OpenDSS COM Documentation

Circuit Class Interface

Updated 8/10/2012

The Circuit Class contains many of the methods and properties one would use to access values related to the circuit, the present solution, and to individual circuit elements. Thus, it is one of the more important to learn how to use to effectively access the OpenDSS throughYou should set a variable to the Circuit interface once the active circuit has been defined.

For example, you might do something like this VB:

```
Set DSSobj = New OpenDSSEngine.DSS  ‘ sets to DSS interface
Set DSSText = DSSObj.Text
…
Set DSSCircuit = DSSObj.ActiveCircuit
Set DSSSolution = DSSCircuit.Solution
```

Where “DSSCircuit” is the variable that points to the Circuit interface.

Then you could use the DSSCircuit variable as follows, for example to obtain all per unit voltages of all buses in the system:

```
‘ Solve Power Flow and put pu voltages in cells
DSSSolution.Solve
If DSSSolution.Converged Then
    V = DSSCircuit.AllBusVmagPu
     …. (process busvoltages ) ….
    For i = Lbound(V) to Ubound(V)
      ThisSheet.Cells(i+3, 4).Value = V(i)
    Next i
End If
```

Note that accessing most of these properties and methods without an active circuit defined will result in no action.

Property Members of Circuit Class

Properties are set by statements of the following format:

```
MyBus = DSSCircuit.Buses(22)
```

All the properties of this interface are read only. Otherwise, you would be able to set a property value by an assignment statement.

As illustrated above, one of the power features of COM functions is that you can retrieve large arrays from the OpenDSS with one statement.

Note that many of the properties simply return a pointer to another class interface not covered by this document. You will have to refer to other documents to learn about them. Note that there are a number of free or inexpensive Type Library Browser that will allow you to explore the exposed elements and properties of OpenDSSEngine.DLL.

ActiveBus: IBus (read only)

Returns an interface to the active bus.

ActiveCktElement: ICktElement (read only)

See ActiveElement.

ActiveClass: IActiveClass (read only)

Returns an interface to the class of the object last referenced or to the class set active the SetActiveClass method (below). The ActiveClass interface will allow you to iterate through all the members of an OpenDSS class by name or using the typical First..Next properties in a loop.

ActiveElement: ICktElement (read only)

Returns an interface to the active circuit element. The active element can be established in a number of ways such as through an Edit or Select command. In the Circuit interface, the SetActiveElement method may be used as well as the methods for iterating through PC elements and PD elements. The CktElements collection property will also set the active circuit element.

AllBusDistances: OleVariant (read only)

Returns all distances from a bus to its parent EnergyMeter element, which is generally in the substation, as a variant array of doubles. Order corresponds to that of all bus properties.

AllBusNames: OleVariant (read only)

Returns a variant array of strings giving the names of all buses in the system. See AllNodeNames.

AllBusVmag: OleVariant (read only)

Similar to AllBusVolts except magnitude only (in actual volts). Returns the voltage (magnitude) for every node in the circuit as a variant array of doubles. The order of the array is the same as AllBusNames property. The array is constructed bus-by-bus and then by node at each bus. Thus, all nodes from each bus are grouped together.

AllBusVmagPu: OleVariant (read only)

Similar to AllBusVmag except that the magnitudes are reported in per unit for all buses with kVBase defined.

AllBusVolts: OleVariant (read only)

Returns the voltage (complex) for every node in the circuit as a variant array of doubles. The order of the array is the same as AllNodeNames property. The array is constructed bus-by-bus and then by node at each bus. Thus, all nodes from each bus are grouped together.

AllElementLosses: OleVariant (read only)

Returns the watt and var losses in each element of the system as a variant array of complex numbers (doubles). Order is the same as AllElementNames.

AllElementNames: OleVariant (read only)

Returns the names of all elements as a variant array of strings.

