OpenDSS Type Library Documentation

June 24, 2014

Version 7.6.3.31

Enumerations

enum MonitorModes

Enum: **dssVI = 0,**
Monitor records Voltage and Current at the terminal (Default

Enum: **dssPower = 1,**
Monitor records kW, kvar or kVA, angle values, etc. at the terminal to which it is connected.

Enum: **dssSequence = 16,**
Reports the monitored quantities as sequence quantities

Enum: **dssMagnitude = 32,**
Reports the monitored quantities in Magnitude Only

Enum: **dssPosOnly = 64,**
Reports the Positive Seq only or avg of all phases

Enum: **dssTaps = 2,**
For monitoring Regulator and Transformer taps

Enum: **dssStates = 3**
For monitoring State Variables (for PC Elements only

enum SolveModes

Enum: **dssSnapShot = 0,**
Solve a single snapshot power flow

Enum: **dssDutyCycle = 6,**
Solve following Duty Cycle load shapes

Enum: **dssDirect = 7,**
Solve direct (forced admittance model

Enum: **dssDaily = 1,**

Solve following Daily load shapes

Enum: **dssMonte1 = 3,**
Monte Carlo Mode 1

Enum: **dssMonte2 = 10,**
Monte Carlo Mode 2

Enum: **dssMonte3 = 11,**
Monte Carlo Mode 3

Enum: **dssFaultStudy = 9,**
Fault study at all buses

Enum: **dssYearly = 2,**
Solve following Yearly load shapes

Enum: **dssMonteFault = 8,**
Monte carlo Fault Study

Enum: **dssPeakDay = 5,**
Solves for Peak Day using Daily load curve

Enum: **dssLD1 = 4,**
Load-duration Mode 1

Enum: **dssLD2 = 12,**
Load-Duration Mode 2

Enum: **dssAutoAdd = 13,**
Auto add generators or capacitors

Enum: **dssHarmonic = 15,**
(no Help string available)

Enum: **dssDynamic = 14**
(no Help string available)

enum Options

Enum: **dssPowerFlow = 1,**
Power Flow load model option

Enum: **dssAdmittance = 2,**

Admittance load model option

Enum: **dssNormalSolve = 0,**
Solution algorithm option - Normal solution mode

Enum: **dssNewtonSolve = 1,**
Solution algorithm option - Newton solution

Enum: **dssStatic = 0,**
Control Mode option - Static

Enum: **dssEvent = 1,**
Control Mode Option - Event driven solution mode

Enum: **dssTime = 2,**
Control mode option - Time driven mode

Enum: **dssMultiphase = 0,**
Circuit model is multiphase (default

Enum: **dssPositiveSeq = 1,**
Circuit model is positive sequence model only

Enum: **dssGaussian = 1,**
Random mode = Gaussian

Enum: **dssUniform = 2,**
Random mode = Uniform

Enum: **dssLogNormal = 3,**
Random Mode = Log normal

Enum: **dssAddGen = 1,**
Add generators in AutoAdd mode (AddType

Enum: **dssAddCap = 2**
Add capacitors in AutoAdd mode (Addtype

enum CapControlModes

Enum: **dssCapControlVoltage = 1,**
voltage control, ON and OFF settings on the PT secondary base

Enum: **dssCapControlKVAR = 2,**

kVAR control, ON and OFF settings on PT / CT base

Enum: **dssCapControlCurrent = 0,**
Current control, ON and OFF settings on CT secondary

Enum: **dssCapControlPF = 4,**
ON and OFF settings are power factor, negative for leading

Enum: **dssCapControlTime = 3**
Time control, ON and OFF settings are seconds from midnight

enum ActionCodes

Enum: **dssActionNone = 0,**
No action

Enum: **dssActionOpen = 1,**
Open a switch

Enum: **dssActionClose = 2,**
Close a switch

Enum: **dssActionReset = 3,**
Reset to the shelf state (unlocked, closed for a switch

Enum: **dssActionLock = 4,**
Lock a switch, prventing both manual and automatic operation

Enum: **dssActionUnlock = 5,**
Unlock a switch, permitting both manual and automatic operation

Enum: **dssActionTapUp = 6,**
Move a regulator tap up

Enum: **dssActionTapDown = 7**
Move a regulator tap down

enum LoadStatus

Enum: **dssLoadVariable = 0,**
(no Help string available)

Enum: **dssLoadFixed = 1,**

(no Help string available)

Enum: **dssLoadExempt = 2**
(no Help string available)

enum LoadModels

Enum: **dssLoadConstPQ = 1,**
(no Help string available)

Enum: **dssLoadConstZ = 2,**
(no Help string available)

Enum: **dssLoadMotor = 3,**
(no Help string available)

Enum: **dssLoadCVR = 4,**
(no Help string available)

Enum: **dssLoadConstI = 5,**
(no Help string available)

Enum: **dssLoadConstPFixedQ = 6,**
(no Help string available)

Enum: **dssLoadConstPFixedX = 7,**
(no Help string available)

Enum: **dssLoadZIPV = 8**
(no Help string available)

enum LineUnits

Enum: **dssLineUnitsNone = 0,**
No line length unit.

Enum: **dssLineUnitsMiles = 1,**
Line length units in miles.

Enum: **dssLineUnitskFt = 2,**
Line length units are in thousand feet.

Enum: **dssLineUnitskm = 3,**

Line length units are km.

Enum: **dssLineUnitsmeter = 4,**
Line length units are meters.

Enum: **dssLineUnitsft = 5,**
Line units in feet.

Enum: **dssLineUnitsinch = 6,**
Line length units are inches.

Enum: **dssLineUnitscm = 7,**
Line units are cm.

Enum: **dssLineUnitsmm = 8,**
Line length units are mm.

Enum: **dssLineUnitsMaxnum = 9**
Maximum number of line units constants.

Interfaces

Text Interface

Property (get): **value = Command [out, retval] Type: BSTR* Command**
Input command string for the DSS.

Property (put): **Command = value [in] Type: BSTR Command**
Input command string for the DSS.

Property (get): **value = Result [out, retval] Type: BSTR* Result**
Result string for the last command.

DSSProperty Interface

Property (get): **value = Name [out, retval] Type: BSTR* Name**
Name of Property

Property (get): **value = Description [out, retval] Type: BSTR* Description**
Description of the property.

Property (get): **value = Val [out, retval] Type: BSTR* Value**
(no Help string available)

Property (put): **Val = value [in] Type: BSTR Value**
(no Help string available)

CktElement Interface

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Full Name of Active Circuit Element

Property (get): **value = NumTerminals [out, retval] Type: long* Value**
Number of Terminals this Circuit Element

Property (get): **value = NumConductors [out, retval] Type: long* Value**
Number of Conductors per Terminal

Property (get): **value = NumPhases [out, retval] Type: long* Value**
Number of Phases

Property (get): **value = BusNames [out, retval] Type: VARIANT* Value**

Variant array of strings. Get Bus definitions to which each terminal is connected. 0-based array.

Property (put): **BusNames = value [in] Type: VARIANT Value**
Variant array of strings. Set Bus definitions for each terminal is connected.

Property (get): **value = Properties [in] Type: VARIANT Indx, [out, retval] Type: IDSSProperty** Value**
Collection of Properties for this Circuit Element (0 based index, if numeric

Property (get): **value = Voltages [out, retval] Type: VARIANT* Value**
Complex array of voltages at terminals

Property (get): **value = Currents [out, retval] Type: VARIANT* Value**
Complex array of currents into each conductor of each terminal

Property (get): **value = Powers [out, retval] Type: VARIANT* Value**
Complex array of powers into each conductor of each terminal

Property (get): **value = Losses [out, retval] Type: VARIANT* Value**
Total losses in the element: two-element complex array

Property (get): **value = PhaseLosses [out, retval] Type: VARIANT* Value**
Complex array of losses by phase

Property (get): **value = SeqVoltages [out, retval] Type: VARIANT* Value**
Double array of symmetrical component voltages at each 3-phase terminal

Property (get): **value = SeqCurrents [out, retval] Type: VARIANT* Value**
Double array of symmetrical component currents into each 3-phase terminal

Property (get): **value = SeqPowers [out, retval] Type: VARIANT* Value**
Double array of sequence powers into each 3-phase teminal

Property (get): **value = Enabled [out, retval] Type: VARIANT_BOOL* Value**
Boolean indicating that element is currently in the circuit.

Property (put): **Enabled = value [in] Type: VARIANT_BOOL Value**
Boolean indicating that element is currently in the circuit.

Property (get): **value = NormalAmps [out, retval] Type: double* Value**
Normal ampere rating for PD Elements

Property (put): **NormalAmps = value [in] Type: double Value**
Normal ampere rating

Property (get): **value = EmergAmps [out, retval] Type: double* Value**
Emergency Ampere Rating for PD elements

Property (put): **EmergAmps = value [in] Type: double Value**
Emergency Ampere Rating

Method: **Open [in] Type: long Term, [in] Type: long Phs**
Open the specified terminal and phase, if non-zero. Else all conductors at terminal.

Method: **Close [in] Type: long Term, [in] Type: long Phs**
Close the specified terminal and phase, if non-zero. Else all conductors at terminal.

Method: **IsOpen [in] Type: long Term, [in] Type: long Phs, [out, retval] Type: VARIANT_BOOL* Value**
Boolean indicating if the specified terminal and, optionally, phase is open.

Property (get): **value = NumProperties [out, retval] Type: long* Value**
Number of Properties this Circuit Element.

Property (get): **value = AllPropertyNames [out, retval] Type: VARIANT* Value**
Variant array containing all property names of the active device.

Property (get): **value = Residuals [out, retval] Type: VARIANT* Value**
Residual currents for each terminal: (mag, angle

Property (get): **value = Yprim [out, retval] Type: VARIANT* Value**
YPrim matrix, column order, complex numbers (paired

Property (get): **value = DisplayName [out, retval] Type: BSTR* Value**
Display name of the object (not necessarily unique

Property (put): **DisplayName = value [in] Type: BSTR Value**
Display name of the object (not necessarily unique

Property (get): **value = Handle [out, retval] Type: long* Value**
Pointer to this object

Property (get): **value = GUID [out, retval] Type: BSTR* Value**
globally unique identifier for this object

Property (get): **value = HasSwitchControl [out, retval] Type: VARIANT_BOOL* Value**
This element has a SwtControl attached.

Property (get): **value = HasVoltControl [out, retval] Type: VARIANT_BOOL* Value**

This element has a CapControl or RegControl attached.

Property (get): **value = EnergyMeter [out, retval] Type: BSTR* Value**
Name of the Energy Meter this element is assigned to.

Property (get): **value = Controller [in] Type: long idx, [out, retval] Type: BSTR* Value**
Full name of the i-th controller attached to this element. Ex: str = Controller(2

Property (get): **value = CplxSeqVoltages [out, retval] Type: VARIANT* Value**
Complex double array of Sequence Voltage for all terminals of active circuit element.

Property (get): **value = CplxSeqCurrents [out, retval] Type: VARIANT* Value**
Complex double array of Sequence Currents for all conductors of all terminals of active circuit element.

Property (get): **value = AllVariableNames [out, retval] Type: VARIANT* Value**
Variant array of strings listing all the published variable names, if a PCElement. Otherwise, null string.

Property (get): **value = AllVariableValues [out, retval] Type: VARIANT* Value**
Variant array of doubles. Values of state variables of active element if PC element.

Property (get): **value = Variable [in] Type: BSTR MyVarName, [out] Type: long* Code, [out, retval] Type: double* Value**
For PCElement, get the value of a variable by name. If Code>0 Then no variable by this name or not a PCelement.

Property (get): **value = Variablei [in] Type: long Idx, [out] Type: long* Code, [out, retval] Type: double* Value**
For PCElement, get the value of a variable by integer index.

Property (get): **value = NodeOrder [out, retval] Type: VARIANT* Value**
Variant array of integer containing the node numbers (representing phases, for example

Property (get): **value = HasOCPDevice [out, retval] Type: VARIANT_BOOL* Value**
True if a recloser, relay, or fuse controlling this ckt element. OCP = Overcurrent Protection

Property (get): **value = NumControls [out, retval] Type: long* Value**
Number of controls connected to this device. Use to determine valid range for index into Controller array.

Property (get): **value = OCPDevIndex [out, retval] Type: long* Value**
Index into Controller list of OCP Device controlling this CktElement

Property (get): **value = OCPDevType [out, retval] Type: long* Value**
0=None; 1=Fuse; 2=Recloser; 3=Relay; Type of OCP controller device

Property (get): **value = CurrentsMagAng [out, retval] Type: VARIANT* Value**
Currents in magnitude, angle format as a variant array of doubles.

Property (get): **value = VoltagesMagAng [out, retval] Type: VARIANT* Value**
Voltages at each conductor in magnitude, angle form as variant array of doubles.

Error Interface

Property (get): **value = Number [out, retval] Type: long* Number**
Error Number

Property (get): **value = Description [out, retval] Type: BSTR* Description**
Description of error for last operation

Circuit Interface

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Name of the active circuit.

Property (get): **value = NumCktElements [out, retval] Type: long* Value**
Number of CktElements in the circuit.

Property (get): **value = NumBuses [out, retval] Type: long* Value**
Total number of Buses in the circuit.

Property (get): **value = NumNodes [out, retval] Type: long* Value**
Total number of nodes in the circuit.

Property (get): **value = Buses [in] Type: VARIANT Index, [out, retval] Type: IBus** Value**
Collection of Buses in the circuit. Index may be string or integer index (0 based

Property (get): **value = CktElements [in] Type: VARIANT Idx, [out, retval] Type: ICktElement** Value**
Collection of CktElements in Circuit

Property (get): **value = Losses [out, retval] Type: VARIANT* Value**
Total losses in active circuit, complex number (two-element array of double

Property (get): **value = LineLosses [out, retval] Type: VARIANT* Value**
Complex total line losses in the circuit

Property (get): **value = SubstationLosses [out, retval] Type: VARIANT* Value**
Complex losses in all transformers designated to substations.

