter object on which to base this one.

MONITOR OBJECT

A monitor is a benign circuit element that is associated with a terminal of another circuit element. It takes a sample when instructed, recording the time and the complex values of voltage and current, or power, at all phases. The data are saved in a file (separate one for each monitor) at the conclusion of a multistep solution or each solution in a Monte Carlo calculation. In essence, it works like a real power monitor. The data in the file may be converted to csv form and, for example, brought into Excel (EPRI provides VBA routines to read the monitor files directly and import either complex voltages and currents or their magnitudes.) The binary form of the monitor file is

Signature (4-byte Integer)	signifies that this is a DSS monitor file = 43756
Version (4-byte integer)	version number of the file
Sample Size (4-byte integer)	No. of quantities saved per sample
Mode (4-byte integer)	Monitor mode

Records follow

```
        <--- All voltages first ---------------->|<--- All currents ----->|
 <hour 1> <sec 1> <V1.re>  <V1.im>  <V2.re>  <V2.im>  .... <I1.re>  <I1.im> ...
 <hour 2> <sec 1> <V1.re>  <V1.im>  <V2.re>  <V2.im>  .... <I1.re>  <I1.im> ...
 <hour 3> <sec 1> <V1.re>  <V1.im>  <V2.re>  <V2.im>  .... <I1.re>  <I1.im> ...
```

If powers are saved then the record has only the power for each phase.

All values are Singles (32-bit). Hours and Seconds values are not included in Sample Size. Recorded values are not necessarily saved as illustrated, depending on Mode (see below). However, the file is always packed singles with each record beginning with the hour and seconds past the hour.

For Monte Carlo runs, the hour is set to the number of the solution and seconds is set to zero.

Monitors may be connected to both power delivery elements and power conversion elements.

Parameters, in order, are:

Element= Name of an existing circuit element to which the monitor is to be connected. Note that there may be more than one circuit element with the same name (not wise, but it is allowed). The monitor will be placed at the first one found in the list.

Terminal= No. of the terminal to which the monitor will be connected.

Mode= Integer bitmask code to describe what it is that the monitor will save. Monitors can save two basic types of quantities: 1) Voltage and current; 2) Power. The Mode codes are defined as follows:

0: Standard mode - V and I,each phase, complex
1: Power each phase, complex (kw and kvars)

+16: Sequence components: V_{012}, I_{012}
+32: Magnitude only
+64: Pos Seq only or Average of phases, if not 3 phases

For example, Mode=33 will save the magnitude of the power (kVA) only in each phase. Mode=112 saves Positive sequence voltages and currents, magnitudes only.

Action= {clear | save} parsing of this parameter forces clearing of the monitor's buffer, or saving to disk.

The monitor file name is constructed from the name of the element and the terminal number as follows: "MON_" + ElementName + TerminalNo + ".MON". For example, a monitor in terminal 1 of element LINEA would create a file called MON_LINEA1.MON.

Example VB Code for Reading A Monitor File

This subroutine loads a monitor file into an Excel spreadsheet. It gets the filename from a spreadsheet cell, opens the file as Binary, and then uses the Get statement to read the file.

```
Public Sub LoadMonitorFile(FileName As String)
    Dim File%
    Dim Signature As Long
    Dim Version As Long
    Dim RecordSize As Long
    Dim SngValue As Single
    Dim Mode As Long

    Dim i%
    Dim iRow As Long


    File% = FreeFile

    Open FileName For Binary As #File%
    Get #File%, , Signature
    Get #File%, , Version
    Get #File%, , RecordSize
    Get #File%, , Mode

    ' based on Mode Put Headings on spreadSheet
    FillInColHeadings Mode, 10, 3
    With ActiveSheet
        iRow = 11

        Do While Not EOF(File%)
            Get #File, , SngValue
            .Cells(iRow, 1) = SngValue
            Get #File, , SngValue
            .Cells(iRow, 2) = SngValue
            For i = 1 To RecordSize
                Get #File, , SngValue
                .Cells(iRow, i + 2) = SngValue
            Next i
            iRow = iRow + 1
        Loop

        If RecordSize < 12 Then  'Clear other cells
            .Range(.Cells(11, RecordSize + 1 + 2), .Cells(iRow - 1, 12 +
2)).ClearContents
        End If

    End With
```

```
        Close #File%

    End Sub
```

Capacitor Object

The capacitor model is basically implemented as a two-terminal power delivery element. However, if you don't specify a connection for the second bus, it will default to the 0 node (ground reference) of the same bus to which the first terminal is connected. That is, it defaults to a grounded wye (star) shunt capacitor bank.

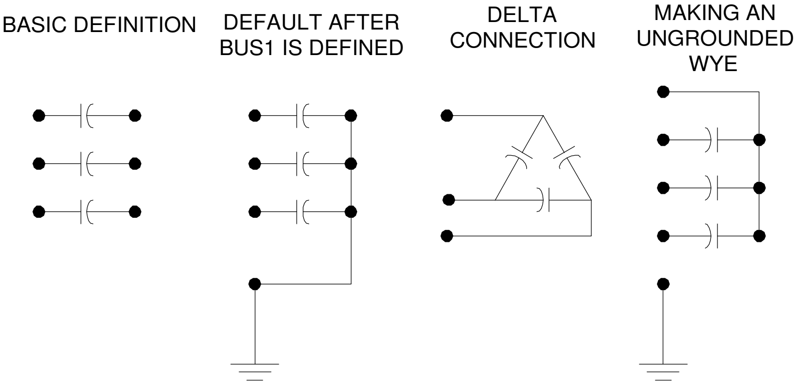


Figure 10: Definition of Capacitor Object

If you specify the connection to be "delta" then the second terminal is eliminated.

If you wish a series capacitor, simply specify a second bus connection.

If you wish an ungrounded wye capacitor, set all the second terminal conductors to an empty node on the first terminal bus, e.g.:

```
Bus1=B1  bus2 = B1.4.4.4    ! for a 3-phase capacitor
```

Of course, any other connection is possibly by explicitly specifying the nodes.

Parameters, in order, are:

Bus1= Definition for the connection of the first bus. When this is set, Bus2 is set to the same bus name, except with all terminals connected to node 0 (ground reference). Set Bus 2 at some time later if you wish a different connection.

Bus2= Bus connection for second terminal. Must always be specified after Bus1 for series capacitor. Not necessary to specify for delta or grd-wye shunt capacitor. Must be specified to achieve an ungrounded neutral point connection.

Phases= Number of phases. Default is 3.

Kvar= Most common of three ways to define a power capacitor. Rated kvar at rated kV, total of all phases. Each phase is assumed equal. Normamps and Emergamps automatically computed. Default is 600.0 kvar.

Kv= Rated kV of the capacitor (not necessarily same as bus rating). For Phases=2 or

Phases=3, enter line-to-line (phase-to-phase) rated voltage. For all other numbers of phases, enter actual can rating. (For Delta connection this is always line-to-line rated voltage). Default is 12.47 kV.

Conn= Connection of bank. One of {wye | ln} for wye connected banks or {delta | ll} for delta (line-line) connected banks. Default is wye (or straight-through for series capacitor).

Cmatrix= Alternate method of defining a capacitor bank. Enter nodal capacitance matrix in µf. Can be used to define either series or shunt banks. Form should be:

Cmatrix=(c_{11} | -c_{21} c_{22} | -c_{31} -c_{32} c_{33})

Cuf= Alternate method of defining a capacitor bank. Enter a value for C in µf. This value is assumed to be the same for a single capacitor is each phase.

Normamps= Normal current rating. Automatically computed if kvar is specified. Otherwise, you need to specify if you wish to use it.

Emergamps= Overload rating. Defaults to 135% of Normamps.

Faultrate= Annual failure rate. Failure events per year. Default is 0.0005.

Pctperm= Percent of faults that are permanent. Default is 100.0.

Basefreq= Base frequency, Hz. Default is 60.0

Like= Name of another Capacitor object on which to base this one.

TRANSFORMER OBJECT

The Transfomer model is implemented as a multi-terminal (two or more) power delivery element.

A transfomer consists of two or more *Windings*, connected in somewhat arbitray fashion (with the standard Wye-Delta defaults, of course). You can specify the parameters of a winding one winding at a time or use arrays to set all the values. Use the "wdg=…" parameter to select a winding.

Transformers have one or more *phases.* The number of conductors per terminal is always one more than the number of phases. For wye- or star-connected windings, the extra conductor is the neutral point. For delta-connected windings, the extra terminal is open internally (you normally leave this connected to node 0).

Parameters, in order, are:

Phases= Number of phases. Default is 3.

Windings= Number of windings. Default is 2.

For defining the winding values one winding at a time, use the following parameters. Always start the winding definition with "wdg = …" when using this method of defining transformer parameters. The remainder of the tags are optional as usual if you keep them in order.

Wdg= Integer representing the winding which will become the active winding for subsequent data.

Bus= Definition for the connection of this winding (each winding is connected to one terminal of the transformer and, hence, to one bus).

Conn= Connection of this winding. One of {wye | ln} for wye connected banks or {delta | ll} for delta (line-line) connected banks. Default is wye.

Kv= Rated voltage of this winding, kV. For transformers designated 2- or 3-phase, enter phase-to-phase kV. For all other designations, enter actual winding kV rating. Two-phase transfomers are assumed to be employed in a 3-phase system. Default is 12.47 kV.

Kva= Base kVA rating (OA rating) of this winding.

Tap = Per unit tap on which this winding is set.

%r = Percent resistance of this winding on the rated kVA base. (Reactance is *between* two windings and is specified separately -- see below.)

rneut = Neutral resistance to ground in ohms for this winding. Ignored if delta winding.

For open ungrounded neutral, set to a **negative** number. Default is –1 (capable of being ungrounded). The DSS defaults to connecting the neutral to node 0 at a bus, so it will still be ground when the system Y is built. To make the neutral floating, explicitly connect it to an unused node at the buse, e.g., Bus=Busname.1.2.3.4, when node 4 will be the explicit neutral node.

xneut = Neutral reactance in ohms for this winding. Ignored if delta winding. Assumed to be in series with neutral resistance. Default is 0.

Use the following parameters to set the winding values using arrays (setting of wdg= … is ignored).

Buses = Array of bus definitions for windings [1, 2. …].

Conns = Array of winding connections for windings [1, 2. …].

KVs = Array of kV ratings following rules stated above for the kV field for windings [1,2,…].

KVAs = Array of base kVA ratings for windings [1,2,…].

Taps = Array of per unit taps for windings [1,2,…].

Use the following parameters to define the reactances of the transformer. For 2- and 3-winding transformers, you may use the conventional XHL, XLT, and XHT parameters. You may also put the values in an array (xscarray), which is required for higher phase order transformers. There are always n*(n-1)/2 different short circuit reactances, where n is the number of windings. *Always use the kVA base of the first winding for entering impedances.* Impedance values are entered in percent.

XHL = Percent reactance high-to-low (winding 1 to winding 2).

XLT = Percent reactance low-to-tertiary (winding 2 to winding 3).

XHT = Percent reactance high-to-tertiary (winding 1 to winding 3).

XscArray = Array of n*(n-1)/2 short circuit reactances in percent on the first winding's kVA base. "n" is number of windings. Order is (12, 13, 14, …1n, 23, 24, … 34, …)

General transformer rating data:

Thermal = Thermal time constant, hrs. Default is 2.

n = Thermal exponent, n, from IEEE/ANSI C57. Default is 0.8.

m = Thermal exponent, m, from IEEE/ANSI C57. Default is 0.8.

flrise = Full-load temperature rise, degrees centigrade. Default is 65.

hsrise = Hot-spot temperatire rise, degrees centigrade. Default is 15.

Loadloss = Losses at full load, kW.

Noloadloss = No load losses, kW

NormHKVA = Normal maximum kVA rating for H winding (1). Usually 100 - 110% of maximum nameplate rating.

EmergHKVA = Emergency maximum kVA rating for H winding (1). Usually 140 - 150% of maximum nameplate rating. This is the amount of loading that will cause 1% loss of life in one day.

Faultrate = Failure rate for transformer. Defaults to 0.007 per year. All are considered permanent.

