OpenDSS COM Documentation

Solution Interface

The Solution interface contains many of the methods and properties one would use to control the solution process and implement custom solution and control algorithms. You should set a variable to the Solution 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
```

Then you would use the DSSSolution variable as follows, for example:

```
' Solve Power Flow
DSSSolution.Solve
If DSSSolution.Converged Then
    V = DSSCircuit.AllBusVmagPu
     …. (process busvoltages ) ….
' Now switch to harmonics mode and solve harmonic solution
    DSSSolution.Mode=dssHarmonic
    DSSSolution.Solve   'Solve for all harmonics
End If
```

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

Properties

AddType: Integer (read /write)

Specify which type of device to add for the AutoAdd function:

1. Add a generator object
2. Add a capacitor object

Algorithm: Integer (read /write)

0: Normal fixed-point iteration solution algorithm
1: Newton method solution algorithm (not to be confused with traditional Newton-Raphson power flow method). Used when it is necessary to solve more ill-conditioned problems.

Capkvar: Double (read /write)

Size of capacitor, in kvar, for capacitor to add in AutoAdd function.

ControlIterations: Integer (read /write)

Present value of the control iteration counter.

ControlMode: Integer (read /write)

Set the control mode by an integer value. One of

- dssStatic (default: typically used for most power flow studies)
- dssEvent
- dssTime

Set this to -1 to turn controls off.

Converged:Boolean (read/write)

Flag to indicate whether the previous circuit power flow solution converged.

dblHour: Double (read /write)

Present time expressed as a floating point number in units of hours.

DefaultDaily: WideString (read /write)

Name of default Daily LoadShape object.

DefaultYearly: WideString (read /write)

Name of default Yearly LoadShape object.

EventLog: OleVariant (read only)

This is a variant array of strings containing the present contents of the Event Log.

Frequency: Double (read /write)

Present value of the solution frequency, Hz.

GenkW: Double (read /write)

Size of the generator, in kW, to add with the AutoAdd function.

GenMult: Double (read /write)

Global multiplier applied to all generators for applicable solution modes.

GenPF: Double (read /write)

Power Factor (PF) of generator assumed for generator being added by AutoAdd function.

Hour: Integer (read /write)

Hour part of the present solution time. See also Seconds and dblHour. Combined with Seconds, gives the complete time value.

Iterations: Integer (read only)

Present value of the iteration counter for the circuit solution (not including the control iterations).

LDCurve: WideString (read /write)

Load-Duration curve name (a Loadshape object) for load-duration solution modes.

LoadModel: Integer (read /write)

One of

- dssPowerFlow (default) = 1
- dssAdmittance = 2

(See Options enumeration in the COM interface)

LoadMult: Double (read /write)

Multiplier applied to all loads in the system that are not marked as fixed. Set this to force a solution at a different load level. Usually, 1.0 = peak load, although the user can define the loads at different levels.

MaxControlIterations: Integer (read /write)

Maximum allowable control iterations. Use this property to retrieve or set the value.

MaxIterations: Integer read (read /write)

Maximum allowable number of iterations for the circuit solution (not counting control iterations).

Mode: Integer (read /write)

Set/Get the present solution mode expressed as an integer. See SolveModes enumeration in the COM interface. Keep in mind that setting the solution mode may have several side effects such as resetting meters and monitors or initializing for another form of solution. Note that when entering Harmonic, Dynamic, or FaultStudy modes, a converged solution of the circuit is required. If necessary, this may be forced by first solving in Direct mode if the normal iterative mode will not converge.

ModeID: WideString (read only)

This property gives the string name for the present solution mode.

Number: Integer (read /write)

The number of solutions to perform for modes such as Daily and Yearly. Note that for applications that wish to interact with the solution loop for the Yearly mode, for example, this property should be set = 1 so that the driving program can retrieve solution data between time steps.

pctGrowth: Double (read /write)

Default percent growth for the loads. Each load may override this by assigning a GrowthShape object to the load.

Random: Integer (read /write)

Specifies the type of random number generator to use for applicable analyses. One of:

- dssGaussian
- dssUniform
- dssLogNormal

(See Options enumeration.)

Seconds: Double (read /write)

Seconds part of the present solution time as a floating point number. See also Hour and dblHour.

StepSize: Double (read /write)

Get/set time step size in seconds.

```
DSSSolution.StepSize = 0.0002 ' for Dynamics
DSSSolution.StepSize = 900 '15-min demand interval data
```

StepsizeHr: Double (write only)

Specify time step size in hours (floating point).

```
DSSSolution.StepsizeMin = 0.25 ' 15-min demand interval data
```

StepsizeMin: Double (write only)

Specify time step size in minutes (floating point).

```
DSSSolution.StepsizeMin = 1.0 'solar plant data
```

SystemYChanged: WordBool (read only)

This is a Boolean flag that indicates whether the elements of the system Y matrix have changed since the last build. This is generally due to editing of object properties or to control action. See BuildYMatrix method.