Property (get): **value = TotalPower [out, retval] Type: VARIANT* Value**
Total power, watts delivered to the circuit

Property (get): **value = AllBusVolts [out, retval] Type: VARIANT* Value**
Complex array of all bus, node voltages from most recent solution

Property (get): **value = AllBusVmag [out, retval] Type: VARIANT* Value**
Array of magnitudes (doubles

Property (get): **value = AllElementNames [out, retval] Type: VARIANT* Value**
Vaiant array of strings containing Full Name of all elements.

Property (get): **value = ActiveElement [out, retval] Type: ICktElement** Value**
Return an interface to the active circuit element

Method: **Disable [in] Type: BSTR Name**
Disable a circuit element by name (removes from circuit but leave in database

Method: **Enable [in] Type: BSTR Name**
Activate (enable

Property (get): **value = Solution [out, retval] Type: ISolution** Value**
Return an interface to the Solution object.

Property (get): **value = ActiveBus [out, retval] Type: IBus** Value**
Return an interface to the active bus.

Method: **FirstPCElement [out, retval] Type: long* Value**
Sets the first Power Conversion (PC

Method: **NextPCElement [out, retval] Type: long* Value**
Gets next PC Element. Returns 0 if no more.

Method: **FirstPDElement [out, retval] Type: long* Value**
Sets the first Power Delivery (PD

Method: **NextPDElement [out, retval] Type: long* Value**
Gets next PD Element. Returns 0 if no more.

Property (get): **value = AllBusNames [out, retval] Type: VARIANT* Value**
Array of strings containing names of all buses in circuit (see AllNodeNames

Property (get): **value = AllElementLosses [out, retval] Type: VARIANT* Value**
Array of total losses (complex

Method: **Sample [void**
Force all Meters and Monitors to take a sample.

Method: **SaveSample [void**
Force all meters and monitors to save their current buffers.

Property (get): **value = Monitors [out, retval] Type: IMonitors** Value**
Returns interface to Monitors collection.

Property (get): **value = Meters [out, retval] Type: IMeters** Value**
Returns interface to Meters (EnergyMeter

Property (get): **value = Generators [out, retval] Type: IGenerators** Value**
Returns a Generators Object interface

Property (get): **value = Settings [out, retval] Type: ISettings** Value**
Returns interface to Settings interface.

Property (get): **value = Lines [out, retval] Type: ILines** Value**
Returns Interface to Lines collection.

Method: **SetActiveElement [in] Type: BSTR FullName, [out, retval] Type: long* Value**
Sets the Active Circuit Element using the full object name (e.g. \i0

Method: ***Capacity [in] Type: double Start, [in] Type: double Increment, [out, retval] Type: double* Value***
(no Help string available)

Method: **SetActiveBus [in] Type: BSTR BusName, [out, retval] Type: long* Value**
Sets Active bus by name. Ignores node list. Returns bus index (zero based

Method: **SetActiveBusi [in] Type: long BusIndex, [out, retval] Type: long* Value**
Sets ActiveBus by Integer value. 0-based index compatible with SetActiveBus return value and AllBusNames indexing. Returns 0 if OK.

Property (get): **value = AllBusVmagPu [out, retval] Type: VARIANT* Value**
Double Array of all bus voltages (each node

Property (get): **value = AllNodeNames [out, retval] Type: VARIANT* Value**
Variant array of strings containing full name of each node in system in same order as returned by

AllBusVolts, etc.

Property (get): **value = SystemY [out, retval] Type: VARIANT* Value**
System Y matrix (after a solution has been performed

Property (get): **value = CtrlQueue [out, retval] Type: ICtrlQueue** Value**
Interface to the main Control Queue

Property (get): **value = AllBusDistances [out, retval] Type: VARIANT* Value**
Returns distance from each bus to parent EnergyMeter. Corresponds to sequence in AllBusNames.

Property (get): **value = AllNodeDistances [out, retval] Type: VARIANT* Value**
Returns an array of distances from parent EnergyMeter for each Node. Corresponds to AllBusVMag sequence.

Property (get): **value = AllNodeVmagByPhase [in] Type: long Phase, [out, retval] Type: VARIANT* Value**
Returns Array of doubles represent voltage magnitudes for nodes on the specified phase.

Property (get): **value = AllNodeVmagPUByPhase [in] Type: long Phase, [out, retval] Type: VARIANT* Value**
Returns array of per unit voltage magnitudes for each node by phase

Property (get): **value = AllNodeDistancesByPhase [in] Type: long Phase, [out, retval] Type: VARIANT* Value**
Returns an array of doubles representing the distances to parent EnergyMeter. Sequence of array corresponds to other node ByPhase properties.

Property (get): **value = AllNodeNamesByPhase [in] Type: long Phase, [out, retval] Type: VARIANT* Value**
Return variant array of strings of the node names for the By Phase criteria. Sequence corresponds to other ByPhase properties.

Property (get): **value = Loads [out, retval] Type: ILoads** Value**
Returns interface to Load element interface

Method: **FirstElement [out, retval] Type: long* Value**
Sets First element of active class to be the Active element in the active circuit. Returns 0 if none.

Method: **NextElement [out, retval] Type: long* Value**
Sets the next element of the active class to be the active element in the active circuit. Returns 0 if

no more elements.

Method: **SetActiveClass [in] Type: BSTR ClassName, [out, retval] Type: long* Value**
Sets the active class by name. Use FirstElement, NextElement to iterate through the class. Returns -1 if fails.

Property (get): **value = ActiveDSSElement [out, retval] Type: IDSSElement** Value**
Returns Interface to the Active DSS object, which could be either a circuit element or a general DSS element.

Property (get): **value = ActiveCktElement [out, retval] Type: ICktElement** Value**
Returns interface to the Active Circuit element (same as ActiveElement

Property (get): **value = ActiveClass [out, retval] Type: IActiveClass** Value**
Returns interface to active class.

Property (get): **value = Transformers [out, retval] Type: ITransformers** Value**
Returns interface to Transformers collection

Property (get): **value = SwtControls [out, retval] Type: ISwtControls** Value**
Returns interface to SwtControls collection.

Property (get): **value = CapControls [out, retval] Type: ICapControls** Value**
Returns interface to CapControls collection

Property (get): **value = RegControls [out, retval] Type: IRegControls** Value**
Returns interfact to RegControls collection

Property (get): **value = Capacitors [out, retval] Type: ICapacitors** Value**
Interface to the active circuit's Capacitors collection.

Property (get): **value = Topology [out, retval] Type: ITopology** Value**
Interface to the active circuit's topology object.

Property (get): **value = Sensors [out, retval] Type: ISensors** Value**
Interface to Sensors in the Active Circuit.

Method: **UpdateStorage [void**
Forces update to all storage classes. Typically done after a solution. Done automatically in intrinsic solution modes.

Property (get): **value = ParentPDElement [out, retval] Type: long* Value**
Sets Parent PD element, if any, to be the active circuit element and returns index>0; Returns 0 if it

fails or not applicable.

Property (get): **value = XYCurves [out, retval] Type: IXYCurves** Value**
Interface to XYCurves in active circuit.

Property (get): **value = PDElements [out, retval] Type: IPDElements** Value**
Interface to PDElements collection

Property (get): **value = Reclosers [out, retval] Type: IReclosers** Value**
(no Help string available)

Property (get): **value = Relays [out, retval] Type: IRelays** Value**
(no Help string available)

Property (get): **value = LoadShapes [out, retval] Type: ILoadShapes** Value**
Interface to OpenDSS Load shapes currently defined.

Property (get): **value = Fuses [out, retval] Type: Fuses** Value**
Return interface to Fuses

Property (get): **value = Isources [out, retval] Type: IISources** Value**
Interface to ISOURCE devices

Bus Interface

Property (get): **value = Name [out, retval] Type: BSTR* Name**
Name of Bus

Property (get): **value = NumNodes [out, retval] Type: long* NumNodes**
Number of Nodes this bus.

Property (get): **value = Voltages [out, retval] Type: VARIANT* Voltages**
Complex array of voltages at this bus.

Property (get): **value = SeqVoltages [out, retval] Type: VARIANT* SeqVoltages**
Double Array of sequence voltages at this bus.

Property (get): **value = Nodes [out, retval] Type: VARIANT* Nodes**
Integer Array of Node Numbers defined at the bus in same order as the voltages.

Property (get): **value = Voc [out, retval] Type: VARIANT* Voc**
Open circuit voltage; Complex array.

Property (get): **value = Isc [out, retval] Type: VARIANT* Isc**

Short circuit currents at bus; Complex Array.

Property (get): **value = puVoltages [out, retval] Type: VARIANT* Value**
Complex Array of pu voltages at the bus.

Property (get): **value = kVBase [out, retval] Type: double* Value**
Base voltage at bus in kV

Property (get): **value = ZscMatrix [out, retval] Type: VARIANT* Value**
Complex array of Zsc matrix at bus. Column by column.

Property (get): **value = Zsc1 [out, retval] Type: VARIANT* Value**
Complex Positive-Sequence short circuit impedance at bus..

Property (get): **value = Zsc0 [out, retval] Type: VARIANT* Value**
Complex Zero-Sequence short circuit impedance at bus.

Method: **ZscRefresh [out, retval] Type: VARIANT_BOOL* Value**
Recomputes Zsc for active bus for present circuit configuration.

Property (get): **value = YscMatrix [out, retval] Type: VARIANT* Value**
Complex array of Ysc matrix at bus. Column by column.

Property (get): **value = Coorddefined [out, retval] Type: VARIANT_BOOL* Value**
False=0 else True. Indicates whether a coordinate has been defined for this bus

Property (get): **value = x [out, retval] Type: double* Value**
X Coordinate for bus (double

Property (put): **x = value [in] Type: double Value**
X Coordinate for bus (double

Property (get): **value = y [out, retval] Type: double* Value**
Y coordinate for bus(double

Property (put): **y = value [in] Type: double Value**
Y coordinate for bus(double

Property (get): **value = Distance [out, retval] Type: double* Value**
Distance from energymeter (if non-zero

Method: **GetUniqueNodeNumber [in] Type: long StartNumber, [out, retval] Type: long* Value**
Returns a unique node number at the active bus to avoid node collisions and adds it to the node

list for the bus.

Property (get): **value = CplxSeqVoltages [out, retval] Type: VARIANT* Value**
Complex Double array of Sequence Voltages (0, 1, 2

Property (get): **value = Lambda [out, retval] Type: double* Value**
Accumulated failure rate downstream from this bus; faults per year

Property (get): **value = N_interrupts [out, retval] Type: double* Value**
Number of interruptions this bus per year

Property (get): **value = Int_Duration [out, retval] Type: double* Value**
Average interruption duration, hr.

Property (get): **value = Cust_Interrupts [out, retval] Type: double* Value**
Annual number of customer-interruptions from this bus

Property (get): **value = Cust_Duration [out, retval] Type: double* Value**
Accumulated customer outage durations

Property (get): **value = N_Customers [out, retval] Type: long* Value**
Total numbers of customers served downline from this bus

Property (get): **value = VLL [out, retval] Type: VARIANT* Value**
For 2- and 3-phase buses, returns variant array of complex numbers represetin L-L voltages in volts. Returns -1.0 for 1-phase bus. If more than 3 phases, returns only first 3.

Property (get): **value = puVLL [out, retval] Type: VARIANT* Value**
Returns Complex array of pu L-L voltages for 2- and 3-phase buses. Returns -1.0 for 1-phase bus. If more than 3 phases, returns only 3 phases.

Property (get): **value = VMagAngle [out, retval] Type: VARIANT* Value**
Variant Array of doubles containing voltages in Magnitude (VLN

Property (get): **value = puVmagAngle [out, retval] Type: VARIANT* Value**
Variant array of doubles containig voltage magnitude, angle pairs in per unit

DSS Interface

Property (get): **value = NumCircuits [out, retval] Type: long* Value**
Number of Circuits currently defined

Property (get): **value = Circuits [in] Type: VARIANT Idx, [out, retval] Type: ICircuit** Value**

Collection of Circuit objects

Property (get): **value = ActiveCircuit [out, retval] Type: ICircuit** Value**
Returns interface to the active circuit.

Property (get): **value = Text [out, retval] Type: IText** Value**
Returns the DSS Text (command-result

Property (get): **value = Error [out, retval] Type: IError** Value**
Returns Error interface.

Method: **NewCircuit [in] Type: BSTR Name, [out, retval] Type: ICircuit** Value**
Make a new circuit and return interface to active circuit.

Method: **ClearAll [void**
Clears all circuit definitions.

Method: **ShowPanel [void**
Shows non-MDI child form of the Main DSS Edit Form

Method: **Start [in] Type: long code, [out, retval] Type: VARIANT_BOOL* Value**
Validate the user and start the DSS. Returns TRUE if successful.

Property (get): **value = Version [out, retval] Type: BSTR* Value**
Get version string for the DSS.

Property (get): **value = DSSProgress [out, retval] Type: IDSSProgress** Value**
Gets interface to the DSS Progress Meter

Property (get): **value = Classes [out, retval] Type: VARIANT* Value**
List of DSS intrinsic classes (names of the classes

Property (get): **value = UserClasses [out, retval] Type: VARIANT* Value**
List of user-defined classes

Property (get): **value = NumClasses [out, retval] Type: long* Value**
Number of DSS intrinsic classes

Property (get): **value = NumUserClasses [out, retval] Type: long* Value**
Number of user-defined classes

Property (get): **value = DataPath [out, retval] Type: BSTR* Value**
DSS Data File Path. Default path for reports, etc. from DSS

Property (put): **DataPath = value [in] Type: BSTR Value**

DSS Data File Path. Default path for reports, etc. from DSS

Method: **Reset [void**
Resets DSS Initialization for restarts, etc from applets

Property (get): **value = AllowForms [out, retval] Type: VARIANT_BOOL* Value**
Default is TRUE. Use this to set to FALSE; Cannot reset to TRUE;

Property (put): **AllowForms = value [in] Type: VARIANT_BOOL Value**
Default is TRUE. Use this to set to FALSE; Cannot reset to TRUE;

Property (get): **value = DefaultEditor [out, retval] Type: BSTR* Value**
Returns the path name for the default text editor.