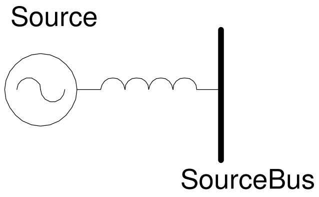
Basefreq= Base frequency, Hz. Default is 60.0

Like= Name of another Transformer object on which to base this one.

Sub = Yes/No. Designates whether this transformer is to be treated as a substation. Default is No.

Default Circuit

When a new circuit is instantiated, it is created as a 3-phase voltage source named "Source" connected to a bus named "SourceBus" with a reasonable short circuit strength for transmission systems feeding distribution substations.



The default values are (see Vsource object definition):

115 kV

3000 MVA short circuit

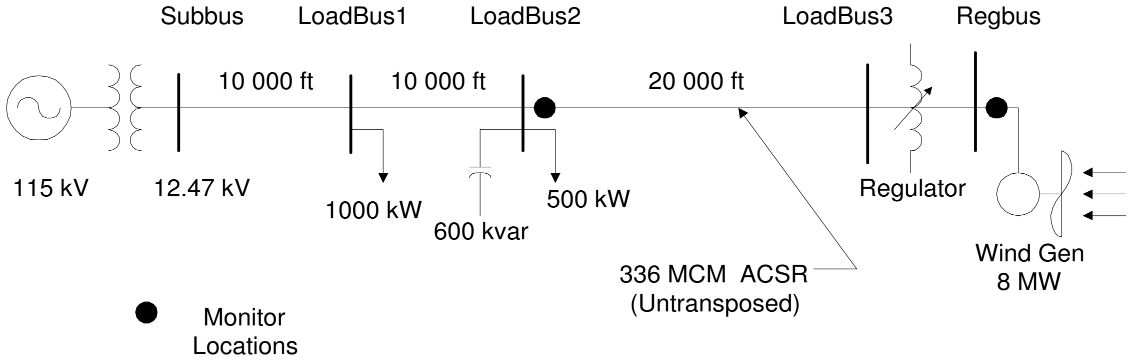
Thus, the circuit can be immediately solved, albeit with a trivial result.

The circuit is also immediately available for adding a substation and/or lines for a quick manual circuit modeling task.

The default circuit model does not include a substation transformer because the user may wish to study more than one substation connected by a non-trivial transmission or subtransmission network.

Examples

Example Circuit 1

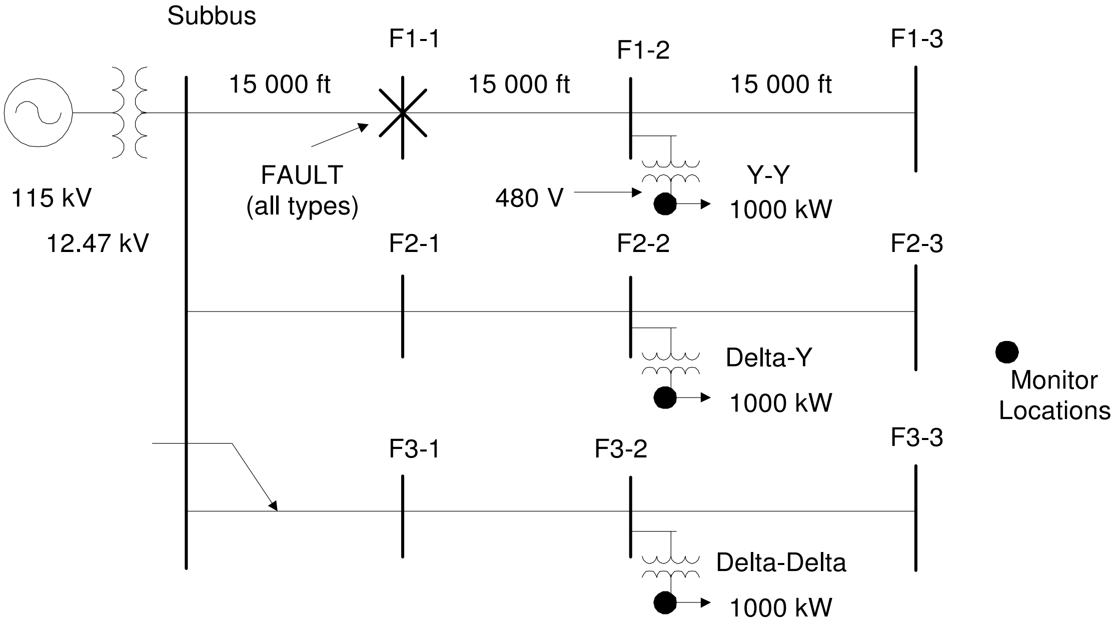


DSS Circuit Description Script

```
new object=circuit.DSSLLibtestckt
~ basekv=115  1.00 0.0 60.0 3 20000 21000 4.0 3.0  !edit the voltage source

new loadshape.day 24 1.0
~   mult=(.3 .3 .3 .35 .36 .39 .41 .48 .52 .59 .62 .94 .87 .91 .95 .95 1.0 .98 .94
.92 .61 .60 .51 .44)
new loadshape.year 24 1.0  ! same as day for now
~   mult=".3 .3 .3 .35 .36 .39 .41 .48 .52 .59 .62 .94 .87 .91 .95 .95 1.0 .98 .94
.92 .61 .60 .51 .44"
new loadshape.wind 2400 0.00027777   ! unit must be hours 1.0/3600.0 = .0002777
~   csvfile=zavwind.csv action=normalize  ! wind turbine characteristi

! define a linecode for the lines - unbalanced 336 MCM ACSR connection
new linecode.336matrix nphases=3   ! horizontal flat construction
~  rmatrix=(0.0868455 |  0.0298305 0.0887966 | 0.0288883 0.0298305  0.0868455) !
ohms per 1000 ft
~  xmatrix=(0.2025449 |  0.0847210 0.1961452 | 0.0719161 0.0847210  0.2025449)
~  cmatrix=(2.74 | -0.70 2.96| -0.34 -0.71 2.74)  !nf per 1000 ft
~ Normamps = 400  Emergamps=600

! Substation transformer
new transformer.sub phases=3 windings=2 buses=(SourceBus subbus) conns='delta wye'
kvs="115 12.47 " kvas="20000 20000" XHL=7

! define the lines
new line.line1 subbus   loadbus1 linecode=336matrix length=10
new line.line2 loadbus1 loadbus2 336matrix 10
new line.line3 Loadbus2 loadbus3 336matrix 20

! define a couple of loads
new load.load1 bus1=loadbus1 phases=3 kv=12.47 kw=1000.0 pf=0.88 model=1 class=1
yearly=year daily=day status=fixed
new load.load2 bus1=loadbus2 phases=3 kv=12.47 kw=500.0 pf=0.88 model=1 class=1
yearly=year daily=day conn=delta status=fixed

! Capacitor with control
new capacitor.C1  bus1=loadbus2  phases=3 kvar=600 kv=12.47
new capcontrol.C1 element=line.line3 1 capacitor=C1 type=current ctratio=1
ONsetting=60 OFFsetting=55 delay=2
```

```
! regulated transformer to DG bus
new transformer.reg1 phases=3 windings=2
~           buses=(loadbus3 regbus)
~           conns='wye wye'
~           kvs="12.47 12.47"
~           kvas="8000 8000"
~           XHL=1              !tiny reactance for a regulator

! Regulator Control definitions
new regcontrol.sub  transformer=sub  winding=2 vreg=125 band=3 ptratio=60 delay=10
new regcontrol.reg1 transformer=reg1 winding=2 vreg=122 band=3 ptratio=60 delay=15

! define a wind generator of 8MW
New generator.gen1   bus1=regbus kV=12.47 kW=8000 pf=1 conn=delta duty=wind
Model=1


! Define some monitors so's we can see what's happenin'

New Monitor.gen1a element=generator.gen1 1   mode=48
New Monitor.line3 element=line.line3 1      mode=48
New Monitor.gen1  element=generator.gen1 1 mode=32

! Define voltage bases so voltage reports come out in per unit
Set voltagebases="115 12.47 .48"
Calcv

Set controlmode=time
Set mode=duty number=2400  hour=0  h=1.0 sec=0  ! Mode resets the monitors
```

EXAMPLE CIRCUIT 2



DSS Circuit Description Script

```
new object=circuit.DSSLLibtestckt
~ basekv=115  1.00 0.0 60.0 3 20000 21000 4.0 3.0  !edit the voltage source

! define a linecode for the lines - unbalanced 336 MCM ACSR connection
new linecode.336matrix nphases=3   ! horizontal flat construction
~  rmatrix=(0.0868455 |  0.0298305 0.0887966 | 0.0288883 0.0298305  0.0868455) !
ohms per 1000 ft
~  xmatrix=(0.2025449 |  0.0847210 0.1961452 | 0.0719161 0.0847210  0.2025449)
~  cmatrix=(2.74 | -0.70 2.96| -0.34 -0.71 2.74)  !nf per 1000 ft
~  Normamps = 400  Emergamps=600

! Substation transformer
new transformer.sub phases=3 windings=2 buses=(SourceBus subbus) conns='delta wye'
kvs="115 12.47 " kvas="20000 20000" XHL=7

! define the lines  (Make sure they have unique names!)

! Feeder 1
new line.line1-1 subbus   F1-1 336matrix 15
new line.line1-2 F1-1     F1-2 336matrix 15
new line.line1-3 F1-2     F1-3 336matrix 15

! Feeder 2
new line.line2-1 subbus   F2-1 336matrix 15
new line.line2-2 F2-1     F2-2 336matrix 15
new line.line2-3 F2-2     F2-3 336matrix 15

! Feeder 3
new line.line3-1 subbus   F3-1 336matrix 15
new line.line3-2 F3-1     F3-2 336matrix 15
new line.line3-3 F3-2     F3-3 336matrix 15
```

```
! Define 3 transformer of different connections on each feeder midpoint
new transformer.TR1 phases=3 windings=2
~           buses=(F1-2 Genbus1)
~           conns='wye wye'
~           kvs="12.47 0.48"
~           kvas="5000 5000"
~           XHL=6

new transformer.TR2 phases=3 windings=2
~           buses=(F2-2 Genbus2)
~           conns='delta wye'
~           kvs="12.47 0.48"
~           kvas="5000 5000"
~           XHL=6

new transformer.TR3 phases=3 windings=2
~           buses=(F3-2 Genbus3)
~           conns='delta delta'
~           kvs="12.47 0.48"
~           kvas="5000 5000"
~           XHL=6

! put some loads on the transformers
new load.load1 bus1=Genbus1 phases=3 kv=0.48 kw=1000.0 pf=0.88 model=1 class=1
status=fixed
new load.load2 bus1=Genbus2 phases=3 kv=0.48 kw=1000.0 pf=0.88 model=1 class=1
status=fixed
new load.load3 bus1=Genbus3 phases=3 kv=0.48 kw=1000.0 pf=0.88 model=1 class=1
status=fixed


! Define monitors at the secondary buses to pick up voltage magnitudes

New Monitor.TR1 element=Transformer.TR1 2  mode=32
New Monitor.TR2 element=Transformer.TR2 2  mode=32
New Monitor.TR3 element=Transformer.TR3 2  mode=32


! Define voltage bases so voltage reports come out in per unit
Set voltagebases="115 12.47 .48"
Calcv

! define all kinds of faults at one bus (will be moved later)
! in Monte Carlo Fault mode, only one will be chosen at a time
New Fault.F1   bus1=F1-1.1 phases=1 r=2
New Fault.F2   bus1=F1-1.2 phases=1 r=2
New Fault.F3   bus1=F1-1.3 phases=1 r=2
New Fault.FALL bus1=F1-1   phases=3 r=2
New Fault.F12  bus1=F1-1.1 bus2=F1-1.2 phases=1 r=2
New Fault.F23  bus1=F1-1.2 Bus2=F1-1.3 phases=1 r=2

Set loadmodel=a
Set Toler=0.001 Random=uniform
```

DSS Command Reference

Object = EXECUTIVE
Property Description

(1) new Create a new object within the DSS. Object becomes the active object Example: New Line.line1 ...

(2) edit Edit an object. The object is selected and it then becomes the active object.

