Chapter 14
Scripts

In previous chapters, Home Modes, the Visual Programmer and the Visual Scheduler were covered in detail. These three tools allow for the creation of sophisticated automation solutions.

However if you are looking for more complexity in programming and decision making, HCA provides for the inclusion of text based scripts that are used as an adjunct to Visual Programs.

What scripting languages are supported? HCA supports any script engine that can use a type library. This means that common ones like VBScript and JavaScript are supported as well as the less common ones like WebBuilder.

What can a script do and how does it do it? HCA provides a rich object model with many methods available for each object. This is all documented in the HCA Object Model appendix to this user guide.

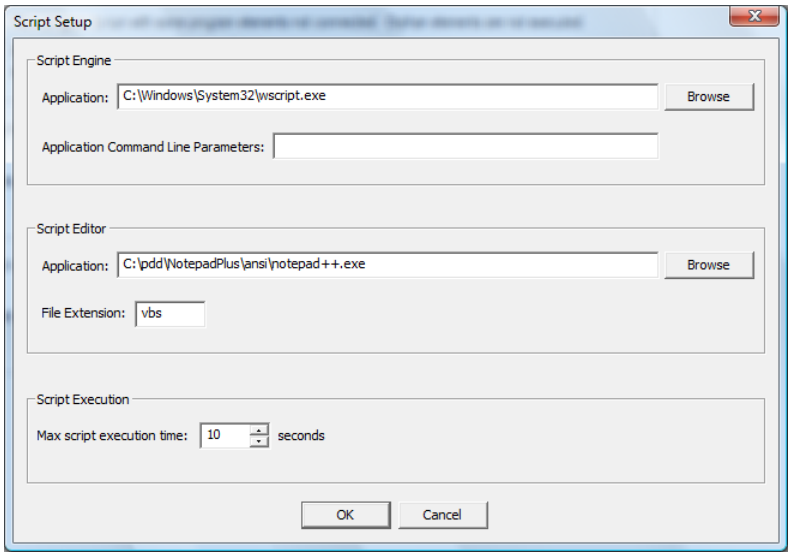
This chapter focuses on how you configure HCA to work with text based scripts, the Visual Programmer Script element, and some thoughts on how scripts are best used.

This is a complex topic! Writing scripts is a true programmer activity and, unlike Visual programs, doesn't have a friendly environment to work in. Included in the HCA installation are script examples you may want to look at prior to starting any of your own.

NOTE: This feature is only available in HCA Plus and does not function in HCA Standard or HCA Limited.

Script setup

Before you can begin using scripts you first must configure HCA for what script engine and editor to use. To do press the *Scripting* button in the ribbon *Tools* category.

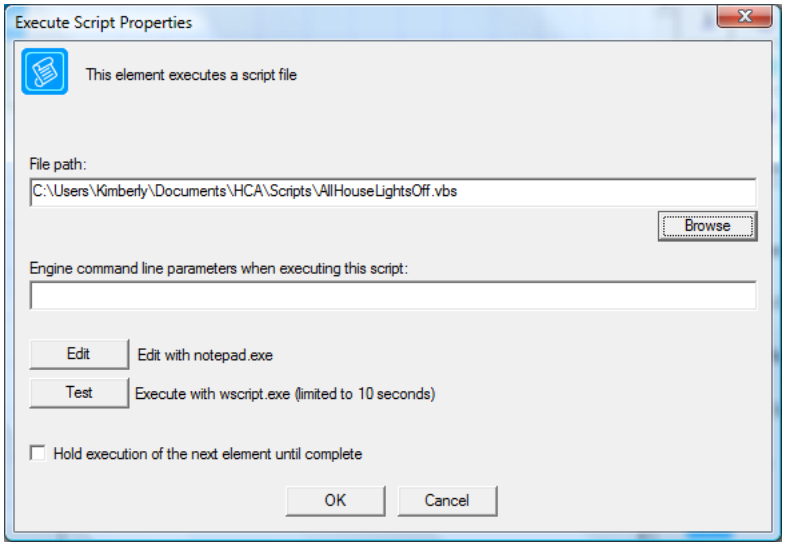


In this dialog are these parameters:

- Script Engine.
  This is the application that will process the script. In the above screen image VBScript is being used.
- Command line parameters to the script engine.
  This will vary between engines
- Script editor.
  In the Script Visual Programmer element is a button that you can use to start an editor to open your script. This may make it simpler to make quick changes as you modify your script to achieve the function desired. In the above screen image the text editor Notepad++ is being used.
- The File extension for your scripts.
  This makes locating your scripts in the Visual Programmer Script element simpler when you use the browse button.
- Script execution time.
  Each script is executed in its own windows process and HCA watches for it to terminate. Due to programming errors, your script might not complete its task in a timely manner – or it may never complete its task. HCA watches for it to complete within the allotted time and if it does not then HCA terminates the process.