Property (get): **value = ActiveClass [out, retval] Type: IActiveClass** Value**
Returns interface to the active class.

Method: **SetActiveClass [in] Type: BSTR ClassName, [out, retval] Type: long* Value**
Sets the Active DSS Class for use with ActiveClass interface. Same as SetActiveClass in Circuit interface.

Property (get): **value = Executive [out, retval] Type: IDSS_Executive** Value**
Interface to DSS Executive commands and options

Property (get): **value = Events [out, retval] Type: IDSSEvents** Value**
Interface to the DSS Events

Property (get): **value = CmathLib [out, retval] Type: ICmathLib** Value**
Returns an interface to the complex math library.

Property (get): **value = Parser [out, retval] Type: IParser** Value**
Returns interface to the OpenDSS Parser library for use by user-written programs.

Solution Interface

Method: **Solve [void**
Execute solution for present solution mode.

Property (get): **value = Mode [out, retval] Type: long* Mode**
Set present solution mode (by a text code - see DSS Help

Property (put): **Mode = value [in] Type: long Mode**
Set present solution mode (by a text code - see DSS Help

Property (get): **value = Frequency [out, retval] Type: double* Frequency**
Set the Frequency for next solution

Property (put): **Frequency = value [in] Type: double Frequency**
Set the Frequency for next solution

Property (get): **value = Hour [out, retval] Type: long* Hour**
Set Hour for time series solutions.

Property (put): **Hour = value [in] Type: long Hour**
Set Hour for time series solutions.

Property (get): **value = Seconds [out, retval] Type: double* Seconds**
Seconds from top of the hour.

Property (put): **Seconds = value [in] Type: double Seconds**
Seconds from top of the hour.

Property (get): **value = StepSize [out, retval] Type: double* StepSize**
Time step size in sec

Property (put): **StepSize = value [in] Type: double StepSize**
Time step size in sec

Property (get): **value = Year [out, retval] Type: long* Year**
Set year for planning studies

Property (put): **Year = value [in] Type: long Year**
Set year for planning studies

Property (get): **value = LoadMult [out, retval] Type: double* LoadMult**
Default load multiplier applied to all non-fixed loads

Property (put): **LoadMult = value [in] Type: double LoadMult**
Default load multiplier applied to all non-fixed loads

Property (get): **value = Iterations [out, retval] Type: long* Iterations**
Number of iterations taken for last solution. (Same as TotalIterations

Property (get): **value = MaxIterations [out, retval] Type: long* MaxIterations**
Max allowable iterations.

Property (put): **MaxIterations = value [in] Type: long MaxIterations**
Max allowable iterations.

Property (get): **value = Tolerance [out, retval] Type: double* Tolerance**
Solution convergence tolerance.

Property (put): **Tolerance = value [in] Type: double Tolerance**
Solution convergence tolerance.

Property (get): **value = Number [out, retval] Type: long* Number**
Number of solutions to perform for Monte Carlo and time series simulations

Property (put): **Number = value [in] Type: long Number**
Number of solutions to perform for Monte Carlo and time series simulations

Property (get): **value = Random [out, retval] Type: long* Random**
Randomization mode for random variables \i0

*Property (put): **Random = value [in] Type: long Random***
Randomization mode for random variables \i0

*Property (get): **value = ModeID [out, retval] Type: BSTR* Value***
ID (text

Property (get): **value = LoadModel [out, retval] Type: long* Value**
Load Model: dssPowerFlow (default

Property (put): **LoadModel = value [in] Type: long Value**
Load Model: dssPowerFlow (default

Property (get): **value = LDCurve [out, retval] Type: BSTR* Value**
Load-Duration Curve name for LD modes

Property (put): **LDCurve = value [in] Type: BSTR Value**
Load-Duration Curve name for LD modes

Property (get): **value = pctGrowth [out, retval] Type: double* Value**
Percent default annual load growth rate

Property (put): **pctGrowth = value [in] Type: double Value**
Percent default annual load growth rate

Property (get): **value = AddType [out, retval] Type: long* Value**
Type of device to add in AutoAdd Mode: dssGen (Default

Property (put): **AddType = value [in] Type: long Value**
Type of device to add in AutoAdd Mode: dssGen (Default

Property (get): **value = GenkW [out, retval] Type: double* Value**
Generator kW for AutoAdd mode

Property (put): **GenkW = value [in] Type: double Value**
Generator kW for AutoAdd mode

Property (get): **value = GenPF [out, retval] Type: double* Value**
PF for generators in AutoAdd mode

Property (put): **GenPF = value [in] Type: double Value**
PF for generators in AutoAdd mode

Property (get): **value = Capkvar [out, retval] Type: double* Value**
Capacitor kvar for adding capacitors in AutoAdd mode

Property (put): **Capkvar = value [in] Type: double Value**
Capacitor kvar for adding capacitors in AutoAdd mode

Property (get): **value = Algorithm [out, retval] Type: long* Value**
Base Solution algorithm: dssNormalSolve | dssNewtonSolve

Property (put): **Algorithm = value [in] Type: long Value**
Base Solution algorithm: dssNormalSolve | dssNewtonSolve

Property (get): **value = ControlMode [out, retval] Type: long* Value**
dssStatic | dssEvent | dssTime Modes for control devices*

Property (put): **ControlMode = value [in] Type: long Value**
dssStatic | dssEvent | dssTime Modes for control devices*

Property (get): **value = GenMult [out, retval] Type: double* Value**
Default Multiplier applied to generators (like LoadMult

Property (put): **GenMult = value [in] Type: double Value**
Default Multiplier applied to generators (like LoadMult

Property (get): **value = DefaultDaily [out, retval] Type: BSTR* Value**
Default daily load shape (defaults to \i0

*Property (put): **DefaultDaily = value [in] Type: BSTR Value***
Default daily load shape (defaults to \i0

*Property (get): **value = DefaultYearly [out, retval] Type: BSTR* Value***
Default Yearly load shape (defaults to \i0

Property (put): ***DefaultYearly = value [in] Type: BSTR Value***
Default Yearly load shape (defaults to \i0

Property (get): ***value = EventLog [out, retval] Type: VARIANT* Value***
Array of strings containing the Event Log

Property (get): **value = dblHour [out, retval] Type: double* Value**
Hour as a double, including fractional part

Property (put): **dblHour = value [in] Type: double Value**
Hour as a double, including fractional part

Property (put): **StepsizeMin = value [in] Type: double Param1**
Set Stepsize in minutes

Property (put): **StepsizeHr = value [in] Type: double Param1**
Set Stepsize in Hr

Property (get): **value = ControlIterations [out, retval] Type: long* Value**
Value of the control iteration counter

Property (put): **ControlIterations = value [in] Type: long Value**
Value of the control iteration counter

Property (get): **value = MaxControlIterations [out, retval] Type: long* Value**
Maximum allowable control iterations

Property (put): **MaxControlIterations = value [in] Type: long Value**
Maximum allowable control iterations

Method: **Sample_DoControlActions [void**
Sample controls and then process the control queue for present control mode and dispatch control actions

Method: **CheckFaultStatus [void**
Executes status check on all fault objects defined in the circuit.

Method: **SolveSnap [void**
Execute the snapshot power flow routine in the DSS that solves at the present state with control actions

Method: **SolveDirect [void**
Executes a direct solution from the system Y matrix, ignoring compensation currents of loads,

generators (includes Yprim only

Method: **SolvePflow [void**
Solves using present power flow method. Iterative solution rather than direct solution.

Method: **SolveNoControl [void**
Similar to SolveSnap except no control actions are checked or executed

Method: **SolvePlusControl [void**
Executes a power flow solution (SolveNoControl

Method: **InitSnap [void**
Initializes some variables for snap shot power flow. SolveSnap does this automatically.

Method: **CheckControls [void**
The normal process for sampling and executing Control Actions and Fault Status and rebuilds Y if necessary.

Method: **SampleControlDevices [void**
Executes a sampling of all intrinsic control devices, which push control actions onto the control queue.

Method: **DoControlActions [void**
Pops control actions off the control queue and dispatches to the proper control element

Method: **BuildYMatrix [in] Type: long BuildOption, [in] Type: long AllocateVI**
Force building of the System Y matrix

Property (get): **value = SystemYChanged [out, retval] Type: VARIANT_BOOL* Value**
Flag that indicates if elements of the System Y have been changed by recent activity.

Property (get): **value = Converged [out, retval] Type: VARIANT_BOOL* Value**
Flag to indicate whether the circuit solution converged

Property (put): **Converged = value [in] Type: VARIANT_BOOL Value**
Flag to indicate whether the circuit solution converged

Property (get): **value = Totaliterations [out, retval] Type: long* Value**
Total iterations including control iterations for most recent solution.

Property (get): **value = MostIterationsDone [out, retval] Type: long* Value**
Max number of iterations required to converge at any control iteration of the most recent solution.

Property (get): **value = ControlActionsDone [out, retval] Type: VARIANT_BOOL* Value**
Flag indicating the control actions are done.

Property (put): **ControlActionsDone = value [in] Type: VARIANT_BOOL Value**
(no Help string available)

Monitors Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Array of all Monitor Names

Property (get): **value = First [out, retval] Type: long* Value**
Sets the first Monitor active. Returns 0 if no monitors.

Property (get): **value = Next [out, retval] Type: long* Value**
Sets next monitor active. Returns 0 if no more.

Method: **Reset [void**
Resets active Monitor object.

Method: **ResetAll [void**
Resets all Monitor Objects

Method: **Sample [void**
Causes active Monitor to take a sample.

Method: **Save [void**
Causes active monitor to save its current sample buffer to its monitor stream. Then you can access the Bytestream or channel data. Most standard solution modes do this automatically.

Method: **Show [void**
Converts monitor file to text and displays with text editor

Property (get): **value = FileName [out, retval] Type: BSTR* Value**
Name of CSV file associated with active Monitor.

Property (get): **value = Mode [out, retval] Type: long* Value**
Set Monitor mode (bitmask integer - see DSS Help

Property (put): **Mode = value [in] Type: long Value**
Set Monitor mode (bitmask integer - see DSS Help

Property (get): **value = Name [out, retval] Type: BSTR* Value**

Sets the active Monitor object by name

Property (put): **Name = value [in] Type: BSTR Value**
Sets the active Monitor object by name

Property (get): **value = ByteStream [out, retval] Type: VARIANT* Value**
Byte Array containing monitor stream values. Make sure a \i0

*Property (get): **value = SampleCount [out, retval] Type: long* Value***
Number of Samples in Monitor at Present

Method: **SampleAll [void**
Causes all Monitors to take a sample of the present state

Method: **SaveAll [void**
Save all Monitor buffers to their respective file streams.

Property (get): **value = Count [out, retval] Type: long* Value**
Number of Monitors

Method: **Process [void**
Post-process monitor samples taken so far, e.g., Pst for mode=4

Method: **ProcessAll [void**
All monitors post-process the data taken so far.

Property (get): **value = FileVersion [out, retval] Type: long* Value**
Monitor File Version (integer

Property (get): **value = RecordSize [out, retval] Type: long* Value**
Size of each record in ByteStream (Integer

Property (get): **value = Header [out, retval] Type: VARIANT* Value**
Header string; Variant array of strings containing Channel names

Property (get): **value = dblHour [out, retval] Type: VARIANT* Value**
Variant array of doubles containgin time value in hours for time-sampled monitor values; Empty if frequency-sampled values for harmonics solution (see dblFreq

Property (get): **value = dblFreq [out, retval] Type: VARIANT* Value**
Variant array of doubles containing frequency values for harmonics mode solutions; Empty for time mode solutions (use dblHour

Property (get): **value = Channel [in] Type: long Index, [out, retval] Type: VARIANT* Value**

Variant array of doubles for the specified channel (usage: MyArray = DSSMonitor.Channel(i

Property (get): **value = NumChannels [out, retval] Type: long* Value**
Number of Channels in the active Monitor

Meters Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Array of all energy Meter names

Property (get): **value = First [out, retval] Type: long* Value**
Set the first energy Meter active. Returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Sets the next energy Meter active. Returns 0 if no more.

Property (get): **value = RegisterNames [out, retval] Type: VARIANT* Value**
Array of strings containing the names of the registers.

Property (get): **value = RegisterValues [out, retval] Type: VARIANT* Value**
Array of all the values contained in the Meter registers for the active Meter.

Method: **Reset [void**
Resets registers of active Meter.

Method: **ResetAll [void**
Resets registers of all Meter objects.

Method: **Sample [void**
Forces active Meter to take a sample.

Method: **Save [void**
Saves meter register values.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Get/Set the active meter name.

Property (put): **Name = value [in] Type: BSTR Value**
Set a meter to be active by name.

Property (get): **value = Totals [out, retval] Type: VARIANT* Value**
Totals of all registers of all meters

Property (get): **value = Peakcurrent [out, retval] Type: VARIANT* Value**

Array of doubles to set values of Peak Current property

Property (put): **Peakcurrent = value [in] Type: VARIANT Value**
Array of doubles to set values of Peak Current property

Property (get): **value = CalcCurrent [out, retval] Type: VARIANT* Value**
Set the magnitude of the real part of the Calculated Current (normally determined by solution

Property (put): **CalcCurrent = value [in] Type: VARIANT Value**
Set the magnitude of the real part of the Calculated Current (normally determined by solution

Property (get): **value = AllocFactors [out, retval] Type: VARIANT* Value**
Array of doubles: set the phase allocation factors for the active meter.