Note that Edit is the default command. You many change a property value simply by giving the full property name and the new value, for example:

line.line1.r1=.04
vsource.source.kvll=230

(3) more Continuation of editing on the active object.

(4) m Continuation of editing on the active object. An abbreviation for More

(5) ~ Continuation of editing on the active object. An abbreviation.

Example:
New Line.Line1 Bus1=aaa bus2=bbb
~ R1=.058
~ X1=.1121

(6) select Selects an element and makes it the active element. You can also specify the active terminal (default = 1).

Syntax:
Select [element=]elementname [terminal=]terminalnumber

Example:
Select Line.Line1
~ R1=.1
(continue editing)

Select Line.Line1 2
Voltages (returns voltages at terminal 2 in Result)

(7) save Saves the present values in both monitors and energy meters in the active circuit.

(8) show Writes selected results to a text file and brings up the editor (see Set Editor=....) with the file for you to browse.

Valid Options:
Show Buses
Show Currents
Show COnvergence (convergence report)
Show Elements [Classname] (shows names of all elements in circuit or all elements of a class)
Show Faults (after Fault Study)

Show Generators
Show MEters
Show Monitor Monitorname
Show PAnel (control panel)
Show Powers [MVA]
Show Voltages (default)
Show Zone EnergyMeterName
Show AutoAdded (see AutoAdd solution mode)
Show Taps (regulated transformers)
Show Overloads (overloaded PD elements)

You may abbreviate with V, C, PA, P, M, and Me, etc.

(9) solve Perform the solution of the present solution mode. You can set any option that you can set with the Set command (see Set). The Solve command is virtually synonymous with the Set command except that a solution is performed after the options are processed.

(10) enable Enables a circuit element. Example: Enable load.loadxxx

(11) disable Disables a circuit element. Example: Disable load.loadxxx

The item remains defined, but is not included in the solution.

(12) plot Not yet implemented. Use monitors to bring data to Excel.

(13) reset {Monitors | Meters | (no argument) } Resets all Monitors or Energymeters as specified. If no argument specified, resets meters and monitors.

(14) compile Reads the designated file name containing DSS commands and processes them as IF they were entered directly into the command line. The file is said to be "compiled"

Syntax:
Compile filename

(15) set Used to set various DSS modes and options.

(1) type Sets the active DSS class type. Same as Class=...

(2) element Sets the active DSS element by name. You can use the complete object spec (class.name) or just the name. if full name is specifed, class becomes the active class, also.

(3) hour Sets the hour used for the start time of the solution.

(4) sec Sets the seconds from the hour for the start time of the solution.

(5) year Sets the Year (integer number) to be used for the solution. for certain solution types, this determines the growth multiplier.

(6) freq Sets the frequency for the solution of the active circuit.

(7) stepsize Sets the time step in sec for the active circuit. Nominally for dynamics solution.

(8) mode Set the solution Mode: One of
Snapshot,
DUtycycle,
DIrect,
Daily,
M1 (Monte Carlo 1),
M2 (Monte Carlo 2),

M3 (Monte Carlo 3),
Faultstudy,
Yearly (follow Yearly curve),
MF (monte carlo fault study)
Peakday,
LD1 (load-duration 1)
LD2 (load-duration 2)
AutoAdd (see AddType)

Side effect: setting the Mode propergy resets all monitors and energy meters. It also resets the time step, etc. to defaults for each mode. After the initial reset, the user must explicitly reset the monitors and/or meters until another Set Mode= command.

(9) random =uniform or =gaussian for Monte Carlo VARiables.

(10) number Number of solutions to perform for monte carlo or dutycycle solutions.

(11) time Specify the solution start time as an array: time=(hour, secs)

(12) class Synonym for Type=. (See above)

(13) object Synonym for Element=. (See above)

(14) circuit Set the active circuit by name.

(15) editor Set the command string required to start up the editor preferred by the user.

(16) tolerance Sets the solution tolerance. Default is 0.0001.

(17) maxiter Sets the maximum allowable iterations for power flow solutions. Default is 15.

(18) h Alternate name for time step size.

(19) loadmodel {Powerflow | Admittance} depending on the type of solution you wish to perform. If admittance, a non-iterative, direct solution is done with all loads and generators modeled by their equivalent admittance.

(20) loadmult Global load multiplier for this circuit. Does not affect loads designated to be "fixed". All other base kW values are multiplied by this number. Defaults to 1.0 when the circuit is created. As with other values, it always stays at the last value to which it was set until changed again.

(21) normvminpu Minimum permissible per unit voltage for normal conditions. Default is 0.95.

(22) normvmaxpu Maximum permissible per unit voltage for normal conditions. Default is 1.05.

(23) emergvminpu Minimum permissible per unit voltage for emergency (contingency) conditions. Default is 0.90.

(24) emergvmaxpu Maximum permissible per unit voltage for emergency (contingency) conditions. Default is 1.08.

(25) %mean Percent mean to use for global load multiplier. Default is 65%.

(26) %stddev Percent Standard deviation to use for global load multiplier. Default is 9%.

(27) LDCurve Set Load-Duration Curve. Global load multiplier is defined by this curve for LD1 and LD2 solution modes. Default is Nil.

(28) %growth Set default annual growth rate, percent, for loads with no growth curve specified. Default is 2.5.

(29) genkw Size of generator, kW, to automatically add to system. Default is 1000.0

(30) genpf Power factor of generator to assume for automatic addition. Default is

1.0.

(31) capkVAR Size of capacitor, kVAR, to automatically add to system. Default is 600.0.

(32) addtype{Generator | Capacitor} Default is Generator. Type of device for AutoAdd Mode.

(33) allowduplicates{YES/TRUE | NO/FALSE} Default is No. Flag to indicate if it is OK to have devices of same name in the same class. If No, then a New command is treated as an Edit command. If Yes, then a New command will always result in a device being added.

(34) zonelock {YES/TRUE | NO/FALSE} Default is No. if No, then meter zones are recomputed each time there is a change in the circuit. If Yes, then meter zones are not recomputed unless they have not yet been computed. Meter zones are normally recomputed on Solve command following a circuit change.

(35) ueweight Weighting factor for UE/EEN in AutoAdd functions. Defaults to 1.0.

Autoadd mode minimizes

(Lossweight * Losses + UEweight * UE).

If you wish to ignore UE, set to 0. This applies only when there are EnergyMeter objects. Otherwise, AutoAdd mode minimizes total system losses.

(36) lossweight Weighting factor for Losses in AutoAdd functions. Defaults to 1.0.

Autoadd mode minimizes

(Lossweight * Losses + UEweight * UE).

If you wish to ignore Losses, set to 0. This applies only when there are EnergyMeter objects. Otherwise, AutoAdd mode minimizes total system losses.

(37) ueregs Which EnergyMeter register(s) to use for UE in AutoAdd Mode. May be one or more registers. if more than one, register values are summed together. Array of integer values > 0. Defaults to 11 (for Load EEN).

for a list of EnergyMeter register numbers, do the "Show Meters" command after defining a circuit.

(38) lossregs Which EnergyMeter register(s) to use for Losses in AutoAdd Mode. May be one or more registers. if more than one, register values are summed together. Array of integer values > 0. Defaults to 13 (for Zone kWh Losses).

for a list of EnergyMeter register numbers, do the "Show Meters" command after defining a circuit.

(39) voltagebases Define legal voltage bases for this circuit. Enter an array of the legal voltage bases, for example:

set voltagebases=".208, .480, 12.47, 24.9, 34.5, 115.0, 230.0"

When the CalcVoltageBases command is issued, a snapshot solution is performed with no load injections and the bus base voltage is set to the nearest legal voltage base. The defaults are as shown in the example above.

(40) algorithm {Normal | Newton} Solution algorithm type. Normal is a fixed point iteration that is a little quicker than the Newton iteration. Normal is adequate for most radial distribution circuits. Newton is more robust for circuits that are difficult to solve.

(41) trapezoidal {YES/TRUE | NO/FALSE} Default is "No". Specifies whether to use trapezoidal integration for accumulating energy meter registers. Applies to EnergyMeter and Generator objects. Default method simply multiplies the present value of the registers times the width of the interval. Trapezoidal is more accurate when there are sharp changes in a load shape or unequal intervals. Trapezoidal is automatically used for some load-duration curve simulations where the interval size varies considerably. Keep in mind that for Trapezoidal, you have to solve one more point than the number of intervals. That is, to do a Daily simulation on a 24-hr load shape, you would set Number=25 to force a solution at the first point again to establish the last (24th) interval.

(42) autobuslist Array of bus names to include in AutoAdd searches. Or, you can specify a text file holding the names, one to a line, by using the syntax (file=filename) instead of the actual array elements. Default is null, which results in the program using either the buses in the EnergyMeter object zones or, if no EnergyMeters, all the buses, which can make for lengthy solution times.

Examples:

Set autobuslist=(bus1, bus2, bus3, ...)
Set autobuslist=(file=buslist.txt)

(43) controlmode {STATIC |EVENT | TIME} Default is "STATIC". Control mode for the solution.
STATIC = Time does not advance. Control actions are executed in order of shortest time to act until all actions are cleared from the control queue. Use this mode for power flow solutions which may require several regulator tap changes per solution.

EVENT = solution is event driven. Only the control actions nearest in time are executed and the time is advanced automatically to the time of the event.

TIME = solution is time driven. Control actions are executed when the time for the pending action is reached or surpassed.

Controls may reset and may choose not to act when it comes their time. Use TIME mode when modeling a control externally to the DSS and a solution mode such as DAILY or DUTYCYCLE that advances time, or set the time (hour and sec) explicitly from the external program.

(44) tracecontrol {YES/TRUE | NO/FALSE} Set to YES to trace the actions taken

in the control queue. Creates a file named TRACE_CONTROLQUEUE.CSV in the default directory. The names of all circuit elements taking an action are logged.

(45) genmultGlobal multiplier for the kW output of every generator in the circuit. Default is 1.0. Applies to Snapshot, Daily, and DutyCycle solution modes. Ignored if generator is designated as Status=Fixed.

(46) defaultdaily Default daily load shape name. Default value is "default", which is a 24-hour curve defined when the DSS is started.

(47) defaultyearly Default yearly load shape name. Default value is "default", which is a 24-hour curve defined when the DSS is started.

(48) allocationfactors Sets all allocation factors for all loads in the active circuit to the value given.

(49) cktmodel {Multiphase | Positive} Default = Multiphase. Designates whether circuit model is to interpreted as a normal multi-phase model or a positive-sequence only model

(50) pricesignal Sets the price signal ($/MWh) for the circuit. Initial value is 25.

(51) pricecurve Sets the curve to use to obtain for price signal. Default is none (null string). If none, price signal either remains constant or is set by an external process. Curve is defined as a loadshape (not normalized) and should correspond to the type of analysis being performed (daily, yearly, load-duration, etc.).

(16) dump Display the properties of either a specific DSS object or a complete dump on all variables in the problem (Warning! Could be very large!). Brings up the default text editor with the text file written by this command.
Syntax: dump [class.obj] [debug]
Examples:

Dump line.line1
Dump transformer.* (dumps all transformers)
Dump (dumps all objects in circuit)

(17) open Opens the specified terminal and conductor of the specified circuit element. If the conductor is not specified, all phase conductors of the terminal are opened.

Examples:
Open line.line1 2
(opens all phases of terminal 2)

Open line.line1 2 3
(opens the 3rd conductor of terminal 2)

(18) close Opposite of the Open command.

(19) // Comment. Command line is ignored.

(20) redirect Synonym for Compile.

(21) help Gives this display.

(22) quit Does nothing at present.

(23) ? Inquiry for property value. Displays a small message box with the value. Specify the full property name.

Example: ? Line.Line1.R1

Note you can set this property merely by saying: Line.line1.r1=.058

(24) next {Year | Hour | t} Increments year, hour, or time as specified. If "t" is specified, then increments time by current step size.

(25) panel Displays main control panel window.

(26) sample Force all monitors and meters to take a sample now.

(27) clear Clear all circuits currently in memory.

(28) about Display "About Box". (Result string set to Version string.)

(29) calcvoltagebasesCalculates voltagebase for buses based on voltage bases defined with Set voltagebases=... command.

