Scripting Example & Tutorial

Jeff Lunglhofer
Jeff@Lunglhofer.com

Several folks requested an example of how to build custom scripts that can be run from CCD Commander to perform custom functions that can't be associated with the current suite of available actions. Since CCDC gives you the ability to run external scripts and programs, that provides the perfect opportunity to effectively generate your own "custom actions" by creating some simple scripts.

Here is a sample script that I run to attempt to shut down my observatory. It does the following things:

1) Verifies that the roof control ASCOM driver is running and connected
2) Verifies that the telescope is connected to the PC
3) If the roof control is connected, but the telescope is NOT, and the roof is OPEN, the script will try to close the roof
4) If the roof is OPEN and the telescope is NOT PARKED it will park the telescope
5) The script will wait for the telescope to park successfully
6) The script will close the roof

Here's the script. This is saved as a .vbs file on my observatory control PC. I'll drop in comments to explain roughly what each step is doing. Comments are created by placing a single quote before the comment.

****

```
‘ These “Dim” statement just create variables that we can use in our scripts
Dim Telescope
Dim Roof_Position
Dim Dome
‘ The ASCOM driver for my roll-roof uses the ASCOM “dome” class, since there is no “roll roof” class

‘ Now we need to link the Telescope and Dome variables to the ASCOM drivers.
‘ The exact name of the ASCOM driver will vary depending on who wrote the driver and what
‘ they named it.
Set Telescope = CreateObject ("ASCOM.GeminiTelescope.Telescope")
Set Dome = CreateObject("ASCOM.StampObservatory.Dome")

‘ After I perfect the script I put this line in, which allows the script to continue running
‘ even if an error is encountered.  When creating a new script, comment out this line
‘ so any errors will be thrown and you’ll get an error message telling you where the
‘ error occurred.
On Error Resume Next

‘ So this is the first line that does anything interesting.  Here we are assigning the value of
‘ the ShutterStatus property to the variable “Roof_Position”.  A property is just like a variable
‘ that is maintained by the ASCOM driver.  Some you can just READ (like we’re doing here)
‘ and others you can actually change.
Roof_Position = Cstr(Dome.ShutterStatus)

‘ If the previous line throws an error that means the roof control ASCOM driver isn’t working
‘ so we can’t do anything. . .so we will quit.  We will flush out the links to the ASCOM drivers
‘ also, those are the “= nothing” statements.
If Err.number <> 0 Then ‘ Only run if there is an error
        MsgBox "Roof Control Not Enabled"
        Set Dome = nothing
        Set Telescope = nothing
        WScript.Quit
End If

‘ Now we will WRITE to the Connected property on the telescope to establish a connection
‘ to the ASCOM driver.  Note that the statement above just created a link, this actually
‘ establishes a connection to the ASCOM driver.  As confusing at this may be, this does NOT
‘ actually connect the driver to the mount, it just connected this SCRIPT to the driver.  I
‘ personally find this super confusing. . .but whatever!
Telescope.Connected = True

‘ If we get an error here, that means the telescope is having problems or not running,
‘ so we will just go ahead and close the roof.  Roof_Position is a variable that we associated
‘ with the “Dome.ShutterStatus” previously.  The values are defined as “0” for OPEN and
‘ “1” for CLOSED.  Note that here we also will use a METHOD, specifically the “CloseShutter”
‘ method.  A method a specific action that you can instruct the ASCOM driver to do, sometimes
‘ you can include additional instructions.  Methods are documented in the ASCOM documentation
‘ and/or by the ASCOM driver author.
If Err.number <> 0 Then ‘ Only run if there is an errror

        If Roof_Position = 0 then ‘ If our roof is OPEN
                Dome.CloseShutter ‘ CLOSE it
                Set Dome = nothing
                Set Telescope = nothing
                WScript.Quit
        End if
End If

‘ At this point we’ve encountered no errors and are ready to perform the normal
‘ script operations

If Roof_Position = 0 then  ‘ If the roof is OPEN
        if Telescope.AtHome = False then ‘ and the telescope is NOT parked
                Telescope.Park ‘ Park the telescope
                Do While Telescope.AtHome = 1 ‘ While the telescope isn’t parked
                        WScript.sleep (100) ‘ Wait 1 second and check again
                Loop
        End if
        Dome.CloseShutter ‘ Now the telescope is parked, so we can close the roof safely
```

```
          Set Dome = nothing
          Set Telescope = nothing
          WScript.Quit
End if

' So the only way we can get here is if everything was working as expected, but
' apparently the roof was already closed.  So just to be safe, we will go ahead
' and park the telescope before we quit

Telescope.Park
Set Dome = nothing
Set Telescope = nothing
```

****

That was a fairly complex example, but simpler scripts are easier to write. . .take this one for example:

```
Dim Telescope
Set Telescope = CreateObject ("ASCOM.GeminiTelescope.Telescope")
Telescope.Connected = True
Telescope.Park
Set Telescope = nothing
```

****

That simple script will just park your telescope. . .something similar could be used to do other "strange" functions, such as setting your park location as the current telescope location:

```
Dim Telescope
Set Telescope = CreateObject ("ASCOM.GeminiTelescope.Telescope")
Telescope.Connected = True
Telescope.SetPark
Set Telescope = nothing
```