Property (put): **AllocFactors = value [in] Type: VARIANT Value**
Array of doubles: set the phase allocation factors for the active meter.

Property (get): **value = MeteredElement [out, retval] Type: BSTR* Value**
Set Name of metered element

Property (put): **MeteredElement = value [in] Type: BSTR Value**
Set Name of metered element

Property (get): **value = MeteredTerminal [out, retval] Type: long* Value**
set Number of Metered Terminal

Property (put): **MeteredTerminal = value [in] Type: long Value**
set Number of Metered Terminal

Property (get): **value = DIFilesAreOpen [out, retval] Type: VARIANT_BOOL* Value**
Global Flag in the DSS to indicate if Demand Interval (DI

Method: **SampleAll [void**
Causes all EnergyMeter objects to take a sample at the present time

Method: **SaveAll [void**
Save All EnergyMeter objects

Method: **OpenAllDIFiles [void**
Open Demand Interval (DI

Method: **CloseAllDIFiles [void**
Close All Demand Interval Files (Necessary at the end of a run

Property (get): **value = CountEndElements [out, retval] Type: long* Value**

Number of zone end elements in the active meter zone.

Property (get): **value = AllEndElements [out, retval] Type: VARIANT* Value**
Variant array of names of all zone end elements.

Property (get): **value = Count [out, retval] Type: long* Value**
Number of Energy Meters in the Active Circuit

Property (get): **value = AllBranchesInZone [out, retval] Type: VARIANT* Value**
Wide string list of all branches in zone of the active energymeter object.

Property (get): **value = CountBranches [out, retval] Type: long* Value**
Number of branches in Active energymeter zone. (Same as sequencelist size

Property (get): **value = SAIFI [out, retval] Type: double* Value**
Returns SAIFI for this meter's Zone. Execute Reliability Calc method first.

Property (get): **value = SequenceIndex [out, retval] Type: long* Value**
Get/set Index into Meter's SequenceList that contains branch pointers in lexical order. Earlier index guaranteed to be upline from later index. Sets PDelement active.

Property (put): **SequenceIndex = value [in] Type: long Value**
Get/set Index into Meter's SequenceList that contains branch pointers in lexical order. Earlier index guaranteed to be upline from later index. Sets PDelement active.

Property (get): **value = SAIFIKW [out, retval] Type: double* Value**
SAIFI based on kW rather than number of customers. Get after reliability calcs.

Method: **DoReliabilityCalc [void**
Calculate SAIFI, etc.

Property (get): **value = SeqListSize [out, retval] Type: long* Value**
Size of Sequence List

Generators Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Array of names of all Generator objects.

Property (get): **value = RegisterNames [out, retval] Type: VARIANT* Value**
Array of Names of all generator energy meter registers

Property (get): **value = RegisterValues [out, retval] Type: VARIANT* Value**

Array of valus in generator energy meter registers.

Property (get): **value = First [out, retval] Type: long* Value**
Sets first Generator to be active. Returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Sets next Generator to be active. Returns 0 if no more.

Property (get): **value = ForcedON [out, retval] Type: VARIANT_BOOL* Value**
Indicates whether the generator is forced ON regardles of other dispatch criteria.

Property (put): **ForcedON = value [in] Type: VARIANT_BOOL Value**
Indicates whether the generator is forced ON regardles of other dispatch criteria.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Sets a generator active by name.

Property (put): **Name = value [in] Type: BSTR Value**
Sets a generator active by name.

Property (get): **value = kV [out, retval] Type: double* Value**
Voltage base for the active generator, kV

Property (put): **kV = value [in] Type: double Value**
Voltage base for the active generator, kV

Property (get): **value = kW [out, retval] Type: double* Value**
kW output for the active generator. kvar is updated for current power factor.

Property (put): **kW = value [in] Type: double Value**
kW output for the active generator. kvar is updated for current power factor

Property (get): **value = kvar [out, retval] Type: double* Value**
kvar output for the active generator. Updates power factor based on present kW value.

Property (put): **kvar = value [in] Type: double Value**
kvar output for the active generator. Updates power factor based on present kW.

Property (get): **value = PF [out, retval] Type: double* Value**
Power factor (pos. = producing vars

Property (put): **PF = value [in] Type: double Value**
Power factor (pos. = producing vars

Property (get): **value = Phases [out, retval] Type: long* Value**

Number of phases

Property (put): **Phases = value [in] Type: long Value**
Number of phases

Property (get): **value = Count [out, retval] Type: long* Value**
Number of Generator Objects in Active Circuit

Property (get): **value = idx [out, retval] Type: long* Value**
Get/Set active Generator by index into generators list. 1..Count

Property (put): **idx = value [in] Type: long Value**
Get/Set active Generator by index into generators list. 1..Count

DSSProgress Interface

Property (put): **PctProgress = value [in] Type: long Param1**
Percent progress to indicate [0..100]

Property (put): **Caption = value [in] Type: BSTR Param1**
Caption to appear on the bottom of the DSS Progress form.

Method: **Show [void**
Shows progress form with null caption and progress set to zero.

Method: **Close [void**
Closes (hides

Settings Interface

Property (get): **value = AllowDuplicates [out, retval] Type: VARIANT_BOOL* Value**
True | False Designates whether to allow duplicate names of objects*

Property (put): **AllowDuplicates = value [in] Type: VARIANT_BOOL Value**
True | False Designates whether to allow duplicate names of objects*

Property (get): **value = ZoneLock [out, retval] Type: VARIANT_BOOL* Value**
True | False Locks Zones on energy meters to prevent rebuilding if a circuit change occurs.*

Property (put): **ZoneLock = value [in] Type: VARIANT_BOOL Value**
True | False Locks Zones on energy meters to prevent rebuilding if a circuit change occurs.*

Property (put): **AllocationFactors = value [in] Type: double Param1**

Sets all load allocation factors for all loads defined by XFKVA property to this value.

Property (get): **value = AutoBusList [out, retval] Type: BSTR* Value**
List of Buses or (File=xxxx

Property (put): **AutoBusList = value [in] Type: BSTR Value**
List of Buses or (File=xxxx

Property (get): **value = CktModel [out, retval] Type: long* Value**
*dssMultiphase * | dssPositiveSeq IIndicate if the circuit model is positive sequence.*

Property (put): **CktModel = value [in] Type: long Value**
*dssMultiphase * | dssPositiveSeq IIndicate if the circuit model is positive sequence.*

Property (get): **value = NormVminpu [out, retval] Type: double* Value**
Per Unit minimum voltage for Normal conditions.

Property (put): **NormVminpu = value [in] Type: double Value**
Per Unit minimum voltage for Normal conditions.

Property (get): **value = NormVmaxpu [out, retval] Type: double* Value**
Per Unit maximum voltage for Normal conditions.

Property (put): **NormVmaxpu = value [in] Type: double Value**
Per Unit maximum voltage for Normal conditions.

Property (get): **value = EmergVminpu [out, retval] Type: double* Value**
Per Unit minimum voltage for Emergency conditions.

Property (put): **EmergVminpu = value [in] Type: double Value**
Per Unit minimum voltage for Emergency conditions.

Property (get): **value = EmergVmaxpu [out, retval] Type: double* Value**
Per Unit maximum voltage for Emergency conditions.

Property (put): **EmergVmaxpu = value [in] Type: double Value**
Per Unit maximum voltage for Emergency conditions.

Property (get): **value = UEweight [out, retval] Type: double* Value**
Weighting factor applied to UE register values.

Property (put): **UEweight = value [in] Type: double Value**
Weighting factor applied to UE register values.

Property (get): **value = LossWeight [out, retval] Type: double* Value**

Weighting factor applied to Loss register values.

Property (put): **LossWeight = value [in] Type: double Value**
Weighting factor applied to Loss register values.

Property (get): **value = UEregs [out, retval] Type: VARIANT* Value**
Array of Integers defining energy meter registers to use for computing UE

Property (put): **UEregs = value [in] Type: VARIANT Value**
Array of Integers defining energy meter registers to use for computing UE

Property (get): **value = LossRegs [out, retval] Type: VARIANT* Value**
Integer array defining which energy meter registers to use for computing losses

Property (put): **LossRegs = value [in] Type: VARIANT Value**
Integer array defining which energy meter registers to use for computing losses

Property (get): **value = Trapezoidal [out, retval] Type: VARIANT_BOOL* Value**
*True | False * Gets value of trapezoidal integration flag in energy meters.*

Property (put): **Trapezoidal = value [in] Type: VARIANT_BOOL Value**
*True | False * Gets value of trapezoidal integration flag in energy meters.*

Property (get): **value = VoltageBases [out, retval] Type: VARIANT* Value**
Array of doubles defining the legal voltage bases in kV L-L

Property (put): **VoltageBases = value [in] Type: VARIANT Value**
Array of doubles defining the legal voltage bases in kV L-L

Property (get): **value = ControlTrace [out, retval] Type: VARIANT_BOOL* Value**
True | False Denotes whether to trace the control actions to a file.*

Property (put): **ControlTrace = value [in] Type: VARIANT_BOOL Value**
True | False Denotes whether to trace the control actions to a file.*

Property (get): **value = PriceSignal [out, retval] Type: double* Value**
Price Signal for the Circuit

Property (put): **PriceSignal = value [in] Type: double Value**
Price Signal for the Circuit

Property (get): **value = PriceCurve [out, retval] Type: BSTR* Value**
Name of LoadShape object that serves as the source of price signal data for yearly simulations, etc.

Property (put): **PriceCurve = value [in] Type: BSTR Value**
Name of LoadShape object that serves as the source of price signal data for yearly simulations, etc.

Lines Interface

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Specify the name of the Line element to set it active.

Property (put): **Name = value [in] Type: BSTR Value**
Specify the name of the Line element to set it active.

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Names of all Line Objects

Property (get): **value = First [out, retval] Type: long* Value**
Invoking this property sets the first element active. Returns 0 if no lines. Otherwise, index of the line element.

Property (get): **value = Next [out, retval] Type: long* Value**
Invoking this property advances to the next Line element active. Returns 0 if no more lines. Otherwise, index of the line element.

Method: **New [in] Type: BSTR Name, [out, retval] Type: long* Value**
Creates a new Line and makes it the Active Circuit Element.

Property (get): **value = Bus1 [out, retval] Type: BSTR* Value**
Name of bus for terminal 1.

Property (put): **Bus1 = value [in] Type: BSTR Value**
Name of bus for terminal 1.

Property (get): **value = Bus2 [out, retval] Type: BSTR* Value**
Name of bus for terminal 2.

Property (put): **Bus2 = value [in] Type: BSTR Value**
Name of bus for terminal 2.

Property (get): **value = LineCode [out, retval] Type: BSTR* Value**
Name of LineCode object that defines the impedances.

Property (put): **LineCode = value [in] Type: BSTR Value**
Name of LineCode object that defines the impedances.

Property (get): **value = Length [out, retval] Type: double* Value**
Length of line section in units compatible with the LineCode definition.

Property (put): **Length = value [in] Type: double Value**
Length of line section in units compatible with the LineCode definition.

Property (get): **value = Phases [out, retval] Type: long* Value**
Number of Phases, this Line element.

Property (put): **Phases = value [in] Type: long Value**
Number of Phases, this Line element.

Property (get): **value = R1 [out, retval] Type: double* Value**
Positive Sequence resistance, ohms per unit length.

Property (put): **R1 = value [in] Type: double Value**
Positive Sequence resistance, ohms per unit length.

Property (get): **value = X1 [out, retval] Type: double* Value**
Positive Sequence reactance, ohms per unit length.

Property (put): **X1 = value [in] Type: double Value**
Positive Sequence reactance, ohms per unit length.

Property (get): **value = R0 [out, retval] Type: double* Value**
Zero Sequence resistance, ohms per unit length.

Property (put): **R0 = value [in] Type: double Value**
Zero Sequence resistance, ohms per unit length.

Property (get): **value = X0 [out, retval] Type: double* Value**
Zero Sequence reactance ohms per unit length.

Property (put): **X0 = value [in] Type: double Value**
Zero Sequence reactance ohms per unit length.

Property (get): **value = C1 [out, retval] Type: double* Value**
Positive Sequence capacitance, nanofarads per unit length.

Property (put): **C1 = value [in] Type: double Value**
Positive Sequence capacitance, nanofarads per unit length.

Property (get): **value = C0 [out, retval] Type: double* Value**
Zero Sequence capacitance, nanofarads per unit length.

Property (put): **C0 = value [in] Type: double Value**
Zero Sequence capacitance, nanofarads per unit length.

Property (get): **value = Rmatrix [out, retval] Type: VARIANT* Value**
Resistance matrix (full

Property (put): **Rmatrix = value [in] Type: VARIANT Value**
Resistance matrix (full

Property (get): **value = Xmatrix [out, retval] Type: VARIANT* Value**
(no Help string available)

Property (put): **Xmatrix = value [in] Type: VARIANT Value**
(no Help string available)

Property (get): **value = Cmatrix [out, retval] Type: VARIANT* Value**
(no Help string available)

Property (put): **Cmatrix = value [in] Type: VARIANT Value**
(no Help string available)

Property (get): **value = NormAmps [out, retval] Type: double* Value**
Normal ampere rating of Line.

Property (put): **NormAmps = value [in] Type: double Value**
Normal ampere rating of Line.

