ArchestrA Script Creator User Manual

October 2014

Usage

The ArchestrA Script Creator is used to create a long ArchestrA script given a text file containing field attributes. The script that is created is a script that is used by me to create relative referencing tag names based on the reference list in a DI Object. See the "Example aaCreateScript Field Attributes.txt" file and the "Example aaCreateScript Script.txt" file for a better understanding. You give the Field Attributes text file, you run the script and you get the script file.

Prerequisites

The script is written in Python version 3.4. Python 3.4 is an easy to use programming language that is available free to download and easy to install and work with. You will need the following:

1. Python 3.4 installed on your PC
2. Python added to your path
3. The aaCSV.py script
4. The exported CSV file for your object

Example

1.1 Run the program

Open up the command prompt and change the directory to the path where your aaCreateScript.py file resides.

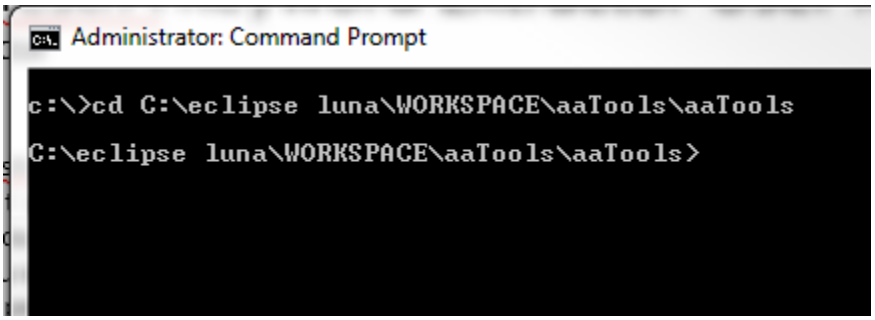

Figure 1.

Type in the program name followed by the input text file (containing the field attributes) followed by the new file name you want the script to be in. See figure 2.

Figure 2.

Press enter and your file will be created.

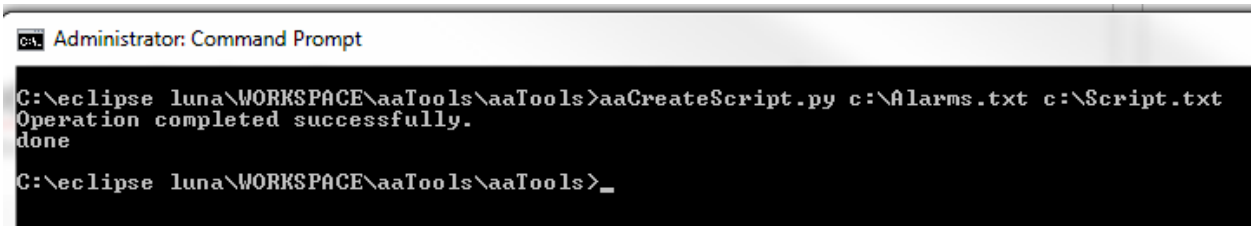

Figure 3.

The typical input file had 87 Field Attributes that looked something like these 5 Field Attributes

```
FA_ALM_ACT_CLOSE_NO_FB
FA_ALM_ACT_CLOSE_TORQUE
FA_ALM_ACT_FAULT_RELAY
FA_ALM_ACT_OPEN_NO_FB
FA_ALM_ACT_OPEN_TORQUE
```
Figure 4.

The newly created script text file contents will look something like this, for each of the 87 Field Attributes:

```
'Wait two execution counts to make sure script executes
If Me.scrAssignIO.ExecutionCnt > 2 Then

'Declare variables

    DIM Datasource AS STRING;
    DIM AliasDB AS INDIRECT;
    DIM FA_ALM_ACT_CLOSE_NO_FB_Exist AS INTEGER;
```
Figure 5. Declaration part.

```
'Datasource is the input/output source you want to allocate to your field attribute
   Datasource = Me.PLCPath + "." + Me.Tagname;

'Create the alias list that is in the Device Integration object using the PLCPath uda to make it more dynamic
   AliasDB.bindto(Me.PLCPath + ".AliasDataBase");
   'The exist variables will return an integer value when the string exists in the alias database in the DI object
   'A string comparison is done between the Alias DB list and the Attribute to see if the attribute exists in the Alias DB
   'The StringChar function looks at the ASCII value for a quote which seperates the alias tags from each other in the aliasDB

   FA_ALM_ACT_CLOSE_NO_FB_Exist = StringInString(AliasDB, (StringChar(34) + Me.TagName + ".FA_ALM_ACT_CLOSE_NO_FB" + StringChar(34)),1,0);
```
Figure 6. See if reference exists in the DI Object

```
 'The value is then assigned to the DIN Field Attribute input source or unassigned if it is not in the AliasDB
 If FA_ALM_ACT_CLOSE_NO_FB_Exist > 0 Then
    Me.FA_ALM_ACT_CLOSE_NO_FB.Input.InputSource = Datasource + ".FA_ALM_ACT_CLOSE_NO_FB";
 Else
    Me.FA_ALM_ACT_CLOSE_NO_FB.Input.InputSource = "---";
 EndIf;
```
Figure 7. Assigning the reference tag name if it exists.

```
'The script is then finished by setting the trigger bit to true and it will stop executing
    Me.AssignIOCmd = True;

EndIf;

'A Log Message flag is added to see if the script executed in the SMC - This can be excluded and is for troubleshooting
'LogMessage (Me.Tagname + "Finished");
```
Figure 8. End of the script.

That is it. Saved a lot of typing and copying and pasting. Now you can copy and paste this script into your ArchestrA script.

See the two example text files for better understanding of what this script does. You can modify and adapt the script to your scripts.

End.