(30) SetkVBase Command to explicitly set the base voltage for a bus. Bus must be previously defined. Parameters in order are:
Bus = {bus name}
kVLL = (line-to-line base kV)
kVLN = (line-to-neutral base kV)

kV base is normally given in line-to-line kV (phase-phase). However, it may also be specified by line-to-neutral kV.
The following exampes are equivalent:

setkvbase Bus=B9654 kVLL=13.2
setkvbase B9654 13.2
setkvbase B9654 kvln=7.62

(31) BuildY Forces rebuild of Y matrix upon next Solve command regardless of need. The usual reason for doing this would be to reset the matrix for another load level when using LoadModel=PowerFlow (the default) when the system is difficult to solve when the load is far from its base value. Works by invalidating the Y primitive matrices for all the Power Conversion elements.

(32) get Returns DSS property values set using the Set command. Result is returne in Result property of the Text interface.

VBA Example:

DSSText.Command = "Get mode"
Answer = DSSText.Result

Multiple properties may be requested on one get. The results are appended and the individual values separated by commas.

See help on Set command for property names.

(33) init This command forces reinitialization of the solution for the next Solve command. To minimize iterations, most solutions start with the previous solution unless there has been a circuit change. However, if the previous solution is bad, it may be necessary to re-initialize. In most cases, a re-initiallization results in a zero-load power flow solution with only the series power delivery elements considered.

(34) export Export various solution values to CSV files for import into other programs. Creates a new CSV file except for Energymeter and Generator objects, for which the results for each device of this class are APPENDED to the

CSV File. You may export to a specific file by specifying the file name as the LAST parameter on the line. Otherwise, the default file names shown below are used. For Energymeter and Generator, specifying the switch "/multiple" (or /m) for the file name will cause a separate file to be written for each meter or generator. The default is for a single file containing all elements.

Syntax for Implemented Exports:

```
Export Voltages [Filename] (EXP_VOLTAGES.CSV)
Export SeqVoltages [Filename] (EXP_SEQVOLTAGES.CSV)
Export Currents [Filename] (EXP_CURRENTS.CSV)
Export Overloads [Filename] EXP_OVERLOADS.CSV)
Export SeqCurrents [Filename] (EXP_SEQCURRENTS.CSV)
Export Powers [MVA] [Filename](EXP_POWERS.CSV)
Export Faultstudy [Filename] (EXP_FAULTS.CSV)
Export Generators [Filename | /m ] (EXP_GENMETERS.CSV)
Export Loads [Filename] (EXP_LOADS.CSV)
Export Meters [Filename | /m ] (EXP_METERS.CSV)
Export Monitors monitorname (file name is assigned)
```

May be abreviated Export V, Export C, etc. Default is "V".

(35) fileedit Edit specified file in default text file editor (see Set Editor= option).

Fileedit EXP_METERS.CSV (brings up the meters export file)

"FileEdit" may be abbreviated to a unique character string.

(36) voltages Returns the voltages for the ACTIVE TERMINAL ONLY of the active circuit element in the Result string. For setting the active circuit element, see the Select command. Returned as magnitude and angle quantities, comma separated, one set per conductor of the terminal.

(37) currents Returns the currents for each conductor of ALL terminals of the active circuit element in the Result string. (See Select command.)Returned as comma-separated magnitude and angle.

(38) powers Returns the powers (complex) going into each conductors of ALL terminals of the active circuit element in the Result string. (See Select command.)Returned as comma-separated kW and kvar.

(39) seqvoltages Returns the sequence voltages at all terminals of the active circuit element (see Select command) in Result string. Returned as comma-separated magnitude and angle pairs.Order of returned values: 0, 1, 2 (for each terminal).

(40) seqcurrents Returns the sequence currents into all terminals of the active circuit element (see Select command) in Result string. Returned as comma-separated magnitude and angle pairs.Order of returned values: 0, 1, 2 (for each terminal).

(41) seqpowers Returns the sequence powers into all terminals of the active circuit element (see Select command) in Result string. Returned as comma-separated magnitude and angle pairs.Order of returned values: 0, 1, 2 (for each terminal).

(42) losses Returns the total losses for the active circuit element (see Select

command) in the Result string in kW, kvar.

(43) phaselosses Returns the losses for the active circuit element (see Select command) for each PHASE in the Result string in comma-separated kW, kvar pairs.

(44) cktlosses Returns the total losses for the active circuit in the Result string in kW, kvar.

(45) allocateloads Estimates the allocation factors for loads that are defined using the XFKVA property. Requires that energymeter objects be defined with the PEAKCURRENT property set. Loads that are not in the zone of an energymeter cannot be allocated.

Object = SOLUTION

Property Description

(1) ------- Use Set Command to set Solution properties.

(2) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = LINECODE

Property Description

(1) nphases Number of phases in the line this line code data represents.

(2) r1 Positive-sequence Resistance, ohms per unit length. See also Rmatrix.

(3) x1 Positive-sequence Reactance, ohms per unit length. See also Xmatrix

(4) r0 Zero-sequence Resistance, ohms per unit length.

(5) x0 Zero-sequence Reactance, ohms per unit length.

(6) c1 Positive-sequence capacitance, nf per unit length. See also Cmatrix.

(7) c0 Zero-sequence capacitance, nf per unit length.

(8) units One of (ohms per ...) {mi|km|kft|m|me}. Default is none; assumes units agree with length unitsgiven in Line object

(9) rmatrix Resistance matrix, lower triangle, ohms per unit length. Order of the matrix is the number of phases. May be used to specify the impedance of any line configuration. For balanced line models, you may use the standard symmetrical component data definition instead.

(10) xmatrix Reactance matrix, lower triangle, ohms per unit length. Order of the matrix is the number of phases. May be used to specify the impedance of any line configuration. For balanced line models, you may use the standard symmetrical component data definition instead.

(11) cmatrix Nodal Capacitance matrix, lower triangle, nf per unit length.Order of the matrix is the number of phases. May be used to specify the shunt capacitance of any line configuration. For balanced line models, you may use the standard symmetrical component data definition instead.

(12) baseFreq Frequency at which impedances are specified.

(13) normamps Normal ampere limit on line. This is the so-called Planning Limit. It may also be the value above which load will have to be dropped in a contingency. Usually about 75% - 80% of the emergency (one-hour) rating.

(14) emergamps Emergency ampere limit on line (usually one-hour rating).

(15) faultrate Number of faults per unit length per year.

(16) pctperm Percentage of the faults that become permanent.

(17) repair Hours to repair.

(18) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = LOADSHAPE
Property Description
(1) npts Number of points to expect in subsequent vector.
(2) interval Time interval for fixed interval data. (hrs) Default = 1
(3) mult Vector of multiplier values
(4) hour Vector of hour values. Only necessary to define for variable interval data. If the data are fixed interval, do not use this property.
(5) mean Mean of the multipliers. Automatically computed when a curve is defined.However, you may set it independently. Used for Monte Carlo load simulations.
(6) stddev Standard deviation of multipliers. This is automatically computed when a vector or file of multipliers is entered. However, you may set it to another value indepently.Is overwritten if you subsequently read in a curve

Used for Monte Carlo load simulations.
(7) csvfile Switch input of load curve data to a csv file containing (year, mult) points, or simply (mult) values for fixed time interval data, one per line.
(8) sngfile Switch input of load curve data to a binary file of singles containing (year, mult) points, or simply (mult) values for fixed time interval data, packed one after another.
(9) dblfile Switch input of load curve data to a binary file of doubles containing (year, mult) points, or simply (mult) values for fixed time interval data, packed one after another.
(10) action NORMALIZE is only defined action. After defining load curve data, setting action=normalize will modify the multipliers so that the peak is 1.0. The mean and std deviation are recomputed.
(11) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = GROWTHSHAPE
Property Description
(1) npts Number of points to expect in subsequent vector.
(2) year Array of year values, or a text file spec, corresponding to the multipliers. Enter only those years where the growth changes. May be any integer sequence -- just so it is consistent. See help on Mult.
(3) mult Array of growth multiplier values, or a text file spec, corresponding to the year values. Enter the multiplier by which you would multiply the previous year's load to get the present year's.

Examples:

Year = "1, 2, 5" Mult="1.05, 1.025, 1.02".
Year= (File=years.txt) Mult= (file=mults.txt).

Text files contain one value per line.

(4) csvfile Switch input of growth curve data to a csv file containing (year, mult) points, one per line.
(5) sngfile Switch input of growth curve data to a binary file of singles containing (year, mult) points, packed one after another.
(6) dblfile Switch input of growth curve data to a binary file of doubles containing (year, mult) points, packed one after another.
(7) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = TCC_CURVE
Property Description
(1) npts Number of points to expect in time-current arrays.
(2) C_array Array of current (or voltage) values corresponding to time values (see help on T_Array).
(3) T_array Array of time values in sec. Typical array syntax: t_array = (1, 2, 3, 4, ...)

Can also substitute a file designation: t_array = (file=filename)

The specified file has one value per line.
(4) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = LINE
Property Description
(1) bus1 Name of bus to which first terminal is connected. Example:
bus1=busname (assumes all terminals connected in normal phase order)
bus1=busname.3.1.2.0 (specify terminal to node connections explicitly)
(2) bus2 Name of bus to which 2nd terminal is connected.
(3) linecode Name of linecode object describing line impedances. If you use a line code, you do not need to specify the impedances here. The line code must have been PREVIOUSLY defined. The values specified last will prevail over those specified earlier (left-to-right sequence of properties). If no line code or impedance data are specified, line object defaults to 336 MCM ACSR on 4 ft spacing.
(4) length Length of line, in units compatible with impedance data. Default is 1.0.
(5) phases Number of phases, this line.
(6) r1 Positive-sequence Resistance, ohms per unit length. See also Rmatrix.
(7) x1 Positive-sequence Reactance, ohms per unit length. See also Xmatrix
(8) r0 Zero-sequence Resistance, ohms per unit length.
(9) x0 Zero-sequence Reactance, ohms per unit length.
(10) c1 Positive-sequence capacitance, nf per unit length. See also Cmatrix.
(11) c0 Zero-sequence capacitance, nf per unit length.
(12) rmatrix Resistance matrix, lower triangle, ohms per unit length. Order of the matrix is the number of phases. May be used to specify the impedance of

any line configuration. For balanced line models, you may use the standard symmetrical component data definition instead.

(13) xmatrix Reactance matrix, lower triangle, ohms per unit length. Order of the matrix is the number of phases. May be used to specify the impedance of any line configuration. For balanced line models, you may use the standard symmetrical component data definition instead.

(14) cmatrix Nodal Capacitance matrix, lower triangle, nf per unit length.Order of the matrix is the number of phases. May be used to specify the shunt capacitance of any line configuration. For balanced line models, you may use the standard symmetrical component data definition instead.

(15) normamps Normal rated current.

(16) emergamps Maximum current.

(17) faultrate No. of failures per year.

(18) pctperm Percent of failures that become permanent.

(19) repair Hours to repair.

(20) basefreq Base Frequency for ratings.

(21) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = VSOURCE

Property Description

(1) bus1 Name of bus to which source is connected.
bus1=busname
bus1=busname.1.2.3

(2) basekv Base Source kV, usually L-L unless you are making a positive-sequence modelin which case, it will be L-N.

(3) pu Per unit of the base voltage that the source is actually operating at. "pu=1.05"

(4) angle Phase angle in degrees of first phase: e.g.,Angle=10.3

(5) frequency Source frequency. Defaults to 60 Hz.

(6) phases Number of phases. Defaults to 3.

(7) MVAsc3 MVA Short circuit, 3-phase fault. Default = 2000. Z1 is determined by squaring the base kv and dividing by this value. For single-phase source, this value is not used.

(8) MVAsc1 MVA Short Circuit, 1-phase fault. Default = 2100. The "single-phase impedance", Zs, is determined by squaring the base kV and dividing by this value. Then Z0 is determined by Z0 = 3Zs - 2Z1. For 1-phase sources, Zs is used directly. Use X0R0 to define X/R ratio for 1-phase source.

(9) x1r1 Positive-sequence X/R ratio. Default = 4.

(10) x0r0 Zero-sequence X/R ratio.Default = 3.