AllNodeDistances: OleVariant (read only)

Returns the distance from all nodes to the parent energy meter in the same order as AllBusVolts, AllBusVmag, and AllBusVMagPu. Returns a variant array of doubles.

AllNodeNames: OleVariant (read only)

Returns the names of all nodes (busname.nodenumber) in the same order as AllBusVolts, AllBusVmag, and AllBusVMagPu. Returns a variant array of strings.

AllNodeDistancesByPhase[Phase: Integer]: OleVariant (read only)

Returns the distance from all nodes to the parent energy meter that match the designated phase number. Returns a variant array of doubles. Matches the order of AllNodeNamesByPhase, AllNodeVmagByPhase, AllNodeVmagPUByPhase.

AllNodeNamesByPhase[Phase: Integer]: OleVariant (read only)

Returns a variant array of strings in order corresponding to AllNodeDistancesByPhase, AllNodeVmagByPhase, AllNodeVmagPUByPhase. Returns only those names whose phase designator matches the specified Phase.

AllNodeVmagByPhase[Phase: Integer]: OleVariant (read only)

Returns variant array of doubles represent the voltage magnitudes for each node whose phase designator matches the specified Phase.

AllNodeVmagPUByPhase[Phase: Integer]: OleVariant (read only)

Per unit version of AllNodeVmagByPhase. Voltage bases should be defined when using this property. Otherwise the actual voltage to ground is returned.

Buses[Index: OleVariant]: IBus (read only)

Set the active bus using the Index variable and returns an interface to the active bus. Index can be either an integer or a string. If an integer, it refers to the bus index in the circuit’s bus list. If a string, the OpenDSS search for the bus by name.

```
Public MyBus as OpenDSSEngine.Bus

    MyBus = DSSCircuit.Buses(22)
    MyBus = DSSCircuit.Buses("Bus22")
```

CktElements[Idx: OleVariant]: ICktElement (read only)

Selects a particular circuit element and returns an interface to the active circuit element. Note that Idx is a variant. The selection may be done either through an integer handle or the name of the circuit element: In VBA, this might be

```
MyElement1 = DSSCircuit.CktElements(22)
MyElement2 = DSSCircuit.CktElements("line.mylinename")
```

CtrlQueue: ICtrlQueue (read only)

Returns a pointer to the CtrlQueue (control queue) interface.

Capacitors: ICapacitors (read only)

Returns a pointer to the Capacitors collection interface, which can be used to access properties of Capacitor objects currently defined in the existing circuit.

CapControls: ICapControls (read only)

Returns a pointer to the CapControls collection interface, which can be used to access properties of CapControls objects currently defined in the existing circuit.

Generators: IGenerators (read only)

Returns a pointer to the Generators collection interface, which can be used to access properties of Generator objects currently defined in the existing circuit.

Losses: OleVariant (read only)

Total watt and var losses in the entire circuit. Returned as a two-element array of doubles.

LineLosses: OleVariant (read only)

Total watt and var losses in all the Line elements in the circuit. Returned as a two-element array of doubles.

Lines: ILines (read only)

Returns a pointer to the Lines Collections interface, which can be used to access various properties of Line objects in the circuit.

Meters: IMeters (read only)

Returns a pointer to the Meters collection interface, which can be used to control and access the properties of EnergyMeter objects in the circuit.

Monitors: IMonitors (read only)

Returns a pointer to the Monitors collection interface, which can be used to control and access the properties of Monitor objects in the circuit. Monitor objects store their results in byte streams.

Name: WideString (read only)

Returns the name of the active circuit.

NumBuses: Integer (read only)

Return the number of buses in the present circuit. Remember that the bus list is not established until it is needed by a Solve command or forced by the MakeBusList command.

NumCktElements: Integer (read only)

Returns the number of circuit elements in the active circuit. See CktElements property.

NumNodes: Integer (read only)

Return the number of nodes in the present circuit. Remember that the bus list, and therefore, the node list, is not established until it is needed by a Solve command or forced by the MakeBusList command.