```
With DSSSolution
```

```
        .SolveNoControl
        .SampleControlDevices
        .DoControlActions
        If .SystemYChanged Then .BuildYMatrix 2, 0
End With
```

Tolerance: Double (read /write)

Solution tolerance for circuit solution. Typically 0.0001 by default.

Year: Integer (read /write)

Get/set the value of the present Year for yearly solutions. Typically, Year=1 represents the base year, but other numbers may be used. Note that setting this property can result in many side effects, such as closing the demand interval files, resetting energymeters, monitors, etc.

Methods

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

Procedure BuildYMatrix(BuildOption: Integer; AllocateVI: Integer);

Forces a build of the system Y matrix.

BuildOption:

1. Series elements only (for zero load snapshot solution)
2. Whole Y matrix

AllocateVI:

0: Do not reallocate system Voltage (V) and current (I) arrays
Else: Reallocates V and I arrays

Procedure CheckControls;

This method is the normal DSS algorithm for checking and executing all control actions in the midst of the solution loop. The procedure is currently implemented as shown below. Besides the control objects, it triggers the status checking of all Fault objects. Finally, if any of the control actions caused a change in the System Y matrix (SystemYChanged variable) it forces a rebuild of the system Y matrix without reallocating the system Voltage array. Note that the control action checking is only performed if the solution is not on the final control iteration. In the built-in solution process, this helps avoid getting the solution and the Y matrix out of synch. The routine also checks to see if the most recent circuit solution converged (ConvergedFlag). If not, no control actions are done.

Note that the various functions this routine calls are also available for direct access from the Solution interface.

```
PROCEDURE TSolutionObj.CheckControls;

Begin
      If ControlIteration < MaxControlIterations then Begin
           IF ConvergedFlag Then Begin
               If ActiveCircuit.LogEvents Then
               LogThisEvent('Control Iteration ' + IntToStr(ControlIteration));
               Sample_DoControlActions;
               Check_Fault_Status;
           End
           ELSE
               ControlActionsDone := TRUE; // Stop solution process if failure
to converge
       End;

       IF SystemYChanged THEN BuildYMatrix(WHOLEMATRIX, FALSE); // Rebuild Y
matrix, but V stays same
End;
```

Procedure CheckFaultStatus;

Fault objects have ON times and may clear if the current gets interrupted. This function triggers a check of the status of each fault defined in the circuit at the present time.

Procedure DoControlActions;

This function pops the appropriate control actions off the control queue and dispatches them to the proper controller.

Procedure InitSnap;

This function initializes iteration counters, etc that occur at the beginning of the SolveSnap function. Invoke this method if you are implementing your own solution process. Calls the SnapShotInit function in the DSS.

Procedure Sample_DoControlActions;

This function combines the SampleControlDevices and DoControlActions functions. That is, it does the two function calls in the middle of the CheckControls method without doing the checking.

Procedure SampleControlDevices;

This function invokes the Sample function of all control devices. If you are implementing an external controller through the COM interface, you would want to explicitly call this method and sample the circuit values for your controller(s) either before or after – prior to actually executing the control actions.

Procedure Solve;

This method is identical to issuing the “Solve” command in the DSS text interface. It execute the solution for the present solution mode and number of times. For example, if the present mode is Yearly with the default settings, the Solve method will execute 8760 power flow solutions following the yearly load shapes and saving the results in EnergyMeter and Monitor objects.

Procedure SolveDirect;

This method invokes the direct solution function inside the DSS directly. This is a non-iterative solution considering only the active sources (Vsource and Isource objects) and the admittances of all other objects. Compensation currents of loads and generators are not included. This is the function used for Mode=Direct or when Loadmodel=Admittance. It is also used for Harmonic solutions at non-power frequencies (Setting Mode=Harmonic has the side effect of setting Loadmodel=Admittance.) It may be necessary to execute this method when the iterative method fails to converge.

Procedure SolveNoControl;

This method solves the circuit in its present state, but does not check for control actions. If implementing your own control externally, call this function and sample the control values after its completion. Push control actions onto the Control Queue (CtrlQueue interface), then call CheckControls. After this, pop actions off the action list in the CtrlQueue interface and execute the requested action in your controller implementation.

Procedure SolvePflow;

This method invokes the iterative power flow solution function inside of the DSS directly.

Procedure SolvePlusControl;

This method executes the SolveCircuit and CheckControls part of the solution process. See SolveSnap method.

Procedure SolveSnap;

This is a direct call to the standard intrinsic function for a solving the circuit, including control actions, for a given snapshot in time. The present implementation inside the DSS is shown below. This is the template for user-defined solution algorithms.

```
FUNCTION TSolutionObj.SolveSnap:Integer;  // solve for now once

VAR
   TotalIterations  :Integer;

Begin
   SnapShotInit;
   TotalIterations    := 0;

   REPEAT

       Inc(ControlIteration);

       Result := SolveCircuit;  // Do circuit solution w/o checking controls

       {Now Check controls}
       CheckControls;

       {For reporting max iterations per control iteration}
       If Iteration > MostIterationsDone  THEN MostIterationsDone := Iteration;
```

```
       TotalIterations := TotalIterations + Iteration;

   UNTIL ControlActionsDone or (ControlIteration >= MaxControlIterations);

   If Not ControlActionsDone and (ControlIteration >= MaxControlIterations)
then  Begin
       DoSimpleMsg('Warning Max Control Iterations Exceeded. ' + CRLF + 'Tip:
Show Eventlog to debug control settings.', 485);
       SolutionAbort := TRUE;   // this will stop this message in dynamic power
flow modes
   End;

   If ActiveCircuit.LogEvents Then LogThisEvent('Solution Done');

   Iteration := TotalIterations;  { so that it reports a more interesting
number }

End;
```