(11) Isc3 Alternate method of defining the source impedance.
3-phase short circuit current, amps. Default is 10000.

(12) Isc1 Alternate method of defining the source impedance.
single-phase short circuit current, amps. Default is 10500.

(13) R1 Alternate method of defining the source impedance.
Positive-sequence resistance, ohms. Default is 1.65.

(14) X1 Alternate method of defining the source impedance.
Positive-sequence reactance, ohms. Default is 6.6.

(15) R0 Alternate method of defining the source impedance. Zero-sequence resistance, ohms. Default is 1.9.

(16) X0 Alternate method of defining the source impedance. Zero-sequence reactance, ohms. Default is 5.7.

(17) basefreq Base Frequency for ratings.

(18) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = LOAD

Property Description

(1) phases Number of Phases, this load. Load is evenly divided among phases.

(2) bus1 Bus to which the load is connected. May include specific node specification.

(3) kv Nominal rated (1.0 per unit) voltage, kV, for load. for 2- and 3-phase loads, specify phase-phase kV. Otherwise, specify actual kV across each branch of the load. If wye (star), specify phase-neutral kV. If delta or phase-phase connected, specify phase-phase kV.

(4) kw Total base kW for the load. Normally, you would enter the maximum kW for the load for the first year and allow it to be adjusted by the load shapes, growth shapes, and global load multiplier.

(5) pf Load power factor. Enter negative for leading powerfactor (when kW and kvar have opposite signs.)

(6) model Integer code for the model to use for load variation with voltage. Valid values are:

1:Standard constant P+jQ load.
2:Constant impedance load. (Default)
3:Const P, Quadratic Q (like a motor).
4:Linear P, Quadratic Q (feeder mix)
5:Not implemented
6:Const P, Fixed Q
7:Const P, Fixed Impedance Q

For Types 6 and 7, only the P is modified by load multipliers.

(7) yearly Load shape to use for yearly simulations. Must be previously defined as a Loadshape object. Defaults to Daily load shape when Daily is defined. The daily load shape is repeated in this case. Otherwise, the default is no variation.

(8) daily Load shape to use for daily simulations. Must be previously defined as a Loadshape object of 24 hrs, typically. Default is no variation (constant) IF not defined. Side effect: Sets Yearly load shape IF not already defined.

(9) duty Load shape to use for duty cycle simulations. Must be previously defined as a Loadshape object. Typically would have time intervals less than 1 hr. Designate the number of points to solve using the Set Number=xxxx command. If there are fewer points in the actual shape, the shape is assumed to repeat. Defaults to Daily curve If not specified.

(10) growth Characteristic to use for growth factors by years. Must be previously defined as a Growthshape object. Defaults to circuit default growth factor (see Set Growth command).

(11) conn ={wye or LN | delta or LL}. Default is wye.

(12) kvar Specify the base kvar. Alternative to specifying the power factor. Side effect: the power factor value is altered to agree.

(13) rneut Neutral resistance of wye (star)-connected load in actual ohms.If entered as a negative value, the neutral is assumed to be open, or floating.

(14) xneut Neutral reactance of wye(star)-connected load in actual ohms. May be + or -.

(15) status ={Variable | Fixed | Exempt}. Default is variable. If Fixed, no load multipliers apply; however, growth multipliers do apply. All multipliers apply to Variable loads. Exempt loads are not modified by the global load multiplier, such as in load duration curves, etc. Daily multipliers do apply, so this is a good way to represent industrial load that stays the same for the period study.

(16) class An arbitrary integer number representing the class of load so that load values may be segregated by load value. Default is 1; not used internally.

(17) vminpu Default = 0.95. Minimum per unit voltage for which the MODEL is assumed to apply. Below this value, the load model reverts to a constant impedance model.

(18) vmaxpu Default = 1.05. Maximum per unit voltage for which the MODEL is assumed to apply. Above this value, the load model reverts to a constant impedance model.

(19) vminnormMinimum per unit voltage for load EEN evaluations, Normal limit. Default = 0, which defaults to system "vminnorm" property (see Set Command under Executive). If this property is specified, it ALWAYS overrides the system specification. This allows you to have different criteria for different loads. Set to zero to revert to the default system value.

(20) vminemerg Minimum per unit voltage for load UE evaluations, Emergency limit. Default = 0, which defaults to system "vminemerg" property (see Set Command under Executive). If this property is specified, it ALWAYS overrides the system specification. This allows you to have different criteria for different loads. Set to zero to revert to the default system value.

(21) xfkva Default = 0.0. Rated kVA of service transformer for allocating loads based on connected kVA at a bus. Side effect: kW, PF, and kvar are modified.

(22) allocationfactor Default = 0.5. Allocation factor for allocating loads based on connected kVA at a bus. Side effect: kW, PF, and kvar are modified by multiplying this factor times the XFKVA (if > 0).

(23) basefreq Base Frequency for ratings.

(24) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = TRANSFORMER

Property Description

(1) phases Number of phases this transformer. Default is 3.

(2) windings Number of windings, this transformers. (Also is the number of terminals) Default is 2.

(3) wdg Set this = to the number of the winding you wish to define. THEN set the values FOR this winding. Repeat FOR each winding. Alternatively, use

the array collections (buses, kvas, etc.) to define the windings. Note: impedances are BETWEEN pairs of windings; they are not the property of a single winding.

(4) bus Bus to which this winding is connected.

(5) conn Connection of this winding. Default is "wye" WITH the neutral solidly grounded.

(6) kv For 2-or 3-phase, enter phase-phase kV rating. Otherwise, kV rating of the actual winding

(7) kva Base kVA rating of the winding. Side effect: forces change of max normal and emerg kva ratings.

(8) tap Per unit tap that this winding is on.

(9) %r Percent resistance this winding. (half of total FOR a 2-winding).

(10) rneut Neutral resistance of wye (star)-connected winding in actual ohms.If entered as a negative value, the neutral is assumed to be open, or floating.

(11) xneut Neutral reactance of wye(star)-connected winding in actual ohms. May be + or -.

(12) buses Use this to specify all the bus connections at once using an array. Example:

```
New Transformer.T1 buses="Hibus, lowbus"
```

(13) conns Use this to specify all the Winding connections at once using an array. Example:

```
New Transformer.T1 buses="Hibus, lowbus" ~ conns=(delta, wye)
```

(14) kvs Use this to specify the kV ratings of all windings at once using an array. Example:

```
New Transformer.T1 buses="Hibus, lowbus"
~ conns=(delta, wye)
~ kvs=(115, 12.47)
```

See kV= property FOR voltage rules.

(15) kvas Use this to specify the kVA ratings of all windings at once using an array.

(16) taps Use this to specify the p.u. tap of all windings at once using an array.

(17) xhl Use this to specify the percent reactance, H-L (winding 1 to winding 2). Use for 2- or 3-winding transformers. On the kva base of winding 1.

(18) xht Use this to specify the percent reactance, H-T (winding 1 to winding 3). Use for 3-winding transformers only. On the kVA base of winding 1.

(19) xlt Use this to specify the percent reactance, L-T (winding 2 to winding 3). Use for 3-winding transformers only. On the kVA base of winding 1.

(20) xscarray Use this to specify the percent reactance between all pairs of windings as an array. All values are on the kVA base of winding 1. The order of the values is as follows:

(x12 13 14... 23 24.. 34 ..)

There will be n(n-1)/2 values, where n=number of windings.

(21) thermal Thermal time constant of the transformer in hours. Typically about 2.

(22) n n Exponent FOR thermal properties in IEEE C57. Typically 0.8.

(23) m m Exponent FOR thermal properties in IEEE C57. Typically 0.9 - 1.0

(24) flrise Temperature rise, deg C, FOR full load. Default is 65.

(25) hsrise Hot spot temperature rise, deg C. Default is 15.

(26) loadloss LoadLoss at full load, kW.

(27) noloadloss No load losses, kW.

(28) normhkvaNormal maximum kVA rating of H winding (winding 1). Usually 100% - 110% ofmaximum nameplate rating, depending on load shape. Defaults to 110% of kVA rating of Winding 1.

(29) emerghkva Emergency (contingency) kVA rating of H winding (winding 1). Usually 140% - 150% ofmaximum nameplate rating, depending on load shape. Defaults to 150% of kVA rating of Winding 1.

(30) sub ={Yes|No} Designates whether this transformer is to be considered a substation.Default is No.

(31) MaxTap Max per unit tap FOR the active winding. Default is 1.10

(32) MinTap Min per unit tap FOR the active winding. Default is 0.90

(33) NumTapsTotal number of taps between min and max tap. Default is 32.

(34) normamps Normal rated current.

(35) emergamps Maximum current.

(36) faultrate No. of failures per year.

(37) pctperm Percent of failures that become permanent.

(38) repair Hours to repair.

(39) basefreq Base Frequency for ratings.

(40) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = REGCONTROL

Property Description

(1) transformer Name of Transformer element to which the RegControl is connected. Do not specify the full object name; "Transformer" is assumed for the object class. Example:

Transformer=Xfmr1

(2) winding Number of the winding of the transformer element to which the RegControl is connected. 1 or 2, typically.

(3) vreg Voltage regulator setting, in VOLTS, for the winding being controlled. Multiplying this value times the ptratio should yield the voltage across the WINDING of the controlled transformer. Default is 120.0

(4) band Bandwidth in VOLTS for the controlled bus (see help for ptratio property). Default is 3.0

(5) ptratio Ratio of the PT that converts the controlled winding voltage to the regulator voltage. Default is 60. If the winding is Wye, the line-to-neutral voltage is used. Else, the line-to-line voltage is used.

(6) CTprim Rating, in Amperes, of the primary CT rating for converting the line amps to control amps.The typical default secondary ampere rating is 5 Amps.

(7) R R setting on the line drop compensator in the regulator, expressed in VOLTS.

(8) X X setting on the line drop compensator in the regulator, expressed in VOLTS.

(9) bus Name of a bus in the system to use as the controlled bus instead of the

bus to which the winding is connected or the R and X line drop compensator settings. Do not specify this value if you wish to use the line drop compensator settings. Default is null string.

(10) delay Time delay, in seconds, from when the voltage goes out of band to when the tap changing begins. This is used to determine which regulator control will act first. Default is 15. You may specify any floating point number to achieve a model of whatever condition is necessary.

(11) reversible {Yes |No} Indicates whether or not the regulator can be switched to regulate in the reverse direction. Default is No.Typically applies only to line regulators and not to LTC on a substation transformer.

(12) revvreg Voltage setting in volts for operation in the reverse direction.

(13) revband Bandwidth for operating in the reverse direction.

(14) revR R line drop compensator setting for reverse direction.

(15) revX X line drop compensator setting for reverse direction.

(16) tapdelay Delay in sec between tap changes. Default is 2. This is how long it takes between changes after the first change.

(17) debugtrace {Yes | No } Default is no. Turn this on to capture the progress of the regulator model for each control iteration. Creates a separate file for each generator named "REG_name.CSV".

(18) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = MONITOR

Property Description

(1) element Name (Full Object name) of element to which the monitor is connected.

(2) terminal Number of the terminal of the circuit element to which the monitor is connected. 1 or 2, typically.

(3) mode Bitmask integer designating the values the monitor is to capture:
0 = Voltages and currents
1 = Powers

Normally, these would be actual phasor quantities from solution.
Combine with adders below to achieve other results:
+16 = Sequence quantities
+32 = Magnitude only
+64 = Positive sequence only or avg of all phases

Mix adder to obtain desired results. For example:
Mode=112 will save positive sequence voltage and current magnitudes only
Mode=48 will save all sequence voltages and currents, but magnitude only.

(4) action {Clear | Save | Take}
(C)lears or (S)aves current buffer.
(T)ake action takes a sample.

Note that monitors are automatically reset (cleared) when the Set Mode= command is issued. Otherwise, the user must explicitly reset all monitors (resetmonitors command) or individual monitors with the Clear action.

(5) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = CAPACITOR

Property Description

(1) bus1 Name of first bus. Examples:
bus1=busname
bus1=busname.1.2.3

(2) bus2 Name of 2nd bus. Defaults to all phases connected to first bus, node 0. (Shunt Wye Connection)
Not necessary to specify for delta (LL) connection

(3) phases Number of phases.