RegControls: IRegControls (read only)

Returns a pointer to the RegControls collection interface, which can be used to access properties of RegControls objects (Regulator controls) currently defined in the existing circuit.

Settings: ISettings (read only)

Return a pointer to the Settings interface. The Settings interface returns several of the most common option values.

Solution: ISolution (read only)

Return a pointer to the Solution interface.

SubstationLosses: OleVariant (read only)

Watt and var losses in all the Transformer elements in the circuit that are designated as substations.. Returned as a two-element array of doubles.

SwtControls: ISwtControls (read only)

Returns a pointer to the SwtControls collection interface, which can be used to access properties of SwtControls objects (Switch controls) currently defined in the existing circuit.

SystemY: OleVariant (read only)

Return the System Y matrix in complex numbers as a variant array of doubles.

Topology: ITopology (read only)

Returns an interface to the active circuit's Topology object.

TotalPower: OleVariant (read only)

Returns the total power in kW and kvar supplied to the circuit by all Vsource and Isource objects. Does not include Generator objects. Returned as a two-element array of doubles.

Transformers: ITransformers (read only)

Returns a pointer to the Transformers Collections interface, which can be used to access various properties of Transformer objects in the circuit.

Methods (Functions) in the Circuit Class

Note "Procedure" is equivalent to a void function in C, or Subroutine, or Sub in other languages.

Function Capacity(Start: Double; Increment: Double): Double;

Executes the DSS capacity function. Start is the per unit load multiplier for the current year at which to start the search. Increment is the per unit value by which the load increments for each step of the analysis. The program sets the load at the Start value for the PRESENT YEAR (including growth) and increments the load until something in the circuit reports an overload or undervoltage violation. The function returns the total load at which the violation occurs or the peak load for the present year if no violations.

```
MaxkWThisYear = DSSCircuit.Capacity(0.9, 0.005)
```

Procedure Disable(const Name: WideString);

Disable a circuit element by name (full name).

Procedure Enable(const Name: WideString);

Enable a circuit element by name (full name).

Function FirstPCElement: Integer;

Sets the first enabled Power Conversion (PC) element in the circuit to be active. If not successful returns a 0.

```
' Cycle through the PC elements
j = DSSCircuit.FirstPCElement
Do While j <> 0

  (whatever)

j = DSSCircuit.NextPCElement
Loop
```

Function NextPCElement: Integer;

Sets the next enabled Power Conversion (PC) element in the circuit to be active. If not successful returns a 0.

Function FirstPDElement: Integer;

Sets the first enabled Power Delivery (PD) element in the circuit to be active. If not successful returns a 0.

Function NextPDElement: Integer;

Sets the next enabled Power Delivery (PD) element in the circuit to be active. If not successful returns a 0.

Procedure Sample;

Forces all energy meters and monitors to take a sample.

Procedure SaveSample;

Forces all energy meters and monitors to save the samples that have been previously taken to disk.

Function SetActiveElement(const FullName: WideString): Integer;

This method sets the active circuit element by name. Use the fully qualified name such as “line.L1” or “transformer.T1”.

Function SetActiveBus(const BusName: WideString): Integer;

Sets the active bus by name. Returns a 0 based index of the bus to use for future direct indexing of bus values returned in arrays. Returns -1 if an error occurs.

```
Public TheActiveBus As DSSEngine.Bus

MyBusIndex = DSSCircuit.SetActiveBus(“Bus23”)
If MyBusIndex >= 0 Then
       Set TheActiveBus = DSSCircuit.ActiveBus
       If TheActiveBus.kVBase > 0.0 then …

End If

‘ Alternative way sans error check
Set TheActiveBus = DSSCircuit.Buses(“Bus23”)
```

Function SetActiveBusi(BusIndex: Integer): Integer;

Sets the active bus by integer index. The index is 0 based. That is, the first bus has an index of 0. Returns -1 if an error occurs.