Property (get): **value = EmergAmps [out, retval] Type: double* Value**
Emergency (maximum

Property (put): **EmergAmps = value [in] Type: double Value**
Emergency (maximum

Property (get): **value = Geometry [out, retval] Type: BSTR* Value**
Line geometry code

Property (put): **Geometry = value [in] Type: BSTR Value**
Line geometry code

Property (get): **value = Rg [out, retval] Type: double* Value**
Earth return resistance value used to compute line impedances at power frequency

Property (put): **Rg = value [in] Type: double Value**
Earth return resistance value used to compute line impedances at power frequency

Property (get): **value = Xg [out, retval] Type: double* Value**
Earth return reactance value used to compute line impedances at power frequency

Property (put): **Xg = value [in] Type: double Value**
Earth return reactance value used to compute line impedances at power frequency

Property (get): **value = Rho [out, retval] Type: double* Value**
Earth Resistivity, m-ohms

Property (put): **Rho = value [in] Type: double Value**
Earth Resistivity, m-ohms

Property (get): **value = Yprim [out, retval] Type: VARIANT* Value**
Yprimitive: Does Nothing at present on Put; Dangerous

Property (put): **Yprim = value [in] Type: VARIANT Value**
Yprimitive: Does Nothing at present on Put; Dangerous

Property (get): **value = NumCust [out, retval] Type: long* Value**
Number of customers on this line section.

Property (get): **value = TotalCust [out, retval] Type: long* Value**
Total Number of customers served from this line section.

Property (get): **value = Parent [out, retval] Type: long* Value**
Sets Parent of the active Line to be the active line. Returns 0 if no parent or action fails.

Property (get): **value = Count [out, retval] Type: long* Value**
Number of Line objects in Active Circuit.

Property (get): **value = Spacing [out, retval] Type: BSTR* Value**
Line spacing code

Property (put): **Spacing = value [in] Type: BSTR Value**
Line spacing code

Property (get): **value = Units [out, retval] Type: long* Value**
(no Help string available)

Property (put): **Units = value [in] Type: long Value**
(no Help string available)

CtrlQueue Interface

Method: **ClearQueue [void**
Clear control queue

Method: **Delete [in] Type: long ActionHandle**
Delete a control action from the DSS control queue by referencing the handle of the action

Property (get): **value = NumActions [out, retval] Type: long* Value**
Number of Actions on the current actionlist (that have been popped off the control queue by CheckControlActions

Property (put): **Action = value [in] Type: long Param1**
Set the active action by index

Property (get): **value = ActionCode [out, retval] Type: long* Value**
Code for the active action. Long integer code to tell the control device what to do

Property (get): **value = DeviceHandle [out, retval] Type: long* Value**
Handle (User defined

Method: **Push [in] Type: long Hour, [in] Type: double Seconds, [in] Type: long ActionCode, [in] Type: long DeviceHandle, [out, retval] Type: long* Value**
Push a control action onto the DSS control queue by time, action code, and device handle (user defined

Method: **Show [void**
Show entire control queue in CSV format

Method: **ClearActions [void**
Clear the Action list.

Property (get): **value = PopAction [out, retval] Type: long* Value**
Pops next action off the action list and makes it the active action. Returns Number of actions remaining.

Loads Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings containing all Load names

Property (get): **value = First [out, retval] Type: long* Value**
Set first Load element to be active; returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**

Sets next Load element to be active; returns 0 of none else index of active load.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Set active load by name.

Property (put): **Name = value [in] Type: BSTR Value**
Set active load by name.

Property (get): **value = Idx [out, retval] Type: long* Value**
Sets active load by index into load list. 1..Count

Property (put): **Idx = value [in] Type: long Value**
Sets active load by index into load list. 1..Count

Property (get): **value = kW [out, retval] Type: double* Value**
Set kW for active Load. Updates kvar based on present PF.

Property (put): **kW = value [in] Type: double Value**
Set kW for active Load. Updates kvar based on present PF.

Property (get): **value = kV [out, retval] Type: double* Value**
Set kV rating for active Load. For 2 or more phases set Line-Line kV. Else actual kV across terminals.

Property (put): **kV = value [in] Type: double Value**
Set kV rating for active Load. For 2 or more phases set Line-Line kV. Else actual kV across terminals.

Property (get): **value = kvar [out, retval] Type: double* Value**
Set kvar for active Load. Updates PF based in present kW.

Property (put): **kvar = value [in] Type: double Value**
Set kvar for active Load. Updates PF based on present kW.

Property (get): **value = PF [out, retval] Type: double* Value**
Set Power Factor for Active Load. Specify leading PF as negative. Updates kvar based on kW value

Property (put): **PF = value [in] Type: double Value**
Set Power Factor for Active Load. Specify leading PF as negative. Updates kvar based on present value of kW.

Property (get): **value = Count [out, retval] Type: long* Value**

Number of Load objects in active circuit.

Property (get): **value = PctMean [out, retval] Type: double* Value**
Average percent of nominal load in Monte Carlo studies; only if no loadshape defined for this load.

Property (put): **PctMean = value [in] Type: double Value**
(no Help string available)

Property (get): **value = PctStdDev [out, retval] Type: double* Value**
Percent standard deviation for Monte Carlo load studies; if there is no loadshape assigned to this load.

Property (put): **PctStdDev = value [in] Type: double Value**
(no Help string available)

Property (get): **value = AllocationFactor [out, retval] Type: double* Value**
Factor for allocating loads by connected xfkva

Property (put): **AllocationFactor = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Cfactor [out, retval] Type: double* Value**
Factor relates average to peak kw. Used for allocation with kwh and kwhdays/

Property (put): **Cfactor = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Class [out, retval] Type: long* Value**
A code number used to separate loads by class or group. No effect on the solution.

Property (put): **Class = value [in] Type: long Value**
(no Help string available)

Property (get): **value = IsDelta [out, retval] Type: VARIANT_BOOL* Value**
Delta loads are connected line-to-line.

Property (put): **IsDelta = value [in] Type: VARIANT_BOOL Value**
(no Help string available)

Property (get): **value = CVRcurve [out, retval] Type: BSTR* Value**
Name of a loadshape with both Mult and Qmult, for CVR factors as a function of time.

Property (put): **CVRcurve = value [in] Type: BSTR Value**

(no Help string available)

Property (get): **value = CVRwatts [out, retval] Type: double* Value**
Percent reduction in P for percent reduction in V. Must be used with dssLoadModelCVR.

Property (put): **CVRwatts = value [in] Type: double Value**
(no Help string available)

Property (get): **value = CVRvars [out, retval] Type: double* Value**
Percent reduction in Q for percent reduction in V. Must be used with dssLoadModelCVR.

Property (put): **CVRvars = value [in] Type: double Value**
(no Help string available)

Property (get): **value = daily [out, retval] Type: BSTR* Value**
Name of the loadshape for a daily load profile.

Property (put): **daily = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = duty [out, retval] Type: BSTR* Value**
Name of the loadshape for a duty cycle simulation.

Property (put): **duty = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = kva [out, retval] Type: double* Value**
Base load kva. Also defined kw and kvar or pf input, or load allocation by kwh or xfkva.

Property (put): **kva = value [in] Type: double Value**
(no Help string available)

Property (get): **value = kwh [out, retval] Type: double* Value**
kwh billed for this period. Can be used with Cfactor for load allocation.

Property (put): **kwh = value [in] Type: double Value**
(no Help string available)

Property (get): **value = kwhdays [out, retval] Type: double* Value**
Length of kwh billing period for average demand calculation. Default 30.

Property (put): **kwhdays = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Model [out, retval] Type: enum LoadModels*, [Value**

The Load Model defines variation of P and Q with voltage.

Property (put): **Model = value [in] Type: enum LoadModels, [Value**
(no Help string available)

Property (get): **value = NumCust [out, retval] Type: long* Value**
Number of customers in this load, defaults to one.

Property (put): **NumCust = value [in] Type: long Value**
(no Help string available)

Property (get): **value = Rneut [out, retval] Type: double* Value**
Neutral resistance for wye-connected loads.

Property (put): **Rneut = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Spectrum [out, retval] Type: BSTR* Value**
Name of harmonic current spectrrum shape.

Property (put): **Spectrum = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = Vmaxpu [out, retval] Type: double* Value**
Maximum per-unit voltage to use the load model. Above this, constant Z applies.

Property (put): **Vmaxpu = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Vminemerg [out, retval] Type: double* Value**
Minimum voltage for unserved energy (UE

Property (put): **Vminemerg = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Vminnorm [out, retval] Type: double* Value**
Minimum voltage for energy exceeding normal (EEN

Property (put): **Vminnorm = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Vminpu [out, retval] Type: double* Value**
Minimum voltage to apply the load model. Below this, constant Z is used.

Property (put): **Vminpu = value [in] Type: double Value**

(no Help string available)

Property (get): **value = xfkVA [out, retval] Type: double* Value**
Rated service transformer kVA for load allocation, using AllocationFactor. Affects kW, kvar, and pf.

Property (put): **xfkVA = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Xneut [out, retval] Type: double* Value**
Neutral reactance for wye-connected loads.

Property (put): **Xneut = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Yearly [out, retval] Type: BSTR* Value**
Name of yearly duration loadshape

Property (put): **Yearly = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = Status [out, retval] Type: enum LoadStatus*, [Value**
Response to load multipliers: Fixed (growth only

Property (put): **Status = value [in] Type: enum LoadStatus, [Value**
(no Help string available)

Property (get): **value = Growth [out, retval] Type: BSTR* Value**
Name of the growthshape curve for yearly load growth factors.

Property (put): **Growth = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = ZIPV [out, retval] Type: VARIANT* Value**
Array of 7 doubles with values for ZIPV property of the LOAD object

Property (put): **ZIPV = value [in] Type: VARIANT Value**
(no Help string available)

Property (get): **value = pctSeriesRL [out, retval] Type: double* Value**
(no Help string available)

Property (put): **pctSeriesRL = value [in] Type: double Value**
Percent of Load that is modeled as series R-L for harmonics studies

Property (get): **value = RelWeight [out, retval] Type: double* Value**
Relative Weighting factor for the active LOAD

Property (put): **RelWeight = value [in] Type: double Value**
Relative Weighting factor for the active LOAD

DSSElement Interface

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Full Name of Active DSS Object (general element or circuit element

Property (get): **value = Properties [in] Type: VARIANT Indx, [out, retval] Type: IDSSProperty** Value**
Collection of properties for Active DSS object (general element or circuit element

Property (get): **value = NumProperties [out, retval] Type: long* Value**
Number of Properties for the active DSS object.

Property (get): **value = AllPropertyNames [out, retval] Type: VARIANT* Value**
Variant array of strings containing the names of all properties for the active DSS object.

ActiveClass Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings consisting of all element names in the active class.

Property (get): **value = First [out, retval] Type: long* Value**
Sets first element in the active class to be the active DSS object. If object is a CktElement, ActiveCktELment also points to this element. Returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Sets next element in active class to be the active DSS object. If object is a CktElement, ActiveCktElement also points to this element. Returns 0 if no more.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Name of the Active Element of the Active Class

Property (put): **Name = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = NumElements [out, retval] Type: long* Value**
Number of elements in this class. Same as Count property.

Property (get): **value = ActiveClassName [out, retval] Type: BSTR* Value**

Returns name of active class.

Property (get): **value = Count [out, retval] Type: long* Value**
Number of elements in Active Class. Same as NumElements Property.

Capacitors Interface

Property (get): **value = kV [out, retval] Type: double* Value**
Bank kV rating. Use LL for 2 or 3 phases, or actual can rating for 1 phase.

Property (put): **kV = value [in] Type: double Value**
Bank kV rating. Use LL for 2 or 3 phases, or actual can rating for 1 phase.

Property (get): **value = kvar [out, retval] Type: double* Value**
Total bank KVAR, distributed equally among phases and steps.

Property (put): **kvar = value [in] Type: double Value**
Total bank KVAR, distributed equally among phases and steps.

Property (get): **value = NumSteps [out, retval] Type: long* Value**
Number of steps (default 1

Property (put): **NumSteps = value [in] Type: long Value**
Number of steps (default 1

Property (get): **value = IsDelta [out, retval] Type: VARIANT_BOOL* Value**
Delta connection or wye?

Property (put): **IsDelta = value [in] Type: VARIANT_BOOL Value**
Delta connection or wye?

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings with all Capacitor names in the circuit.

Property (get): **value = First [out, retval] Type: long* Value**
Sets the first Capacitor active. Returns 0 if no more.

Property (get): **value = Next [out, retval] Type: long* Value**
Sets the next Capacitor active. Returns 0 if no more.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Sets the acitve Capacitor by Name.

Property (put): **Name = value [in] Type: BSTR Value**

Sets the acitve Capacitor by Name.

Property (get): **value = Count [out, retval] Type: long* Value**
Number of Capacitor objects in active circuit.

Transformers Interface

Property (get): **value = NumWindings [out, retval] Type: long* Value**
Number of windings on this transformer. Allocates memory; set or change this property first.

Property (put): **NumWindings = value [in] Type: long Value**
Number of windings on this transformer. Allocates memory; set or change this property first.

Property (get): **value = XfmrCode [out, retval] Type: BSTR* Value**
Name of an XfrmCode that supplies electircal parameters for this Transformer.

Property (put): **XfmrCode = value [in] Type: BSTR Value**
Name of an XfrmCode that supplies electircal parameters for this Transformer.

Property (get): **value = Wdg [out, retval] Type: long* Value**
Active Winding Number from 1..NumWindings. Update this before reading or setting a sequence of winding properties (R, Tap, kV, kVA, etc.

Property (put): **Wdg = value [in] Type: long Value**
Active Winding Number from 1..NumWindings. Update this before reading or setting a sequence of winding properties (R, Tap, kV, kVA, etc.

Property (get): **value = R [out, retval] Type: double* Value**
Active Winding resistance in %

Property (put): **R = value [in] Type: double Value**
Active Winding resistance in %

Property (get): **value = Tap [out, retval] Type: double* Value**
Active Winding tap in per-unit.

