OpenDSS COM Documentation

CtrlQueue Interface

Sept 2012

The CtrlQueue interface contains properties and methods for interacting between internal DSS control objects and control algorithms written in external programs.

DSS Control Queue

The OpenDSS program has an internal control dispatching process implemented as a queue of action requests at specified times. This control queue is primarily for managing *discrete* controls such as tap-changing voltage regulators and switched capacitors that can have delayed actions. For example, when a voltage regulator goes out of band, it pushes a message on the Control Queue with a requested time for a tap-changing action. This simulates starting the timer in the control. When the requested time is reached, the message is sent back to the regulator, which decides at that time whether or not it really needs to change taps. This simulates the timer expiring and putting a signal on an AND gate with the out-of-band signal, which is the way some regulator controls work.

Continuous controls generally act directly on variables within the program and any delay is accomplished through time integration. Continuous controls are primarily used in dynamics simulations while discrete controls are mainly for sequential power flow simulations. They are often implemented by user-written DLLs.

The Control Queue gets populated after a converged solution is achieved. Each of the control objects currently enabled in the circuit are polled. If they determine that they need to make a delayed change, they will push a control action onto the Control Queue using codes that are unique to each class of control object. When it comes time to execute the action, the OpenDSS pops the appropriate control actions off the Control Queue and dispatches each of them to the appropriate control handler (a virtual function called *DoPendingAction* in the module).

The CtrlQueue COM interface instantiates a proxy that provides a DoPendingAction function to the OpenDSS and, thus imitates an internal OpenDSS control object.

COM Interface Control Proxy

The COM interface contains a proxy for any control objects that might be implemented in some program external to the OpenDSS and driving the OpenDSS through the COM interface. The COM Control Proxy contains an Action List that contains a list of actions that have been popped off the OpenDSS Control Queue (see Figure 1). User-written control algorithms should check this List and execute the requested action, if it is still appropriate to do so.

The Action List dispatches control requests using a Device Handle and an Action Code. Controls implemented in external languages via the COM interface should be associated

with a particular Device Handle for dispatching by the COM Control Proxy object. At this time, Device Handles and Action Codes are defined in the external user-written code, but a future version may also employ handles and codes managed by the simulator.

Typically, an external control would step through the solution one step at a time. After achieving a converged circuit solution, the control would execute the functions that sample the quantities of interest. It could then *Push* any control actions onto the OpenDSS Control Queue and rely on the OpenDSS to dispatch the control. The OpenDSS would know the action request came from the COM Control Proxy and dispatch any actions back to the Proxy where they are accumulated in the Action Queue. The user-written control would then be responsible for checking the Action List, setting the Active Action (directly or by popping off the list in sequence), and then dispatching the action code to the proper control algorithm.

All discrete external controls with time-delayed actions should employ the OpenDSS Control Queue for proper synchronization with other active controllers in the circuit. When the time comes to execute an action, external controls may act directly upon circuit objects through the COM interface. For example, if a user-written relay control determines that a device terminal needs to be opened, it can do so through either the text interface or the CktElement interface.