(4) kvar Total kvar, all phases. Evenly divided among phases.

(5) kv For 2, 3-phase, kV phase-phase. Otherwise specify actual can rating.

(6) conn ={wye | delta |LN |LL} Default is wye, which is equivalent to LN

(7) cmatrix Nodal cap. matrix, lower triangle, microfarads, of the following form:

cmatrix="c11 | -c21 c22 | -c31 -c32 c33"

(8) cuf Capacitance, each phase, microfarads.

(9) normamps Normal rated current.

(10) emergamps Maximum current.

(11) faultrate No. of failures per year.

(12) pctperm Percent of failures that become permanent.

(13) repair Hours to repair.

(14) basefreq Base Frequency for ratings.

(15) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = CAPCONTROL

Property Description

(1) element Full object name of the circuit element, typically a line or transformer, to which the capacitor control's PT and/or CT are connected.There is no default; must be specified.

(2) terminal Number of the terminal of the circuit element to which the CapControl is connected. 1 or 2, typically. Default is 1.

(3) capacitor Name of Capacitor element which the CapControl controls. Do not specify the full object name; "Capacitor" is assumed for the object class. Example:

Capacitor=cap1

(4) type {Current | voltage | kvar |time } Control type. Specify the ONsetting and OFFsetting appropriately with the type of control. (See help for ONsetting)

(5) PTratio Ratio of the PT that converts the controlled winding voltage to the regulator voltage. Default is 60. If the winding is Wye, the line-to-neutral voltage is used. Else, the line-to-line voltage is used.

(6) CTratio Ratio of the CT from line amps to control ampere setting for current and kvar control types.

(7) ONsetting Value at which the control arms to switch the capacitor ON.

Type of Control:

Current: Line Amps / CTratio
Voltage: Line-Neutral Volts / PTratio
kvar: Total kvar, all phases
Time: Hrs from Midnight as a floating point number (decimal). 7:30am would be entered as 7.5.

(8) OFFsetting Value at which the control arms to switch the capacitor OFF. (See help for ONsetting)

(9) delay Time delay, in seconds, from when the control is armed before it sends out the switching command. The control may reset before the action actually occurs. This is used to determine which capacity control will act first. Default is 15. You may specify any floating point number to achieve a model of whatever condition is necessary.

(10) VoltOverride {Yes | No} Default is No. Switch to indicate whether VOLTAGE OVERRIDE is to be considered. Vmax and Vmin must be set to reasonable values if this property is Yes.

(11) Vmax Maximum voltage, in volts. If the voltage across the capacitor divided by the PTRATIO is greater than this voltage, the capacitor will switch OFF regardless of other control settings. Default is 126 (goes with a PT ratio of 60 for 12.47 kV system).

(12) Vmin Minimum voltage, in volts. If the voltage across the capacitor divided by the PTRATIO is less than this voltage, the capacitor will switch ON regardless of other control settings. Default is 115 (goes with a PT ratio of 60 for 12.47 kV system).

(13) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = FAULT

Property Description

(1) bus1 Name of first bus. Examples:
bus1=busname
bus1=busname.1.2.3

(2) bus2 Name of 2nd bus. Defaults to all phases connected to first bus, node 0. (Shunt Wye Connection)

(3) phases Number of Phases. Default is 3.

(4) r Resistance, each phase, ohms. Default is 0.0001. Assumed to be Mean value if gaussian random mode.Max value if uniform mode. A Fault is actually a series resistance that defaults to a wye connection to ground on the second terminal. You may reconnect the 2nd terminal to achieve whatever connection. Use the Gmatrix property to specify an arbitrary conductance matrix.

(5) %stddev Percent standard deviation in resistance to assume for Monte Carlo fault (MF) solution mode. Default is 0 (no variation from mean).

(6) Gmatrix Use this to specify a nodal conductance (G) matrix to represent some arbitrary resistance network. Specify in lower triangle form as usual for DSS matrices.

(7) normamps Normal rated current.
(8) emergamps Maximum current.
(9) faultrate No. of failures per year.
(10) pctperm Percent of failures that become permanent.
(11) repair Hours to repair.
(12) basefreq Base Frequency for ratings.
(13) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = GENERATOR
Property Description
(1) phases Number of Phases, this Generator. Power is evenly divided among phases.
(2) bus1 Bus to which the Generator is connected. May include specific node specification.
(3) kv Nominal rated (1.0 per unit) voltage, kV, for Generator. For 2- and 3-phase Generators, specify phase-phase kV. Otherwise, specify actual kV across each branch of the Generator. If wye (star), specify phase-neutral kV. If delta or phase-phase connected, specify phase-phase kV.
(4) kw Total base kW for the Generator. A positive value denotes power coming OUT of the element,
which is the opposite of a load. This value is modified depending on the dispatch mode. Unaffected by the global load multiplier and growth curves. If you want there to be more generation, you must add more generators or change this value.
(5) pf Generator power factor. Default is 0.80. Enter negative for leading powerfactor (when kW and kvar have opposite signs.)
A positive power factor for a generator signifies that the generator produces vars
as is typical for a synchronous generator. Induction machines would be specified with a negative power factor.
(6) model Integer code for the model to use for generation variation with voltage. Valid values are:

1:Generator injects a constant kW at specified power factor.
2:Generator is modeled as a constant admittance.
3:Const kW, constant kV. Somewhat like a conventional transmission power flow P-V generator.
(7) yearly Dispatch shape to use for yearly simulations. Must be previously defined as a Loadshape object. If this is not specified, the daily dispatch shape is repeated. If the generator is assumed to be ON continuously, specify this value as FIXED, or designate a curve that is 1.0 per unit at all times. Nominally for 8760 simulations. If there are fewer points in the designated shape than the number of points in the solution, the curve is repeated.
(8) daily Dispatch shape to use for daily simulations. Must be previously defined as a Loadshape object of 24 hrs, typically. If generator is assumed to be ON continuously, specify this value as FIXED, or designate a Loadshape objectthat is 1.0 perunit for all hours.

(9) duty Load shape to use for duty cycle dispatch simulations such as for wind generation. Must be previously defined as a Loadshape object. Typically would have time intervals less than 1 hr -- perhaps, in seconds. Designate the number of points to solve using the Set Number=xxxx command. If there are fewer points in the actual shape, the shape is assumed to repeat.

(10) dispmode {Default | Loadlevel | Price } Default = Default. Dispatch mode. In default mode, gen is either always on or follows dispatch curve as specified. Otherwise, the gen comes on when either the global default load level or the price level exceeds the dispatch value.

(11) dispvalue Dispatch value.
If = 0.0 Then Generator follow dispatch curves, if any.
If > 0 Then Generator is ON only when either the price signal exceeds this value or the load multiplier (set loadmult=) times the default yearly growth factor exceeds this value. Then the generator follows dispatch curves, if any (see also Status).

(12) conn ={wye|LN|delta|LL}. Default is wye.

(13) kvar Specify the base kvar. Alternative to specifying the power factor. Side effect: the power factor value is altered to agree.

(14) rneut Neutral resistance of wye (star)-connected Generator in actual ohms.If entered as a negative value, the neutral is assumed to be open, or floating.

(15) xneut Neutral reactance of wye(star)-connected Generator in actual ohms. May be + or -.

(16) status ={Fixed|Variable}. If Fixed, then dispatch multipliers do not apply. The generator is alway at full power when it is ON. Default is Variable (follows curves).

(17) class An arbitrary integer number representing the class of Generator so that Generator values may be segregated by class.

(18) vpu Per Unit voltage set point for Model = 3 (typical power flow model). Default is 1.0.

(19) maxkvar Maximum kvar limit for Model = 3. Defaults to twice the specified load kvar. Always reset this if you change PF or kvar properties.

(20) minkvar Minimum kvar limit for Model = 3. Enter a negative number if generator can absorb vars. Defaults to negative of Maxkvar. Always reset this if you change PF or kvar properties.

(21) pvfactor Deceleration factor for P-V generator model (Model=3). Default is 0.1. If the circuit converges easily, you may want to use a higher number such as 1.0. Use a lower number if solution diverges. Use Debugtrace=yes to create a file that will trace the convergence of a generator model.

(22) debugtrace {Yes | No } Default is no. Turn this on to capture the progress of the generator model for each iteration. Creates a separate file for each generator named "GEN_name.CSV".

(23) vminpu Default = 0.95. Minimum per unit voltage for which the Model is assumed to apply. Below this value, the load model reverts to a constant impedance model.

(24) vmaxpu Default = 1.05. Maximum per unit voltage for which the Model is assumed to apply. Above this value, the load model reverts to a constant impedance model.

(25) forceon {Yes | No} Forces generator ON despite requirements of other dispatch

modes. Stays ON until this property is set to NO, or an internal algorithm cancels the forced ON state.
(26) basefreq Base Frequency for ratings.
(27) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = RELAY
Property Description
(1) MonitoredObj Full object name of the circuit element, typically a line, transformer, load, or generator, to which the relay's PT and/or CT are connected. This is the "monitored" element. There is no default; must be specified.
(2) MonitoredTerm Number of the terminal of the circuit element to which the Relay is connected. 1 or 2, typically. Default is 1.
(3) SwitchedObj Name of circuit element switch that the Relay controls. Specify the full object name.Defaults to the same as the Monitored element. This is the "controlled" element.
(4) SwitchedTerm Number of the terminal of the controlled element in which the switch is controlled by the Relay. 1 or 2, typically. Default is 1.
(5) type {Current | voltage | Reversepower } Relay type. Default is overcurrent relay (Current) Specify the curve and pickup settings appropriate for each type.
(6) Phasecurve Name of the TCC Curve object that determines the phase trip. Must have been previously defined as a TCC_Curve object. Default is none (ignored). For overcurrent relay, multiplying the current values in the curve by the "phasetrip" valuw gives the actual current.
(7) Groundcurve Name of the TCC Curve object that determines the ground trip. Must have been previously defined as a TCC_Curve object. Default is none (ignored).For overcurrent relay, multiplying the current values in the curve by the "groundtrip" valuw gives the actual current.
(8) PhaseTrip Multiplier or actual phase amps for the phase TCC curve. Defaults to 1.0.
(9) GroundTrip Multiplier or actual ground amps (3I0) for the ground TCC curve. Defaults to 1.0.
(10) PhaseInst Actual amps (Current relay) or kW (reverse power relay) for instantaneous phase trip which is assumed to happen in 0.01 sec + Delay Time. Default is 0.0, which signifies no inst trip. Use this value for specifying the Reverse Power threshold (kW) for reverse power relays.
(11) GroundInst Actual amps for instantaneous ground trip which is assumed to happen in 0.01 sec + Delay Time.Default is 0.0, which signifies no inst trip.
(12) Reset Reset time in sec for relay. Default is 15.
(13) RecloseIntervals Array of reclose intervals. Default is (0.5, 2.0, 2.0) seconds. NOTE: Reverse power relay is one shot to lockout, so this is ignored. A locked out relay must be closed manually (set action=close).
(14) Overvoltcurve TCC Curve object to use for overvoltage relay. Curve is assumed to be defined with per unit voltage values. Voltage base should be defined for the relay. Default is none (ignored).
(15) Undervoltcurve TCC Curve object to use for undervoltage relay. Curve is assumed to be defined with per unit voltage values. Voltage base should

be defined for the relay. Default is none (ignored).

(16) kvbase Voltage base (kV) for the relay. Specify line-line for 3 phase devices; line-neutral for 1-phase devices. Relay assumes the number of phases of the monitored element. Default is 0.0, which results in assuming the voltage values in the "TCC" curve are specified in actual line-to-neutral volts.

(17) delay Fixed delay time (sec) added to relay time. Default is 0.0. Reverse power relay uses this exclusively. Added to trip time of current and voltage relays. Could use in combination with inst trip value to obtain a definite time overcurrent relay.

(18) action {Trip/Open | Close} Action that overrides the relay control. Simulates manual control on breaker. "Trip" or "Open" causes the controlled element to open and lock out. "Close" causes the controlled element to close and the relay to reset to its first operation.

(19) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

Object = ENERGYMETER

Property Description

(1) element Name (Full Object name) of element to which the monitor is connected.