Property (put): **Tap = value [in] Type: double Value**
Active Winding tap in per-unit.

Property (get): **value = MinTap [out, retval] Type: double* Value**
Active Winding minimum tap in per-unit.

Property (put): **MinTap = value [in] Type: double Value**

Active Winding minimum tap in per-unit.

Property (get): **value = MaxTap [out, retval] Type: double* Value**
Active Winding maximum tap in per-unit.

Property (put): **MaxTap = value [in] Type: double Value**
Active Winding maximum tap in per-unit.

Property (get): **value = NumTaps [out, retval] Type: long* Value**
Active Winding number of tap steps betwein MinTap and MaxTap.

Property (put): **NumTaps = value [in] Type: long Value**
Active Winding number of tap steps betwein MinTap and MaxTap.

Property (get): **value = kV [out, retval] Type: double* Value**
Active Winding kV rating. Phase-phase for 2 or 3 phases, actual winding kV for 1 phase transformer.

Property (put): **kV = value [in] Type: double Value**
Active Winding kV rating. Phase-phase for 2 or 3 phases, actual winding kV for 1 phase transformer.

Property (get): **value = kVA [out, retval] Type: double* Value**
Active Winding kVA rating. On winding 1, this also determines normal and emergency current ratings for all windings.

Property (put): **kVA = value [in] Type: double Value**
Active Winding kVA rating. On winding 1, this also determines normal and emergency current ratings for all windings.

Property (get): **value = Xneut [out, retval] Type: double* Value**
Active Winding neutral reactance [ohms] for wye connections.

Property (put): **Xneut = value [in] Type: double Value**
Active Winding neutral reactance [ohms] for wye connections.

Property (get): **value = Rneut [out, retval] Type: double* Value**
Active Winding neutral resistance [ohms] for wye connections. Set less than zero for ungrounded wye.

Property (put): **Rneut = value [in] Type: double Value**
Active Winding neutral resistance [ohms] for wye connections. Set less than zero for ungrounded wye.

Property (get): **value = IsDelta [out, retval] Type: VARIANT_BOOL* Value**
Active Winding delta or wye connection?

Property (put): **IsDelta = value [in] Type: VARIANT_BOOL Value**
Active Winding delta or wye connection?

Property (get): **value = Xhl [out, retval] Type: double* Value**
Percent reactance between windings 1 and 2, on winding 1 kVA base. Use for 2-winding or 3-winding transformers.

Property (put): **Xhl = value [in] Type: double Value**
Percent reactance between windings 1 and 2, on winding 1 kVA base. Use for 2-winding or 3-winding transformers.

Property (get): **value = Xht [out, retval] Type: double* Value**
Percent reactance between windigns 1 and 3, on winding 1 kVA base. Use for 3-winding transformers only.

Property (put): **Xht = value [in] Type: double Value**
Percent reactance between windigns 1 and 3, on winding 1 kVA base. Use for 3-winding transformers only.

Property (get): **value = Xlt [out, retval] Type: double* Value**
Percent reactance between windings 2 and 3, on winding _1_ kVA base. Use for 3-winding transformers only.

Property (put): **Xlt = value [in] Type: double Value**
Percent reactance between windings 2 and 3, on winding _1_ kVA base. Use for 3-winding transformers only.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Sets a Transformer active by Name.and 3, on winding _1_ kVA base. Use for 3-winding transformers only.

Property (put): **Name = value [in] Type: BSTR Value**
Sets a Transformer active by Name.and 3, on winding _1_ kVA base. Use for 3-winding transformers only.

Property (get): **value = First [out, retval] Type: long* Value**
Sets the first Transformer active. Returns 0 if no more.

Property (get): **value = Next [out, retval] Type: long* Value**

Sets the next Transformer active. Returns 0 if no more.

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings with all Transformer names in the active circuit.

Property (get): **value = Count [out, retval] Type: long* Value**
(no Help string available)

SwtControls Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings with all SwtControl names in the active circuit.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Sets a SwtControl active by Name.

Property (put): **Name = value [in] Type: BSTR Value**
Sets a SwtControl active by Name.

Property (get): **value = First [out, retval] Type: long* Value**
Sets the first SwtControl active. Returns 0 if no more.

Property (get): **value = Next [out, retval] Type: long* Value**
Sets the next SwtControl active. Returns 0 if no more.

Property (get): **value = Action [out, retval] Type: enum ActionCodes*, [Value**
Open or Close the switch. No effect if switch is locked. However, Reset removes any lock and then closes the switch (shelf state

Property (put): **Action = value [in] Type: enum ActionCodes, [Value**
Open or Close the switch. No effect if switch is locked. However, Reset removes any lock and then closes the switch (shelf state

Property (get): **value = IsLocked [out, retval] Type: VARIANT_BOOL* Value**
The lock prevents both manual and automatic switch operation.

Property (put): **IsLocked = value [in] Type: VARIANT_BOOL Value**
The lock prevents both manual and automatic switch operation.

Property (get): **value = Delay [out, retval] Type: double* Value**
Time delay [s] betwen arming and opening or closing the switch. Control may reset before actually operating the switch.

Property (put): **Delay = value [in] Type: double Value**

Time delay [s] betwen arming and opening or closing the switch. Control may reset before actually operating the switch.

Property (get): **value = SwitchedObj [out, retval] Type: BSTR* Value**
Full name of the switched element.

Property (put): **SwitchedObj = value [in] Type: BSTR Value**
Full name of the switched element.

Property (get): **value = SwitchedTerm [out, retval] Type: long* Value**
Terminal number where the switch is located on the SwitchedObj

Property (put): **SwitchedTerm = value [in] Type: long Value**
Terminal number where the switch is located on the SwitchedObj

Property (get): **value = Count [out, retval] Type: long* Value**
(no Help string available)

CapControls Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings with all CapControl names.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Sets a CapControl active by name.

Property (put): **Name = value [in] Type: BSTR Value**
Sets a CapControl active by name.

Property (get): **value = First [out, retval] Type: long* Value**
Sets the first CapControl as active. Return 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Gets the next CapControl in the circut. Returns 0 if none.

Property (get): **value = Mode [out, retval] Type: enum CapControlModes*, [Value**
Type of automatic controller.

Property (put): **Mode = value [in] Type: enum CapControlModes, [Value**
Type of automatic controller.

Property (get): **value = Capacitor [out, retval] Type: BSTR* Value**
Name of the Capacitor that is controlled.

Property (put): **Capacitor = value [in] Type: BSTR Value**
Name of the Capacitor that is controlled.

Property (get): **value = MonitoredObj [out, retval] Type: BSTR* Value**
Full name of the element that PT and CT are connected to.

Property (put): **MonitoredObj = value [in] Type: BSTR Value**
Full name of the element that PT and CT are connected to.

Property (get): **value = MonitoredTerm [out, retval] Type: long* Value**
Terminal number on the element that PT and CT are connected to.

Property (put): **MonitoredTerm = value [in] Type: long Value**
Terminal number on the element that PT and CT are connected to.

Property (get): **value = CTratio [out, retval] Type: double* Value**
Transducer ratio from pirmary current to control current.

Property (put): **CTratio = value [in] Type: double Value**
Transducer ratio from pirmary current to control current.

Property (get): **value = PTratio [out, retval] Type: double* Value**
Transducer ratio from primary feeder to control voltage.

Property (put): **PTratio = value [in] Type: double Value**
Transducer ratio from primary feeder to control voltage.

Property (get): **value = ONSetting [out, retval] Type: double* Value**
Threshold to arm or switch on a step. See Mode for units.

Property (put): **ONSetting = value [in] Type: double Value**
Threshold to arm or switch on a step. See Mode for units.

Property (get): **value = OFFSetting [out, retval] Type: double* Value**
Threshold to switch off a step. See Mode for units.

Property (put): **OFFSetting = value [in] Type: double Value**
Threshold to switch off a step. See Mode for units.

Property (get): **value = Vmax [out, retval] Type: double* Value**
With VoltOverride, swtich off whenever PT voltage exceeds this level.

Property (put): **Vmax = value [in] Type: double Value**
With VoltOverride, swtich off whenever PT voltage exceeds this level.

Property (get): **value = Vmin [out, retval] Type: double* Value**
With VoltOverride, switch ON whenever PT voltage drops below this level.

Property (put): **Vmin = value [in] Type: double Value**
With VoltOverride, switch ON whenever PT voltage drops below this level.

Property (get): **value = UseVoltOverride [out, retval] Type: VARIANT_BOOL* Value**
Enables Vmin and Vmax to override the control Mode

Property (put): **UseVoltOverride = value [in] Type: VARIANT_BOOL Value**
Enables Vmin and Vmax to override the control Mode

Property (get): **value = Delay [out, retval] Type: double* Value**
Time delay [s] to switch on after arming. Control may reset before actually switching.

Property (put): **Delay = value [in] Type: double Value**
Time delay [s] to switch on after arming. Control may reset before actually switching.

Property (get): **value = DelayOff [out, retval] Type: double* Value**
Time delay [s] before swithcing off a step. Control may reset before actually switching.

Property (put): **DelayOff = value [in] Type: double Value**
Time delay [s] before swithcing off a step. Control may reset before actually switching.

Property (get): **value = DeadTime [out, retval] Type: double* Value**
(no Help string available)

Property (put): **DeadTime = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Count [out, retval] Type: long* Value**
Number of CapControls in Active Circuit

RegControls Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings containing all RegControl names

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Get/set Active RegControl name

Property (put): **Name = value [in] Type: BSTR Value**
Sets a RegControl active by name

Property (get): **value = First [out, retval] Type: long* Value**
Sets the first RegControl active. Returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Sets the next RegControl active. Returns 0 if none.

Property (get): **value = MonitoredBus [out, retval] Type: BSTR* Value**
Name of a remote regulated bus, in lieu of LDC settings

Property (put): **MonitoredBus = value [in] Type: BSTR Value**
Name of a remote regulated bus, in lieu of LDC settings

Property (get): **value = Transformer [out, retval] Type: BSTR* Value**
Name of the transformer this regulator controls

Property (put): **Transformer = value [in] Type: BSTR Value**
Name of the transformer this regulator controls

Property (get): **value = TapWinding [out, retval] Type: long* Value**
Tapped winding number

Property (put): **TapWinding = value [in] Type: long Value**
Tapped winding number

Property (get): **value = Winding [out, retval] Type: long* Value**
Winding number for PT and CT connections

Property (put): **Winding = value [in] Type: long Value**
Winding number for PT and CT connections

Property (get): **value = CTPrimary [out, retval] Type: double* Value**
CT primary ampere rating (secondary is 0.2 amperes

Property (put): **CTPrimary = value [in] Type: double Value**
CT primary ampere rating (secondary is 0.2 amperes

Property (get): **value = PTratio [out, retval] Type: double* Value**
PT ratio for voltage control settings

Property (put): **PTratio = value [in] Type: double Value**
PT ratio for voltage control settings

Property (get): **value = ForwardR [out, retval] Type: double* Value**
LDC R setting in Volts

Property (put): **ForwardR = value [in] Type: double Value**
LDC R setting in Volts

Property (get): **value = ForwardX [out, retval] Type: double* Value**
LDC X setting in Volts

Property (put): **ForwardX = value [in] Type: double Value**
LDC X setting in Volts

Property (get): **value = ReverseR [out, retval] Type: double* Value**
Reverse LDC R setting in Volts.

Property (put): **ReverseR = value [in] Type: double Value**
Reverse LDC R setting in Volts.

Property (get): **value = ReverseX [out, retval] Type: double* Value**
Reverse LDC X setting in volts.

Property (put): **ReverseX = value [in] Type: double Value**
Reverse LDC X setting in volts.

Property (get): **value = IsReversible [out, retval] Type: VARIANT_BOOL* Value**
Regulator can use different settings in the reverse direction. Usually not applicable to substation transformers.

Property (put): **IsReversible = value [in] Type: VARIANT_BOOL Value**
Regulator can use different settings in the reverse direction. Usually not applicable to substation transformers.

Property (get): **value = IsInverseTime [out, retval] Type: VARIANT_BOOL* Value**
Time delay is inversely adjsuted, proportinal to the amount of voltage outside the regulating band.

Property (put): **IsInverseTime = value [in] Type: VARIANT_BOOL Value**
Time delay is inversely adjsuted, proportinal to the amount of voltage outside the regulating band.

Property (get): **value = Delay [out, retval] Type: double* Value**
Time delay [s] after arming before the first tap change. Control may reset before actually changing taps.

Property (put): **Delay = value [in] Type: double Value**
Time delay [s] after arming before the first tap change. Control may reset before actually

changing taps.

Property (get): **value = TapDelay [out, retval] Type: double* Value**
Time delay [s] for subsequent tap changes in a set. Control may reset before actually changing taps.

Property (put): **TapDelay = value [in] Type: double Value**
Time delay [s] for subsequent tap changes in a set. Control may reset before actually changing taps.

Property (get): **value = MaxTapChange [out, retval] Type: long* Value**
Maximum tap change per iteration in STATIC solution mode. 1 is more realistic, 16 is the default for a faster soluiton.

Property (put): **MaxTapChange = value [in] Type: long Value**
Maximum tap change per iteration in STATIC solution mode. 1 is more realistic, 16 is the default for a faster soluiton.

Property (get): **value = VoltageLimit [out, retval] Type: double* Value**
First house voltage limit on PT secondary base. Setting to 0 disables this function.

Property (put): **VoltageLimit = value [in] Type: double Value**
First house voltage limit on PT secondary base. Setting to 0 disables this function.

Property (get): **value = ForwardBand [out, retval] Type: double* Value**
Regulation bandwidth in forward direciton, centered on Vreg

Property (put): **ForwardBand = value [in] Type: double Value**
Regulation bandwidth in forward direciton, centered on Vreg

Property (get): **value = ForwardVreg [out, retval] Type: double* Value**
Target voltage in the forward direction, on PT secondary base.