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

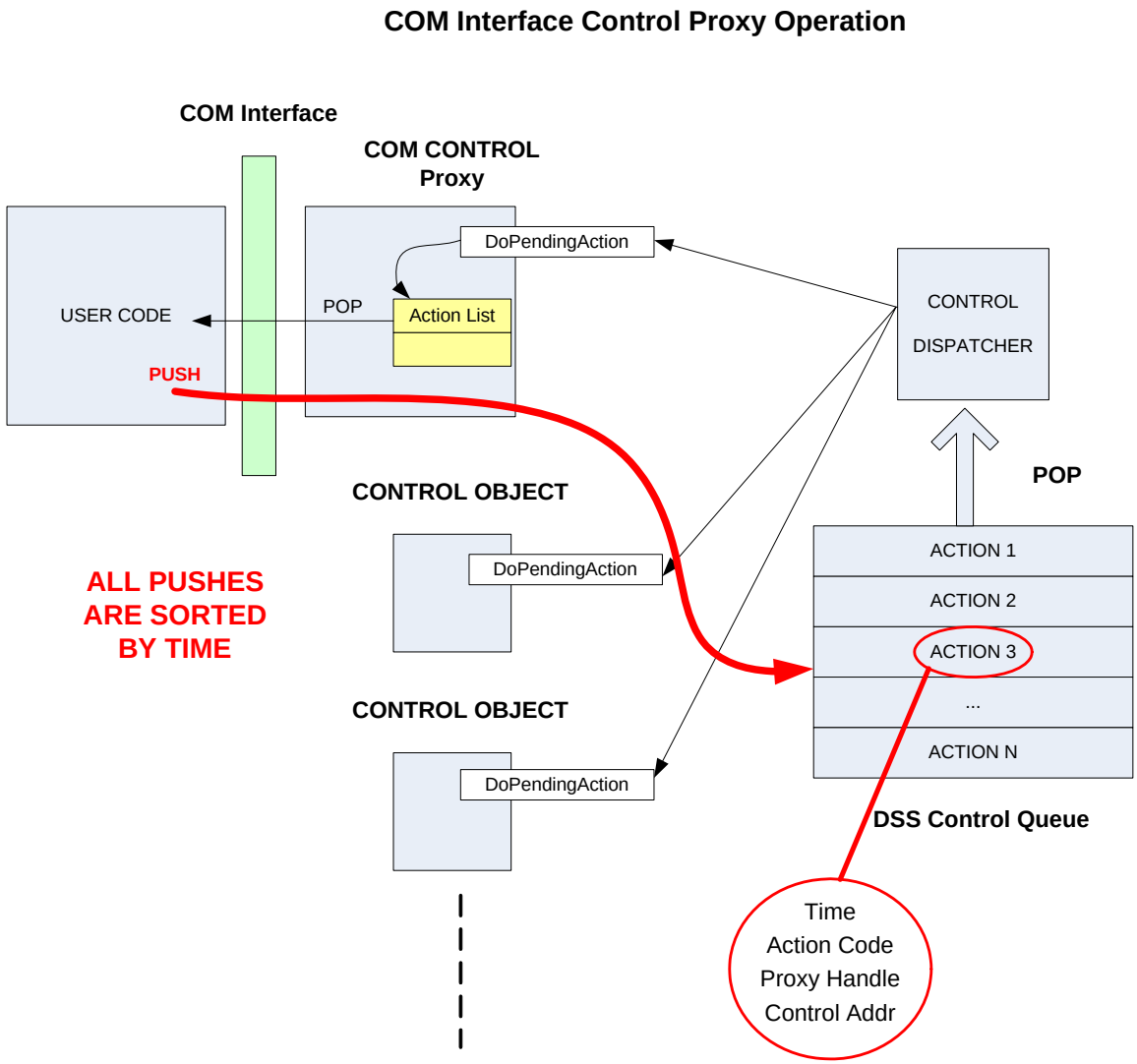

Figure 1. Illustration of how the COM Control Proxy interacts with the DSS Control Queue

Properties

NumActions:Integer (Read only)

Returns the number of actions in the COM Control Proxy Action List. Check this value to see if the OpenDSS control queue has popped any actions off the list that are destined for the user-written control

Action(Index: Integer): (Write Only)

One way to set which of the Actions in the Action List the *Active Action*. Sets it by index value after checking to make sure that Index is within the valid range (based on NumActions).

ActionCode: Integer: (Read Only)

Action Code for the Active Action in the COM Control Proxy Action List. Use this to determine what user-defined controls are supposed to do. Can be any long integer of the programmer's choosing.

DeviceHandle:Integer (Read only)

Returns a handle to the user's control device from the Active Action.

PopAction: Integer (Read Only)

Sets the Active Action by popping the next action off the COM Control Proxy action list.

Methods

Function ClearQueue: HResult;

This clears all actions from the DSS Control Queue. The return value is always zero.

Function Delete(ActionHandle: Integer): HResult

Deletes an Action from the DSS Control Queue by the handle that is returned when the action is added. (The Push function returns the handle.) The return value is always zero.

Function Push(Hour: Integer; Seconds: Double; ActionCode: Integer; DeviceHandle: Integer): HResult;

Pushes an Action onto the DSS Control Queue. The return value is the handle to the action on the queue. Specify the time of the requested action in Hour, Seconds. Push an ActionCode that is meaningful to the user-written control object. DeviceHandle is a user-defined handle to the control object that will be used to dispatch the control action to the proper control device when the Action is popped off the Control Queue. The DSS will return this handle to the control proxy in the COM interface when the action is popped.

Function Show: HResult;

This function executes the procedure inside the OpenDSS to show a text file in CSV form containing the present contents of the DSS Control Queue. The return value is always zero.

When this command is complete the Result property of the Text interface contains the name of the file, which you may use for further processing. (The Result property must be accessed before doing anything else.)

Function ClearActions: HResult;

This clears all actions from the Control Proxy's Action List (they are popped off the list). The return value is always zero.

Example 1

The following MATLAB snippet demonstrates how to execute a single solution by performing the solution and control sampling separately. After polling all the control elements (SampleControlDevices) the contents of the CtrlQueue are displayed. This example uses the IEEE 123 bus test feeder.

```
[DSSStartOK, DSSObj, DSSText] = DSSStartup;
if DSSStartOK
    DSSText.command='Compile (C:\opendss\IEEETestCases\123Bus\IEEE123Master.dss)';
    % Set up the interface variables
    DSSCircuit=DSSObj.ActiveCircuit;
    DSSSolution=DSSCircuit.Solution;

    DSSText.Command='RegControl.creg1a.maxtapchange=1  Delay=15  !Allow only one tap
change per solution. This one moves first';
    DSSText.Command='RegControl.creg2a.maxtapchange=1  Delay=30  !Allow only one tap
change per solution';
    DSSText.Command='RegControl.creg3a.maxtapchange=1  Delay=30  !Allow only one tap
change per solution';
    DSSText.Command='RegControl.creg4a.maxtapchange=1  Delay=30  !Allow only one tap
change per solution';
    DSSText.Command='RegControl.creg3c.maxtapchange=1  Delay=30  !Allow only one tap
change per solution';
    DSSText.Command='RegControl.creg4b.maxtapchange=1  Delay=30  !Allow only one tap
change per solution';
    DSSText.Command='RegControl.creg4c.maxtapchange=1  Delay=30  !Allow only one tap
change per solution';

    DSSText.Command='Set MaxControlIter=30';

    % Solve executes the solution for the present solution mode, which is "snapshot".

    DSSSolution.SolveNoControl;
    disp(['Result='  DSSText.Result])
    if DSSSolution.Converged
       a = 'Solution Converged';
       disp(a)
    else
       a = 'Solution did not Converge';
       disp(a)
    end
```

```
    DSSText.Command='Export Voltages';
    disp(DSSText.Result)

    DSSSolution.SampleControlDevices;
    DSSCircuit.CtrlQueue.Show;
    disp(DSSText.Result)
    DSSSolution.DoControlActions;
    DSSCircuit.CtrlQueue.Show;

    DSSText.Command='Buscoords Buscoords.dat   ! load in bus coordinates';
else
    a = 'DSS Did Not Start'
    disp(a)
end
```

Example 2

{Still under construction}