(2) terminal Number of the terminal of the circuit element to which the monitor is connected. 1 or 2, typically.

(3) action {Clear (reset) | Save | Take | Zonedump | Allocate}

(A)llocate = Allocate loads on the meter zone to match PeakCurrent.
(C)lear or (R)eset = reset all registers to zero
(S)ave = saves the current register values to a file.
File name is "MTR_metername.CSV".
(T)ake = Takes a sample at present solution
(Z)onedump = Dump names of elements in meter zone to a file
File name is "Zone_metername.CSV".

(4) option Enter a string ARRAY of any combination of the following. Options processed left-to-right:

(E)xcess : (default) UE/EEN is estimate of energy over capacity
(T)otal : UE/EEN is total energy after capacity exceeded
(R)adial : (default) Treats zone as a radial circuit
(M)esh : Treats zone as meshed network (not radial).

Example: option=(E, R)

(5) kwnormal Upper limit on kW load in the zone, Normal configuration. Default is 0.0 (ignored). Overrides limits on individual lines for overload EEN.

(6) kwemerg Upper limit on kW load in the zone, Emergency configuration. Default is 0.0 (ignored). Overrides limits on individual lines for overload UE.

(7) peakcurrent ARRAY of current magnitudes representing the peak currents measured at this location for the load allocation function. Default is (400, 400, 400). Enter one current for each phase

(8) like Make like another object, e.g.:

New Capacitor.C2 like=c1 ...

COM Interface Reference

DSSLIB Interface Definition

Enumerations

This section lists enumerations exposed by DSSLIB.

```
Public Enum MonitorModes
        dssVI=0
        dssPower=1
        dssSequence=16
        dssMagnitude=32
        dssPosOnly=64
End Enum
Public Enum Options
        dssPowerFlow=1
        dssAdmittance=2
        dssNormalSolve=0
        dssNewtonSolve=1
        dssStatic=0
        dssEvent=1
        dssTime=2
        dssMultiphase=0
        dssPositiveSeq=1
        dssGaussian=1
        dssUniform=2
        dssLogNormal=3
        dssAddGen=1
        dssAddCap=2
End Enum
Public Enum SolveModes
        dssSnapShot=0
        dssDutyCycle=6
        dssDirect=7
        dssDaily=1
        dssMonte1=3
        dssMonte2=10
        dssMonte3=11
        dssFaultStudy=9
        dssYearly=2
        dssMonteFault=8
        dssPeakDay=5
        dssLD1=4
        dssLD2=12
        dssAutoAdd=13
End Enum
```

Interfaces

This section lists the Classes exposed by DSSLIB. For each class, the methods and events are listed.

IBus {4CE105CD-E76E-11D2-B339-00A024DB8C85}

Methods
Property Get Name() As String
Name of Bus_F
Property Get NumNodes() As Long
Number of Nodes this bus._F
Property Get Voltages() As Variant
Complex array of voltages at this bus._F
Property Get SeqVoltages() As Variant
Double Array of sequence voltages at this bus._F
Property Get Nodes() As Variant
Integer Array of Node Numbers defined at the bus._F
Property Get Voc() As Variant
Open circuit voltage; Complex array.*#4ß_
Property Get Isc() As Variant
Short circuit currents at bus; Complex Array.
Events
None

ICircuit {4CE105CA-E76E-11D2-B339-00A024DB8C85}

Methods
Property Get Name() As String
Name of the active circuit.*#4*#a*#4*#e_*#10_*#12_*#1
Property Get NumCktElements() As Long
Number of CktElements in the circuit.*#4*#a*#4*#e_*#10_*#12_*#1
Property Get NumBuses() As Long
Total number of Buses in the circuit.*#4*#a*#4*#e_*#10_*#12_*#1
Property Get NumNodes() As Long
Total number of nodes in the circuit.*#4*#a*#4*#e_*#10_*#12_*#1
Property Get Buses(ByVal Index As Variant) As IBus
Collection of Buses in the circuit
Property Get CktElements(ByVal Idx As Variant) As ICktElement
Collection of CktElements in Circuit
Property Get Losses() As Variant
Total losses in active circuit, complex number (two-element array of double).
Property Get LineLosses() As Variant
Complex total line losses in the circuit
Property Get SubstationLosses() As Variant
Complex losses in all transformers designated to substations.
Property Get TotalPower() As Variant
Total power, watts delivered to the circuit
Property Get AllBusVolts() As Variant

Complex array of all bus, node voltages from most recent solution
Property Get AllBusVmag() As Variant
Array of magnitudes (doubles) of voltages at all buses
Property Get AllElementNames() As Variant
Array of strings containing all element names.
Property Get ActiveElement() As ICktElement
Return an interface to the active circuit element
Sub Disable(ByVal Name As String)
Disable a circuit element by name (removes from circuit but leave in database)
Sub Enable(ByVal Name As String)
Activate (enable) a disabled device.
Property Get Solution() As ISolution
Return an interface to the Solution object.
Property Get ActiveBus() As IBus
Return an interface to the active bus.
Function FirstPCElement() As Long
Sets the first Power Conversion (PC) element to be the active element.
Function NextPCElement() As Long
Gets next PC Element. Returns 0 if no more.
Function FirstPDElement() As Long
Sets the first Power Delivery (PD) element to be the active element.
Function NextPDElement() As Long
Gets next PD Element. Returns 0 if no more.
Property Get AllBusNames() As Variant
Array of strings containing names of all buses in circuit.
Property Get AllElementLosses() As Variant
Array of total losses (complex) in each circuit element
Sub Sample
Force all Meters and Monitors to take a sample.
Sub SaveSample
Force all meters and monitors to save their current buffers.
Property Get Monitors() As IMonitors
Returns a Monitors object interface.
Property Get Meters() As IMeters
Returns a Meters object interface (Energy Meters)
Property Get Generators() As IGenerators
Returns a Generators Object interface
Property Get Settings() As ISettings
Returns interface to Settings interface.
Property Get Lines() As ILines
Returns Interface to Line elements interface.
Function SetActiveElement(ByVal FullName As String) As Long
Sets the Active Circuit Element using the full object name (e.g. "generator.g1"). Returns 0 if not found. Else positive integer.
Events
None

ICktElement {4CE105C6-E76E-11D2-B339-00A024DB8C85}

Methods

Property Get Name() As String
Name of Circuit Element*#4*#2ß_
Property Get NumTerminals() As Long
Number of Terminals this Circuit Element*#4*#2ß_
Property Get NumConductors() As Long
Number of Conductors per Terminal*#4*#2ß_
Property Get NumPhases() As Long
Number of Phases*#4*#2ß_
Property Get BusNames() As Variant
Set Names of the Buses to which each terminal is connected.*#2ß_
Property Let BusNames(RHS As Variant)
Set Names of the Buses to which each terminal is connected.*#2ß_
Property Get Properties(ByVal Indx As Variant) As IDSSProperty
Collection of Properties for this Circuit Element (0 based index, if numeric)
Property Get Voltages() As Variant
Complex array of voltages at terminals
Property Get Currents() As Variant
Complex array of currents into each conductor of each terminal
Property Get Powers() As Variant
Complex array of powers into each conductor of each terminal
Property Get Losses() As Variant
Total losses in the element: two-element complex array
Property Get PhaseLosses() As Variant
Complex array of losses by phase
Property Get SeqVoltages() As Variant
Double array of symmetrical component voltages at each 3-phase terminal
Property Get SeqCurrents() As Variant
Double array of symmetrical component currents into each 3-phase terminal
Property Get SeqPowers() As Variant
Double array of sequence powers into each 3-phase teminal
Property Get Enabled() As Boolean
Boolean indicating that element is currently in the circuit.
Property Let Enabled(RHS As Boolean)
Boolean indicating that element is currently in the circuit.
Property Get NormalAmps() As Double
Normal ampere rating
Property Let NormalAmps(RHS As Double)
Normal ampere rating
Property Get EmergAmps() As Double
Emergency Ampere Rating
Property Let EmergAmps(RHS As Double)
Emergency Ampere Rating
Sub Open(ByVal Term As Long, ByVal Phs As Long)
Open the specified terminal and phase, if non-zero. Else all conductors at terminal.
Sub Close(ByVal Term As Long, ByVal Phs As Long)
Close the specified terminal and phase, if non-zero. Else all conductors at terminal.
Function IsOpen(ByVal Term As Long, ByVal Phs As Long) As Boolean
Boolean indicating if the specified terminal and, optionally, phase is open.

Property Get NumProperties() As Long
Number of Properties this Circuit Element.
Property Get AllPropertyNames() As Variant
Variant array containing all property names of the active device.
Events
None

IDSS {4CE105CF-E76E-11D2-B339-00A024DB8C85}

Methods
Property Get NumCircuits() As Long
Number of Circuits currently defined*#4*#8*#0*#6_*#16ß_
Property Get Circuits(ByVal Idx As Variant) As ICircuit
Collection of Circuit objects*#4*#8*#0*#6_*#16ß_
Property Get ActiveCircuit() As ICircuit
Returns interface to the active circuit.*#4*#8*#0*#6_*#16ß_
Property Get Text() As IText
Returns the DSS Text (command-result) interface.*#4*#8*#0*#6_*#16ß_
Property Get Error() As IError
Returns Error interface.*#4*#8*#0*#6_*#16ß_
Function NewCircuit(ByVal Name As String) As ICircuit
Make a new circuit and return interface to active circuit.
Sub ClearAll
Clears all circuit definitions.
Sub ShowPanel
Shows control panel for DSS.
Function Start(ByVal code As Long) As Boolean
Validate the user and start the DSS. (old method) Returns TRUE if successful.
Property Get Version() As String
Get version string for the DSS.
Property Get DSSProgress() As IDSSProgress
Gets interface to the DSS Progress Meter
Events
None

IDSSProgress {FAA79841-525D-11D3-AD84-444553540000}

Methods
Property Let PctProgress(RHS As Long)
Percent progress to indicate [0..100]*#4ß_
Property Let Caption(RHS As String)
Caption to appear on the bottom of the DSS Progress form.
Sub Show
Shows progress form with null caption and progress set to zero.
Sub Close
Closes (hides) DSS Progress form.
Events
None

IDSSProperty {4CE105C4-E76E-11D2-B339-00A024DB8C85}

Methods

Property Get Name() As String

Name of Property*#4ß_

Property Get Description() As String

Description of the property.*#4ß_

Property Get Val() As String

Property Let Val(RHS As String)

Events

None

IError {4CE105C8-E76E-11D2-B339-00A024DB8C85}

Methods

Property Get Number() As Long

Error Number*#4ß_

Property Get Description() As String

Description of error for last operation*#4ß_

Events

None

IGenerators {4FFBDD30-0474-11D3-B339-00A024DB8C85}

Methods

Property Get AllNames() As Variant

Array of names of all Generator objects.*#4ß_

Property Get RegisterNames() As Variant

Array of Names of all generator energy meter registers

Property Get RegisterValues() As Variant

Array of valus in generator energy meter registers.

Property Get First() As Long

Sets first Generator to be active. Returns 0 if none.

Property Get Next() As Long

Sets next Generator to be active. Returns 0 if no more.

Property Get ForcedON() As Boolean

Indicates whether the generator is forced ON regardles of other dispatch criteria.

Property Let ForcedON(RHS As Boolean)

Indicates whether the generator is forced ON regardles of other dispatch criteria.

Property Get Name() As String

Sets a generator active by name.

Property Let Name(RHS As String)

Sets a generator active by name.