Property (put): **ForwardVreg = value [in] Type: double Value**
Target voltage in the forward direction, on PT secondary base.

Property (get): **value = ReverseBand [out, retval] Type: double* Value**
Bandwidth in reverse direction, centered on reverse Vreg.

Property (put): **ReverseBand = value [in] Type: double Value**
Bandwidth in reverse direction, centered on reverse Vreg.

Property (get): **value = ReverseVreg [out, retval] Type: double* Value**

Target voltage in the revese direction, on PT secondary base.

Property (put): **ReverseVreg = value [in] Type: double Value**
Target voltage in the revese direction, on PT secondary base.

Property (get): **value = Count [out, retval] Type: long* Value**
Number of RegControl objects in Active Circuit

Property (get): **value = TapNumber [out, retval] Type: long* Value**
(no Help string available)

Property (put): **TapNumber = value [in] Type: long Value**
Integer number of the tap that the controlled transformer winding is currentliy on.

Topology Interface

Property (get): **value = NumLoops [out, retval] Type: long* Value**
Number of loops

Property (get): **value = NumIsolatedBranches [out, retval] Type: long* Value**
Number of isolated branches (PD elements and capacitors

Property (get): **value = AllLoopedPairs [out, retval] Type: VARIANT* Value**
Variant array of all looped element names, by pairs.

Property (get): **value = AllIsolatedBranches [out, retval] Type: VARIANT* Value**
Variant array of all isolated branch names.

Property (get): **value = NumIsolatedLoads [out, retval] Type: long* Value**
Number of isolated loads

Property (get): **value = AllIsolatedLoads [out, retval] Type: VARIANT* Value**
Variant array of all isolated load names.

Property (get): **value = BranchName [out, retval] Type: BSTR* Value**
Name of the active branch.

Property (put): **BranchName = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = First [out, retval] Type: long* Value**
Sets the first branch active, returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**

Sets the next branch active, returns 0 if no more.

Property (get): **value = ActiveBranch [out, retval] Type: long* Value**
Returns index of the active branch

Property (get): **value = ForwardBranch [out, retval] Type: long* Value**
Move forward in the tree, return index of new active branch or 0 if no more

Property (get): **value = BackwardBranch [out, retval] Type: long* Value**
MOve back toward the source, return index of new active branch, or 0 if no more.

Property (get): **value = LoopedBranch [out, retval] Type: long* Value**
Move to looped branch, return index or 0 if none.

Property (get): **value = ParallelBranch [out, retval] Type: long* Value**
Move to directly parallel branch, return index or 0 if none.

Property (get): **value = FirstLoad [out, retval] Type: long* Value**
First load at the active branch, return index or 0 if none.

Property (get): **value = NextLoad [out, retval] Type: long* Value**
Next load at the active branch, return index or 0 if no more.

Property (get): **value = ActiveLevel [out, retval] Type: long* Value**
Topological depth of the active branch

Property (get): **value = BusName [out, retval] Type: BSTR* Value**
(no Help string available)

Property (put): **BusName = value [in] Type: BSTR Value**
Set the active branch to one containing this bus, return index or 0 if not found

DSS_Executive Interface

Property (get): **value = NumCommands [out, retval] Type: long* Value**
Number of DSS Executive Commands

Property (get): **value = NumOptions [out, retval] Type: long* Value**
Number of DSS Executive Options

Property (get): **value = Command [in] Type: long i, [out, retval] Type: BSTR* Value**
Get i-th command

Property (get): **value = Option [in] Type: long i, [out, retval] Type: BSTR* Value**

Get i-th option

Property (get): **value = CommandHelp [in] Type: long i, [out, retval] Type: BSTR* Value**
Get help string for i-th command

Property (get): **value = OptionHelp [in] Type: long i, [out, retval] Type: BSTR* Value**
Get help string for i-th option

Property (get): **value = OptionValue [in] Type: long i, [out, retval] Type: BSTR* Value**
Get present value of i-th option

DSSEvents Interface

Sensors Interface

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Name of the active sensor.

Property (put): **Name = value [in] Type: BSTR Value**
Set the active Sensor by name.

Property (get): **value = Count [out, retval] Type: long* Value**
Number of Sensors in Active Circuit.

Property (get): **value = First [out, retval] Type: long* Value**
Sets the first sensor active. Returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Sets the next Sensor active. Returns 0 if no more.

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of Sensor names.

Property (get): **value = IsDelta [out, retval] Type: VARIANT_BOOL* Value**
True if measured voltages are line-line. Currents are always line currents.

Property (put): **IsDelta = value [in] Type: VARIANT_BOOL Value**
(no Help string available)

Property (get): **value = ReverseDelta [out, retval] Type: VARIANT_BOOL* Value**
True if voltage measurements are 1-3, 3-2, 2-1.

Property (put): **ReverseDelta = value [in] Type: VARIANT_BOOL Value**

(no Help string available)

Property (get): **value = PctError [out, retval] Type: double* Value**
Assumed percent error in the Sensor measurement. Default is 1.

Property (put): **PctError = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Weight [out, retval] Type: double* Value**
Weighting factor for this Sensor measurement with respect to other Sensors. Default is 1.

Property (put): **Weight = value [in] Type: double Value**
(no Help string available)

Property (get): **value = MeteredElement [out, retval] Type: BSTR* Value**
Full Name of the measured element

Property (put): **MeteredElement = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = MeteredTerminal [out, retval] Type: long* Value**
Number of the measured terminal in the measured element.

Property (put): **MeteredTerminal = value [in] Type: long Value**
(no Help string available)

Method: **Reset [void**
Clear the active Sensor.

Method: **ResetAll [void**
Clear all Sensors in the Active Circuit.

Property (get): **value = kVbase [out, retval] Type: double* Value**
Voltage base for the sensor measurements. LL for 2 and 3-phase sensors, LN for 1-phase sensors.

Property (put): **kVbase = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Currents [out, retval] Type: VARIANT* Value**
Array of doubles for the line current measurements; don't use with kWS and kVARS.

Property (put): **Currents = value [in] Type: VARIANT Value**
(no Help string available)

Property (get): **value = kVS [out, retval] Type: VARIANT* Value**

Array of doubles for the LL or LN (depending on Delta connection

Property (put): **kVS = value [in] Type: VARIANT Value**
(no Help string available)

Property (get): **value = kVARS [out, retval] Type: VARIANT* Value**
Array of doubles for Q measurements. Overwrites Currents with a new estimate using kWS.

Property (put): **kVARS = value [in] Type: VARIANT Value**
(no Help string available)

Property (get): **value = kWS [out, retval] Type: VARIANT* Value**
Array of doubles for P measurements. Overwrites Currents with a new estimate using kVARS.

Property (put): **kWS = value [in] Type: VARIANT Value**
(no Help string available)

XYCurves Interface

Property (get): **value = Count [out, retval] Type: long* Value**
Number of XYCurve Objects

Property (get): **value = First [out, retval] Type: long* Value**
Sets first XYcurve object active; returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Advances to next XYCurve object; returns 0 if no more objects of this class

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Name of active XYCurve Object

Property (put): **Name = value [in] Type: BSTR Value**
Get Name of active XYCurve Object

Property (get): **value = Npts [out, retval] Type: long* Value**
Get/Set Number of points in X-Y curve

Property (put): **Npts = value [in] Type: long Value**
Get/Set Number of Points in X-Y curve

Property (get): **value = Xarray [out, retval] Type: VARIANT* Value**
Get/Set X values as a Variant array of doubles. Set Npts to max number expected if setting

Property (put): **Xarray = value [in] Type: VARIANT Value**

Get/Set X values as a Variant array of doubles. Set Npts to max number expected if setting

Property (get): **value = Yarray [out, retval] Type: VARIANT* Value**
Get/Set Y values in curve; Set Npts to max number expected if setting

Property (put): **Yarray = value [in] Type: VARIANT Value**
Get/Set Y values in curve; Set Npts to max number expected if setting

Property (get): **value = x [out, retval] Type: double* Value**
Set X value or get interpolated value after setting Y

Property (put): **x = value [in] Type: double Value**
(no Help string available)

Property (get): **value = y [out, retval] Type: double* Value**
Y value for present X or set this value then get corresponding X

Property (put): **y = value [in] Type: double Value**
Set Y value or get interpolated Y value after setting X

Property (get): **value = Xshift [out, retval] Type: double* Value**
Amount to shift X value from original curve

Property (put): **Xshift = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Yshift [out, retval] Type: double* Value**
amount to shift Y valiue from original curve

Property (put): **Yshift = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Xscale [out, retval] Type: double* Value**
Factor to scale X values from original curve

Property (put): **Xscale = value [in] Type: double Value**
Factor to scale X values from original curve

Property (get): **value = Yscale [out, retval] Type: double* Value**
Factor to scale Y values from original curve

Property (put): **Yscale = value [in] Type: double Value**
Amount to scale Y values from original curve. Represents a curve shift.

PDElements Interface

Property (get): **value = Count [out, retval] Type: long* Value**
Number of PD elements (including disabled elements

Property (get): **value = First [out, retval] Type: long* Value**
Set the first enabled PD element to be the active element. Returns 0 if none found.

Property (get): **value = Next [out, retval] Type: long* Value**
Advance to the next PD element in the circuit. Enabled elements only. Returns 0 when no more elements.

Property (get): **value = IsShunt [out, retval] Type: VARIANT_BOOL* Value**
Variant boolean indicating of PD element should be treated as a shunt element rather than a series element. Applies to Capacitor and Reactor elements in particular.

Property (get): **value = FaultRate [out, retval] Type: double* Value**
Get/Set Number of failures per year. For LINE elements: Number of failures per unit length per year.

Property (put): **FaultRate = value [in] Type: double Value**
(no Help string available)

Property (get): **value = pctPermanent [out, retval] Type: double* Value**
Get/Set percent of faults that are permanent (require repair

Property (put): **pctPermanent = value [in] Type: double Value**
(no Help string available)

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Get/Set name of active PD Element. Returns null string if active element is not PDElement type.

Property (put): **Name = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = Lambda [out, retval] Type: double* Value**
Failure rate for this branch. Faults per year including length of line.

Property (get): **value = AccumulatedL [out, retval] Type: double* Value**
accummulated failure rate for this branch on downline

Property (get): **value = RepairTime [out, retval] Type: double* Value**
Average time to repair a permanent fault on this branch, hours.

Property (get): **value = Numcustomers [out, retval] Type: long* Value**
Number of customers, this branch

Property (get): **value = Totalcustomers [out, retval] Type: long* Value**
Total number of customers from this branch to the end of the zone

Property (get): **value = ParentPDElement [out, retval] Type: long* Value**
Sets the parent PD element to be the active circuit element. Returns 0 if no more elements upline.

Property (get): **value = FromTerminal [out, retval] Type: long* Value**
Number of the terminal of active PD element that is on the \i0

Reclosers Interface

Property (get): ***value = AllNames [out, retval] Type: VARIANT* Value***
Variant array of strings with names of all Reclosers in Active Circuit

Property (get): **value = Count [out, retval] Type: long* Value**
Number of Reclosers in active circuit.

Property (get): **value = First [out, retval] Type: long* Value**
Set First Recloser to be Active Ckt Element. Returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Iterate to the next recloser in the circuit. Returns zero if no more.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Get Name of active Recloser or set the active Recloser by name.

Property (put): **Name = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = MonitoredObj [out, retval] Type: BSTR* Value**
Full name of object this Recloser is monitoring.

Property (put): **MonitoredObj = value [in] Type: BSTR Value**
Set monitored object by full name.

Property (get): **value = MonitoredTerm [out, retval] Type: long* Value**
Terminal number of Monitored object for the Recloser

Property (put): **MonitoredTerm = value [in] Type: long Value**

(no Help string available)

Property (get): **value = SwitchedObj [out, retval] Type: BSTR* Value**
Full name of the circuit element that is being switched by the Recloser.

Property (put): **SwitchedObj = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = SwitchedTerm [out, retval] Type: long* Value**
Terminal number of the controlled device being switched by the Recloser

Property (put): **SwitchedTerm = value [in] Type: long Value**
(no Help string available)

Property (get): **value = NumFast [out, retval] Type: long* Value**
Number of fast shots

Property (put): **NumFast = value [in] Type: long Value**
(no Help string available)

Property (get): **value = Shots [out, retval] Type: long* Value**
Number of shots to lockout (fast + delayed

Property (put): **Shots = value [in] Type: long Value**
(no Help string available)

Property (get): **value = RecloseIntervals [out, retval] Type: VARIANT* Value**
Variant Array of Doubles: reclose intervals, s, between shots.

Property (get): **value = PhaseTrip [out, retval] Type: double* Value**
Phase trip curve multiplier or actual amps

Property (put): **PhaseTrip = value [in] Type: double Value**
Phase Trip multiplier or actual amps

Property (get): **value = PhaseInst [out, retval] Type: double* Value**
Phase instantaneous curve multipler or actual amps

Property (put): **PhaseInst = value [in] Type: double Value**
(no Help string available)

Property (get): **value = GroundTrip [out, retval] Type: double* Value**
Ground (3I0

Property (put): **GroundTrip = value [in] Type: double Value**

(no Help string available)

Property (get): **value = GroundInst [out, retval] Type: double* Value**
Ground (3I0

Property (put): **GroundInst = value [in] Type: double Value**
Ground (3I0

Method: **Open [void**
Open recloser's controlled element and lock out the recloser

Method: **Close [void**
Close the switched object controlled by the recloser. Resets recloser to first operation.