The script element

To use a script in a Visual Program, add the Script element.



In this dialog is specified the path to the script file. Use the browse button to locate it. Script files can reside anyplace on your disk but it is recommended that you keep your scripts in the HCA area of your documents in the sub-folder Scripts.

There are two buttons on this dialog:

- Edit.
  This starts whatever text editor you have previously configured in Script Setup. The name of the application is shown here for your reference.

- Test.
  This starts the script engine to execute your script. Whatever action you have developed your script to do will then happen. The script engine and the max execution time is what was previously configured in Script Setup. The name of the engine and max execution time is shown here for your reference.

Also in this dialog is a checkbox to say if the Visual Program execution continues with the next element immediately or waits until the script is complete.

Using scripts

Why use a script when Visual Programs have so many different elements that you can use? There may be times where it simply is convenient to use a script rather than a visual program just because of the number of elements you would need or your need to be kept up to date as your design changes.

A good example of this is shown in a sample script included with the HCA installation. This is the script AllHouseLightsOn.vbs

```
Option Explicit

Dim HCA
Set HCA = GetObject(,"HCA.Object")

   Dim count
   Dim i
   Dim name

   count = HCA.device.count()

   For i = 0 To count - 1

      name = HCA.device.Name(i)

      if (HCA.device.SupportsDim(name)) then
         HCA.device.On (name)
      end if

   Next

Set HCA = Nothing
```

This simple script turns on all the lights in your home. You may have one light or one hundred it doesn't matter. If you did have a hundred lights it would be a very large Visual Program and you would have to update it each time you added or removed a light. With this script it is much simpler.

Another reason is to perform some kind of test that HCA can't do. For example, this script determines if today is the 3rd Monday of the month.

```
Option Explicit

' Sets the rcScript variable in HCA to one if
' today is the 3rd Monday of the month
' Otherwise sets it to zero

Dim HCA
Set HCA = GetObject(,"HCA.Object")

Dim date
date = Now()

Dim y, m, d
y=Year(date)
m=Month(date)
```

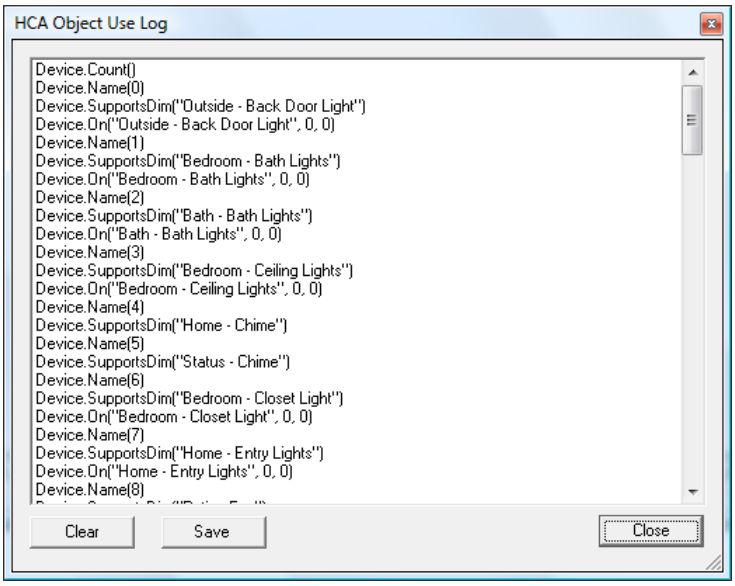
```
d=Day(date)

dim ith
ith=3

dim i
i = 1

do while (ith > 0)

   date=DateSerial(y,m,i)

   if (Weekday(date) = vbMonday) then
      ith=ith-1
      if (ith = 0) then exit do
   end if

   i = i + 1

loop

Dim rc
if (i = d) then
   rc = 1
else
   rc = 0
end if

HCA.Flag.Set "rcScript", rc
Set HCA = Nothing
```

This is a test that HCA can't perform with the Visual Programmer Test element.

Helpful Tool

Included in HCA is a helpful tool in working with scripts. The feature is normally hidden a bit as it wouldn't make sense to HCA users not working with scripts. To expose this tool, open the application and press the *HCA Options* button. Select the *Extra* tab and enter the code OBJUSE. Once you have done that, then open the Object Use Log with the *Remote Access Viewer* button in the ribbon *Control* category.

When open, the viewer shows all the calls to any HCA object and method as well as its parameters and if the method generates an error it shows that also.



It’s not really a “log” as such it is more of a Viewer as things only “log” when the viewer is open.

The viewer can be cleared with the *Clear* button and the contents saved to a text file with the *Save* button.

Instead of using the Extra feature code OBJUSE you can use the code OBJERR. Instead of showing all the calls to the object method, only those methods that result in an error are shown.