Events

None

ILines {083E2420-7B74-11D4-B33A-00A024DB8C85}

Methods

Property Get Name() As String
Specify the name of the Line element to set it active.
Property Let Name(RHS As String)
Specify the name of the Line element to set it active.
Property Get AllNames() As Variant
Names of all Line Objects
Property Get First() As Long
Invoking this property sets the first element active. Returns 0 if no lines. Otherwise, index of the line element.
Property Get Next() As Long
Invoking this property advances to the next Line element active. Returns 0 if no more lines. Otherwise, index of the line element.
Function New(ByVal Name As String) As Long
Creates a new Line and makes it the Active Circuit Element.__
Property Get Bus1() As String
Name of bus for terminal 1.
Property Let Bus1(RHS As String)
Name of bus for terminal 1.
Property Get Bus2() As String
Name of bus for terminal 2.
Property Let Bus2(RHS As String)
Name of bus for terminal 2.
Property Get LineCode() As String
Name of LineCode object that defines the impedances.
Property Let LineCode(RHS As String)
Name of LineCode object that defines the impedances.
Property Get Length() As Double
Length of line section in units compatible with the LineCode definition.
Property Let Length(RHS As Double)
Length of line section in units compatible with the LineCode definition.
Property Get Phases() As Long
Number of Phases, this Line element.
Property Let Phases(RHS As Long)
Number of Phases, this Line element.
Property Get R1() As Double
Positive Sequence resistance, ohms per unit length.
Property Let R1(RHS As Double)
Positive Sequence resistance, ohms per unit length.
Property Get X1() As Double
Positive Sequence reactance, ohms per unit length.
Property Let X1(RHS As Double)
Positive Sequence reactance, ohms per unit length.
Property Get R0() As Double
Zero Sequence resistance, ohms per unit length.
Property Let R0(RHS As Double)
Zero Sequence resistance, ohms per unit length.
Property Get X0() As Double
Zero Sequence reactance ohms per unit length.
Property Let X0(RHS As Double)

Zero Sequence reactance ohms per unit length.
Property Get C1() As Double
Positive Sequence capacitance, nanofarads per unit length.
Property Let C1(RHS As Double)
Positive Sequence capacitance, nanofarads per unit length.
Property Get C0() As Double
Zero Sequence capacitance, nanofarads per unit length.
Property Let C0(RHS As Double)
Zero Sequence capacitance, nanofarads per unit length.
Property Get Rmatrix() As Variant
Resistance matrix (full), ohms per unit length. Variant array of doubles.
Property Let Rmatrix(RHS As Variant)
Resistance matrix (full), ohms per unit length. Variant array of doubles.
Property Get Xmatrix() As Variant

Property Let Xmatrix(RHS As Variant)

Property Get Cmatrix() As Variant

Property Let Cmatrix(RHS As Variant)

Property Get NormAmps() As Double
Normal ampere rating of Line.
Property Let NormAmps(RHS As Double)
Normal ampere rating of Line.
Property Get EmergAmps() As Double
Emergency (maximum) ampere rating of Line.
Property Let EmergAmps(RHS As Double)
Emergency (maximum) ampere rating of Line.‘E
Events
None

IMeters {4FFBDD2C-0474-11D3-B339-00A024DB8C85}

Methods
Property Get AllNames() As Variant
Array of all energy Meter names
Property Get First() As Long
Set the first energy Meter active. Returns 0 if none.
Property Get Next() As Long
Sets the next energy Meter active. Returns 0 if no more.
Property Get RegisterNames() As Variant
Array of strings containing the names of the registers.
Property Get RegisterValues() As Variant
Array of all the values contained in the Meter registers for the active Meter.
Sub Reset
Resets registers of active Meter.
Sub ResetAll
Resets registers of all Meter objects.

Sub Sample
Forces active Meter to take a sample._
Sub Save
Saves meter register values._
Property Get Name() As String
Set a meter to be active by name._
Property Let Name(RHS As String)
Set a meter to be active by name._
Events
None

IMonitors {4FFBDD28-0474-11D3-B339-00A024DB8C85}

Methods
Property Get AllNames() As Variant
Array of all Monitor Names
Property Get First() As Long
Sets the first Monitor active. Returns 0 if no monitors.
Property Get Next() As Long
Sets next monitor active. Returns 0 if no more.
Sub Reset
Resets active Monitor object.
Sub ResetAll
Resets all Monitor Objects
Sub Sample
Causes active Monitor to take a sample.
Sub Save
Causes active monitor to save its current buffer to disk.
Sub Show
Converts monitor file to text and displays with text editor
Property Get FileName() As String
Name of disk file associated with active Monitor.
Property Get Mode() As Long
Set Monitor mode (bitmask integer - see DSS Help)
Property Let Mode(RHS As Long)
Set Monitor mode (bitmask integer - see DSS Help)
Property Get Name() As String
Sets the active Monitor object by name
Property Let Name(RHS As String)
Sets the active Monitor object by name
Events
None

ISettings {A7ED88D0-735A-11D4-B33A-00A024DB8C85}

Methods
Property Get AllowDuplicates() As Boolean
True | False* Designates whether to allow duplicate names of objects
Property Let AllowDuplicates(RHS As Boolean)
True | False* Designates whether to allow duplicate names of objects

Property Get ZoneLock() As Boolean
True | False* Locks Zones on energy meters to prevent rebuilding if a circuit change occurs.
Property Let ZoneLock(RHS As Boolean)
True | False* Locks Zones on energy meters to prevent rebuilding if a circuit change occurs.
Property Let AllocationFactors(RHS As Double)
Sets all load allocation factors for all loads defined by XFKVA property to this value.
Property Get AutoBusList() As String
List of Buses or (File=xxxx) syntax for the AutoAdd solution mode.
Property Let AutoBusList(RHS As String)
List of Buses or (File=xxxx) syntax for the AutoAdd solution mode.
Property Get CktModel() As Long
dssMultiphase * | dssPositiveSeq IIndicate if the circuit model is positive sequence.
Property Let CktModel(RHS As Long)
dssMultiphase * | dssPositiveSeq IIndicate if the circuit model is positive sequence.
Property Get NormVminpu() As Double
Per Unit minimum voltage for Normal conditions.
Property Let NormVminpu(RHS As Double)
Per Unit minimum voltage for Normal conditions.
Property Get NormVmaxpu() As Double
Per Unit maximum voltage for Normal conditions.
Property Let NormVmaxpu(RHS As Double)
Per Unit maximum voltage for Normal conditions.
Property Get EmergVminpu() As Double
Per Unit minimum voltage for Emergency conditions.
Property Let EmergVminpu(RHS As Double)
Per Unit minimum voltage for Emergency conditions.
Property Get EmergVmaxpu() As Double
Per Unit maximum voltage for Emergency conditions.
Property Let EmergVmaxpu(RHS As Double)
Per Unit maximum voltage for Emergency conditions.
Property Get UEweight() As Double
Weighting factor applied to UE register values.
Property Let UEweight(RHS As Double)
Weighting factor applied to UE register values.
Property Get LossWeight() As Double
Weighting factor applied to Loss register values.
Property Let LossWeight(RHS As Double)
Weighting factor applied to Loss register values.
Property Get UEregs() As Variant
Array of Integers defining energy meter registers to use for computing UE
Property Let UEregs(RHS As Variant)
Array of Integers defining energy meter registers to use for computing UE
Property Get LossRegs() As Variant
Integer array defining which energy meter registers to use for computing losses
Property Let LossRegs(RHS As Variant)
Integer array defining which energy meter registers to use for computing losses
Property Get Trapezoidal() As Boolean

True | False * Gets value of trapezoidal integration flag in energy meters.
Property Let Trapezoidal(RHS As Boolean)
True | False * Gets value of trapezoidal integration flag in energy meters.
Property Get VoltageBases() As Variant
Array of doubles defining the legal voltage bases in kV L-L
Property Let VoltageBases(RHS As Variant)
Array of doubles defining the legal voltage bases in kV L-L
Property Get ControlTrace() As Boolean
True | False* Denotes whether to trace the control actions to a file.
Property Let ControlTrace(RHS As Boolean)
True | False* Denotes whether to trace the control actions to a file.
Property Get PriceSignal() As Double
Price Signal for the Circuit
Property Let PriceSignal(RHS As Double)
Price Signal for the Circuit
Property Get PriceCurve() As String
Name of LoadShape object that serves as the source of price signal data for yearly simulations, etc.
Property Let PriceCurve(RHS As String)
Name of LoadShape object that serves as the source of price signal data for yearly simulations, etc.
Events
None

ISolution {4CE105D1-E76E-11D2-B339-00A024DB8C85}

Methods
Sub Solve
Execute solution for present solution mode.
Property Get Mode() As Long
Set present solution mode (by a text code - see DSS Help)
Property Let Mode(RHS As Long)
Set present solution mode (by a text code - see DSS Help)
Property Get Frequency() As Double
Set the Frequency for next solution
Property Let Frequency(RHS As Double)
Set the Frequency for next solution
Property Get Hour() As Long
Set Hour for time series solutions.
Property Let Hour(RHS As Long)
Set Hour for time series solutions.
Property Get Seconds() As Double
Seconds from top of the hour._
Property Let Seconds(RHS As Double)
Seconds from top of the hour.
Property Get StepSize() As Double
Time step size in sec
Property Let StepSize(RHS As Double)
Time step size in sec

Property Get Year() As Long
Set year for planning studies
Property Let Year(RHS As Long)
Set year for planning studies
Property Get LoadMult() As Double
Default load multiplier applied to all non-fixed loads
Property Let LoadMult(RHS As Double)
Default load multiplier applied to all non-fixed loads
Property Get Iterations() As Long
Number of iterations take for last solution.
Property Get MaxIterations() As Long
Max allowable iterations.
Property Let MaxIterations(RHS As Long)
Max allowable iterations.
Property Get Tolerance() As Double
Solution convergence tolerance.
Property Let Tolerance(RHS As Double)
Solution convergence tolerance.
Property Get Number() As Long
Number of solutions to perform for Monte Carlo and time series simulations
Property Let Number(RHS As Long)
Number of solutions to perform for Monte Carlo and time series simulations
Property Get Random() As Long
Randomization mode for random variables "Gaussian" or "Uniform"
Property Let Random(RHS As Long)
Randomization mode for random variables "Gaussian" or "Uniform"
Property Get ModeID() As String
ID (text) of the present solution mode
Property Get LoadModel() As Long
Load Model: dssPowerFlow (default) | dssAdmittance
Property Let LoadModel(RHS As Long)
Load Model: dssPowerFlow (default) | dssAdmittance
Property Get LDCurve() As String
Load-Duration Curve name for LD modes
Property Let LDCurve(RHS As String)
Load-Duration Curve name for LD modes
Property Get pctGrowth() As Double
Percent default annual load growth rate
Property Let pctGrowth(RHS As Double)
Percent default annual load growth rate
Property Get AddType() As Long
Type of device to add in AutoAdd Mode: dssGen (Default) | dssCap
Property Let AddType(RHS As Long)
Type of device to add in AutoAdd Mode: dssGen (Default) | dssCap
Property Get GenkW() As Double
Generator kW for AutoAdd modedssGen (Default) | dssCap
Property Let GenkW(RHS As Double)
Generator kW for AutoAdd modedssGen (Default) | dssCap
Property Get GenPF() As Double

PF for generators in AutoAdd modedssGen (Default) | dssCap
Property Let GenPF(RHS As Double)
PF for generators in AutoAdd mode'E
Property Get Capkvar() As Double
Capacitor kvar for adding capacitors in AutoAdd mode'E
Property Let Capkvar(RHS As Double)
Capacitor kvar for adding capacitors in AutoAdd mode
Property Get Algorithm() As Long
Base Solution algorithm: dssNormalSolve | dssNewtonSolve
Property Let Algorithm(RHS As Long)
Base Solution algorithm: dssNormalSolve | dssNewtonSolve
Property Get ControlMode() As Long
dssStatic* | dssEvent | dssTime Modes for control devices
Property Let ControlMode(RHS As Long)
dssStatic* | dssEvent | dssTime Modes for control devices
Property Get GenMult() As Double
Default Multiplier applied to generators (like LoadMult)
Property Let GenMult(RHS As Double)
Default Multiplier applied to generators (like LoadMult)
Property Get DefaultDaily() As String
Default daily load shape (defaults to "Default")
Property Let DefaultDaily(RHS As String)
Default daily load shape (defaults to "Default")
Property Get DefaultYearly() As String
Default Yearly load shape (defaults to "Default")
Property Let DefaultYearly(RHS As String)
Default Yearly load shape (defaults to "Default")
Events
None

IText {4CE105C1-E76E-11D2-B339-00A024DB8C85}

Methods
Property Get Command() As String
Input command string for the DSS.
Property Let Command(RHS As String)
Input command string for the DSS.
Property Get Result() As String
Result string for the last command.
Events
None

Index