Property (get): **value = idx [out, retval] Type: long* Value**
Get/Set the active Recloser by index into the recloser list. 1..Count

Property (put): **idx = value [in] Type: long Value**
Get/Set the Active Recloser by index into the recloser list. 1..Count

Relays Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings containing names of all Relay elements

Property (get): **value = Count [out, retval] Type: long* Value**
Number of Relays in circuit

Property (get): **value = First [out, retval] Type: long* Value**
Set First Relay active. If none, returns 0.

Property (get): **value = Next [out, retval] Type: long* Value**
Advance to next Relay object. Returns 0 when no more relays.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Get name of active relay.

Property (put): **Name = value [in] Type: BSTR Value**
Set Relay active by name

Property (get): **value = MonitoredObj [out, retval] Type: BSTR* Value**
Full name of object this Relay is monitoring.

Property (put): **MonitoredObj = value [in] Type: BSTR Value**

(no Help string available)

Property (get): **value = MonitoredTerm [out, retval] Type: long* Value**
Number of terminal of monitored element that this Relay is monitoring.

Property (put): **MonitoredTerm = value [in] Type: long Value**
(no Help string available)

Property (get): **value = SwitchedObj [out, retval] Type: BSTR* Value**
Full name of element that will be switched when relay trips.

Property (put): **SwitchedObj = value [in] Type: BSTR Value**
(no Help string available)

Property (get): **value = SwitchedTerm [out, retval] Type: long* Value**
(no Help string available)

Property (put): **SwitchedTerm = value [in] Type: long Value**
Terminal number of the switched object that will be opened when the relay trips.

Property (get): **value = idx [out, retval] Type: long* Value**
Get/Set active Relay by index into the Relay list. 1..Count

Property (put): **idx = value [in] Type: long Value**
Get/Set Relay active by index into relay list. 1..Count

CmathLib Interface

Property (get): **value = cmplx [in] Type: double RealPart, [in] Type: double ImagPart, [out, retval] Type: VARIANT* Value**
Convert real and imaginary doubles to Variant array of doubles

Property (get): **value = cabs [in] Type: double realpart, [in] Type: double imagpart, [out, retval] Type: double* Value**
Return abs value of complex number given in real and imag doubles

Property (get): **value = cdang [in] Type: double RealPart, [in] Type: double ImagPart, [out, retval] Type: double* Value**
Returns the angle, in degrees, of a complex number specified as two doubles: Realpart and imagpart.

Property (get): **value = ctopolardeg [in] Type: double RealPart, [in] Type: double ImagPart, [out, retval] Type: VARIANT* Value**

Convert complex number to magnitude and angle, degrees. Returns variant array of two doubles.

Property (get): **value = pdegtocomplex [in] Type: double magnitude, [in] Type: double angle, [out, retval] Type: VARIANT* Value**
Convert magnitude, angle in degrees to a complex number. Returns Variant array of two doubles.

Property (get): **value = cmul [in] Type: double a1, [in] Type: double b1, [in] Type: double a2, [in] Type: double b2, [out, retval] Type: VARIANT* Value**
Multiply two complex numbers: (a1, b1

Property (get): **value = cdiv [in] Type: double a1, [in] Type: double b1, [in] Type: double a2, [in] Type: double b2, [out, retval] Type: VARIANT* Value**
Divide two complex number: (a1, b1

Parser Interface

Property (get): **value = CmdString [out, retval] Type: BSTR* Value**
String to be parsed. Loading this string resets the Parser to the beginning of the line. Then parse off the tokens in sequence.

Property (put): **CmdString = value [in] Type: BSTR Value**
String to be parsed. Loading this string resets the Parser to the beginning of the line. Then parse off the tokens in sequence.

Property (get): **value = NextParam [out, retval] Type: BSTR* Value**
Get next token and return tag name (before = sign

Property (get): **value = AutoIncrement [out, retval] Type: VARIANT_BOOL* Value**
Default is FALSE. If TRUE parser automatically advances to next token after DblValue, IntValue, or StrValue. Simpler when you don't need to check for parameter names.

Property (put): **AutoIncrement = value [in] Type: VARIANT_BOOL Value**
Default is FALSE. If TRUE parser automatically advances to next token after DblValue, IntValue, or StrValue. Simpler when you don't need to check for parameter names.

Property (get): **value = DblValue [out, retval] Type: double* Value**
Return next parameter as a double.

Property (get): **value = IntValue [out, retval] Type: long* Value**
Return next parameter as a long integer.

Property (get): **value = StrValue [out, retval] Type: BSTR* Value**
Return next parameter as a string

Property (get): **value = WhiteSpace [out, retval] Type: BSTR* Value**
Get the characters used for White space in the command string. Default is blank and Tab.

Property (put): **WhiteSpace = value [in] Type: BSTR Value**
Set the characters used for White space in the command string. Default is blank and Tab.

Property (get): **value = BeginQuote [out, retval] Type: BSTR* Value**
Get String containing the the characters for Quoting in OpenDSS scripts. Matching pairs defined in EndQuote. Default is \i0

*Property (put): **BeginQuote = value [in] Type: BSTR Value***
Set String containing the the characters for Quoting in OpenDSS scripts. Matching pairs defined in EndQuote. Default is \i0

*Property (get): **value = EndQuote [out, retval] Type: BSTR* Value***
String containing characters, in order, that match the beginning quote characters in BeginQuote. Default is \i0

*Property (put): **EndQuote = value [in] Type: BSTR Value***
String containing characters, in order, that match the beginning quote characters in BeginQuote. Default is \i0

*Property (get): **value = Delimiters [out, retval] Type: BSTR* Value***
String defining hard delimiters used to separate token on the command string. Default is , and =. The = separates token name from token value. These override whitesspace to separate tokens.

Property (put): **Delimiters = value [in] Type: BSTR Value**
String defining hard delimiters used to separate token on the command string. Default is , and =. The = separates token name from token value. These override whitesspace to separate tokens.

Method: **ResetDelimiters [void**
Reset delimiters to their default values.

Property (get): **value = Vector [in] Type: long ExpectedSize, [out, retval] Type: VARIANT* Value**
Returns token as variant array of doubles. For parsing quoted array syntax.

Property (get): **value = Matrix [in] Type: long ExpectedOrder, [out, retval] Type: VARIANT* Value**
Use this property to parse a Matrix token in OpenDSS format. Returns square matrix of order specified. Order same as default Fortran order: column by column.

Property (get): **value = SymMatrix [in] Type: long ExpectedOrder, [out, retval] Type:**

VARIANT* Value
Use this property to parse a matrix token specified in lower triangle form. Symmetry is forced.

LoadShapes Interface

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Get the Name of the active Loadshape

Property (put): **Name = value [in] Type: BSTR Value**
Set the active Loadshape by name

Property (get): **value = Count [out, retval] Type: long* Value**
Number of Loadshape objects currently defined in Loadshape collection

Property (get): **value = First [out, retval] Type: long* Value**
Set the first loadshape active and return integer index of the loadshape. Returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Advance active Loadshape to the next on in the collection. Returns 0 if no more loadshapes.

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings containing names of all Loadshape objects currently defined.

Property (get): **value = Npts [out, retval] Type: long* Value**
Get Number of points in active Loadshape.

Property (put): **Npts = value [in] Type: long Value**
Set number of points to allocate for active Loadshape.

Property (get): **value = Pmult [out, retval] Type: VARIANT* Value**
Variant array of Doubles for the P multiplier in the Loadshape.

Property (put): **Pmult = value [in] Type: VARIANT Value**
Variant array of doubles containing the P array for the Loadshape.

Property (get): **value = Qmult [out, retval] Type: VARIANT* Value**
Variant array of doubles containing the Q multipliers.

Property (put): **Qmult = value [in] Type: VARIANT Value**
Variant array of doubles containing the Q multipliers.

Method: **Normalize [void**
Normalize the P and Q curves based on either Pbase, Qbase or simply the peak value of the curve.

Property (get): **value = TimeArray [out, retval] Type: VARIANT* Value**
Time array in hours correscponding to P and Q multipliers when the Interval=0.

Property (put): **TimeArray = value [in] Type: VARIANT Value**
Time array in hours correscponding to P and Q multipliers when the Interval=0.

Property (get): **value = HrInterval [out, retval] Type: double* Value**
Fixed interval time value, hours

Property (put): **HrInterval = value [in] Type: double Value**
Fixed interval time value, hours.

Property (get): **value = MinInterval [out, retval] Type: double* Value**
Fixed Interval time value, in minutes

Property (put): **MinInterval = value [in] Type: double Value**
Fixed Interval time value, in minutes

Method: **New [in] Type: BSTR Name**
Make a new Loadshape

Property (get): **value = Pbase [out, retval] Type: double* Value**
Base for normalizing P curve. If left at zero, the peak value is used.

Property (put): **Pbase = value [in] Type: double Value**
Base for normalizing P curve. If left at zero, the peak value is used.

Property (get): **value = Qbase [out, retval] Type: double* Value**
Base for normalizing Q curve. If left at zero, the peak value is used.

Property (put): **Qbase = value [in] Type: double Value**
Base for normalizing Q curve. If left at zero, the peak value is used.

Property (get): **value = UseActual [out, retval] Type: VARIANT_BOOL* Value**
T/F flag to let Loads know to use the actual value in the curve rather than use the value as a multiplier.

Property (put): **UseActual = value [in] Type: VARIANT_BOOL Value**
T/F flag to let Loads know to use the actual value in the curve rather than use the value as a multiplier.

Property (get): **value = Sinterval [out, retval] Type: double* Value**
Fixed interval data time interval, seconds

Property (put): **Sinterval = value [in] Type: double Value**
Fixed interval data time interval, seconds

Fuses Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings containing names of all Fuses in the circuit

Property (get): **value = Count [out, retval] Type: long* Value**
Number of Fuse elements in the circuit

Property (get): **value = First [out, retval] Type: long* Value**
Set the first Fuse to be the active fuse. Returns 0 if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Advance the active Fuse element pointer to the next fuse. Returns 0 if no more fuses.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Get the name of the active Fuse element

Property (put): **Name = value [in] Type: BSTR Value**
Set the active Fuse element by name.

Property (get): **value = MonitoredObj [out, retval] Type: BSTR* Value**
Full name of the circuit element to which the fuse is connected.

Property (put): **MonitoredObj = value [in] Type: BSTR Value**
Full name of the circuit element to which the fuse is connected.

Property (get): **value = MonitoredTerm [out, retval] Type: long* Value**
Terminal number to which the fuse is connected.

Property (put): **MonitoredTerm = value [in] Type: long Value**
Number of the terminal to which the fuse is connected

Property (get): **value = SwitchedObj [out, retval] Type: BSTR* Value**
Full name of the circuit element switch that the fuse controls. Defaults to the MonitoredObj.

Property (put): **SwitchedObj = value [in] Type: BSTR Value**
Full name of the circuit element switch that the fuse controls. Defaults to MonitoredObj.

Property (get): **value = SwitchedTerm [out, retval] Type: long* Value**
Number of the terminal containing the switch controlled by the fuse.

Property (put): **SwitchedTerm = value [in] Type: long Value**
Number of the terminal of the controlled element containing the switch controlled by the fuse.

Property (get): **value = TCCcurve [out, retval] Type: BSTR* Value**
Name of the TCCcurve object that determines fuse blowing.

Property (put): **TCCcurve = value [in] Type: BSTR Value**
Name of the TCCcurve object that determines fuse blowing.

Property (get): **value = RatedCurrent [out, retval] Type: double* Value**
Multiplier or actual amps for the TCCcurve object. Defaults to 1.0. Multipliy current values of TCC curve by this to get actual amps.

Property (put): **RatedCurrent = value [in] Type: double Value**
Multiplier or actual fuse amps for the TCC curve. Defaults to 1.0. Has to correspond to the Current axis of TCCcurve object.

Property (get): **value = Delay [out, retval] Type: double* Value**
A fixed delay time in seconds added to the fuse blowing time determined by the TCC curve. Default is 0.

Property (put): **Delay = value [in] Type: double Value**
Fixed delay time in seconds added to the fuse blowing time to represent fuse clear or other delay.

Method: **Open [void**
Manual opening of fuse

Method: **Close [void**
Close the fuse back in and reset.

Method: **IsBlown [void**
Current state of the fuses. TRUE if any fuse on any phase is blown. Else FALSE.

Property (get): **value = idx [out, retval] Type: long* Value**
Get/set active fuse by index into the list of fuses. 1 based: 1..count

Property (put): **idx = value [in] Type: long Value**
Set Fuse active by index into the list of fuses. 1..count

Property (get): **value = NumPhases [out, retval] Type: long* Value**
Number of phases, this fuse.

ISources Interface

Property (get): **value = AllNames [out, retval] Type: VARIANT* Value**
Variant array of strings containing names of all ISOURCE elements.

Property (get): **value = Count [out, retval] Type: long* Value**
Count: Number of ISOURCE elements.

Property (get): **value = First [out, retval] Type: long* Value**
Set the First ISOURCE to be active; returns Zero if none.

Property (get): **value = Next [out, retval] Type: long* Value**
Sets the next ISOURCE element to be the active one. Returns Zero if no more.

Property (get): **value = Name [out, retval] Type: BSTR* Value**
Get name of active ISOURCE

Property (put): **Name = value [in] Type: BSTR Value**
Set Active ISOURCE by name

Property (get): **value = Amps [out, retval] Type: double* Value**
Get the magnitude of the ISOURCE in amps

Property (put): **Amps = value [in] Type: double Value**
Set the magnitude of the ISOURCE, amps

Property (get): **value = AngleDeg [out, retval] Type: double* Value**
Phase angle for ISOURCE, degrees

Property (put): **AngleDeg = value [in] Type: double Value**
Phase angle for ISOURCE, degrees

Property (get): **value = Frequency [out, retval] Type: double* Value**
The present frequency of the ISOURCE, Hz

Property (put): **Frequency = value [in] Type: double Value**
Set the present frequency for the ISOURCE