# **Automation Engine**

# Scripting in Automation Engine





## Contents

1. Scripting	3
2. Automation Engine Script Runner	4
3. Installing Script Runner	5
4. Configuring Script Runner	7
5. Scripting for Script Runner On Mac OS.	8
5.1 Using AppleScript on Script Runner	8
5.1.1 Using AppleScript On Script Runner: A Sample Case	11
5.2 Using Shell Script on Script Runner	14
6. Scripting for Script Runner on Windows	
6.1 Using Windows Script on Script Runner	
6.1.1 Using Windows Script On Script Runner: A Sample Case	21
6.2 Using Batch File for Scripting on Windows	
7. Using ExtendScript on Script Runner(On MacOS/Windows)	
7.1 Using ExtendScript on Script Runner: A Sample Case	
8. Appendix : Script Samples	
8.1 AppleScript Code Samples	
8.2 Shell Script Code Sample	
8.3 Windows Script Code Samples	
8.4 Batch File Code Sample	
8.5 ExtendScript Code Samples	

# 1. Scripting

You can write Scripts (small programs) to automate the execution of certain tasks during a workflow. Writing and using such scripts is called scripting. You need the Script Runner application to link scripting with your Automation Engine workflows. You can automate actions from Adobe applications (e.g. Illustrator, Photoshop, InDesign) and third party tools (e.g. Alwan) using Scripts.

Some of the instances where scripting is useful are given below:

- to ensure file format integrity (standardization) for workflow inputs.
- to use standardized PDF as input during the Preflight process.
- to automate Adobe Illustrator, Photoshop, InDesign and InDesign Server via ExtendScript on Mac Os and Windows.

#### **Supported Script Types**

- AppleScript (on Mac OS)
- ShellScript (on Mac OS)
- Batch files (on Windows)
- Windows Script (VBScript, JScript) (on Windows)
- ExtendScript (on Mac OS and Windows)

More info on Scripting :Getting Started with Scripting

More info on Scripting in workflows: Use case: Scripting

## 2. Automation Engine Script Runner

The Automation Engine(AE) Script Runner is a standalone AE server component which runs scripts on behalf of AE. You can add customization to your workflow by adding a **Run Script** task. This task will run on the Automation Engine server while the execution of the script will be done on the Script Runner application which can be installed on Windows or Macintosh. When you launch the workflow, the following will take place:



- 1. The Run Script task sends a request to run the specified script.
- 2. The Script Runner processes the request accordingly and runs the script.
- 3. The Script Runner sends the results back to the server.
- 4. The workflow will continue with the outputs from this task.

# 3. Installing Script Runner

Scripting helps to customize some steps in your work flows. You can achieve this by adding a **Run** Script task to the work flow where you want customization. The Automation Engine will use Script Runner to run scripts which are stored either locally or on a server. When you launch the workflow in Automation Engine, the Script Runner runs the script which contains the main function incorporating the inputs from the Run Script task, output folder and some optional script parameters from the **Run** Script ticket. The workflow in Automation Engine continues with the contents of the output folder which contains the outputs from the script. To achieve this, you need to do the following steps :

1. Download Automation Engine Script runner. You can download it via the web access to the Automation Engine Server. Client Apps > Tools .



- On Mac OS, open the 'dmg' file after downloading the Script Runner which contains the installer package. Double-click to start the installation.
- On Windows, double-click the downloaded installer to start the installation.
- **2.** Install the Script Runner by following the instructions in the Installshield Wizard/Assistant and make sure it is running in your computer.
  - On Windows, open Start > All Programs > Esko > Automation Engine Script Runner > Preferences.
  - On Mac OS, open Applications > Automation Engine Script Runner > Esko > Automation Engine Script Runner > Script Runner Preferences.



In the Script Runner Preferences window, you can:

- check if the Script Runner is actually running
- start / stop the Script Runner
- enable(disable) Start at login
- view and change the port the Script Runner is communicating with
- view and change the default folders for scripts

	Start at login		
	Port	1083	
	Port	1303	
	Batch File Folder:	C:\Esko\bg_data_fastserverscrrunnt_v100\Scripts\Batch	Browse
Ð	xtendScript Folder:	C:\Esko\bg_data_fastserverscrrunnt_v100\Scripts\Extend	Browte
Wind	dows Script Folder:	C:\Esko\bg data fastserverscrrunnt v100\Scripts\Windov	Browte

## 4. Configuring Script Runner

You can configure your Script Runner to your Automation Engine server. This computer must have the Script Runner running.

For more information on installing a Script Runner, refer to Getting Started with Scripting

- 1. Choose Tools > Configure .
- 2. Choose Scripts from the categories on the left. Choose File > New.

Configure		
File Edit		
AE     AE	Host: AW12DL303 Port: 1983 Available Scripts on EAW12DL303: Type A ExtendScript ExtendScript ExtendScript ExtendScript ExtendScript ExtendScript ExtendScript ExtendScript ExtendScript Windows Script	Test Connection           Name           EskoAlPrint.jsx           EskoIDDPrint.jsx           EskoIDDSaveAsPDF.jsx           EskoIDServerPrint.jsx           EskoIDServerSaveAsPDF.jsx           EskoIDServerCaluSFunction.vbs
A Notification     Odystar     Odystar     Preflight Tools     Processing Preferences     Products     Scripts     BEOUMAC     Wostname_6     BEOU     BEOU     BEOU     MAAAMAAC	Available Scripts on Automation Engine: Type  Batch File ExtendScript ExtendScript ExtendScript Windows Script Windows Script	Name XPR.bat EskoIDDSaveAsPDF.jsx EskoIDDSaveAsPDF_noparams.jsx EskoAcroCalIJSFunction.vbs EskoAcroPrint.js
Hostname_7		

- 3. Give a suitable name to the Script Runner by choosing File > Rename
- 4. Enter the computer name or IP address of the Script Runner computer in the Host field.
- **5.** Enter the **Port** used to connect this computer to your Automation Engine server. By default, this is 1983.
- 6. Click the Test Connection button. If you have configured the Scrpt Runner successfully, you will see either all the scripts available on the Script Runner or a message 'No scripts available'.



# 5. Scripting for Script Runner On Mac OS.

A Script Runner on Mac OS supports AppleScript and Shell Script to automate operations.

#### Note:

Sample scripts are provided as-is with no warranty of fitness for a particular purpose. These scripts are solely intended to demonstrate techniques for accomplishing common tasks. Additional script logic and error-handling may need to be added to achieve the desired results in your specific environment.

It is up to the user to verify that his intended use of the offered automation functionality is compliant with any third party license agreement and/or other restrictions applicable to any non-Esko products.

## 5.1 Using AppleScript on Script Runner

AppleScript is a scripting language that makes direct control of scriptable applications and of many parts of the Mac OS possible. An AppleScriptable application is one that makes its operations and data available in response to AppleScript messages, called Apple events.

We recommend using AppleScript because:

- · it is highly integrated into the Mac OS
- it is supported by a lot of third party applications
- it has a high level of accessibility for scripting beginners
- 1. Open the AppleScript Editor and add following code.



Options	Description
main	This function will be called by the Script Runner. Only the code in this main function gets executed.
inputs	The first argument of the main function: a list of input file paths (type: list of strings).

Options	Description
outputFolder	Second argument of the main function: the folder where AE expects the script's result files. AE will continue the flow with the files you write in this folder. If you leave this folder empty, AE will continue the flow with the inputs of the Run Script task (type: string).
params	Third argument of the main function: additional script parameters injected into the script via the Run Script ticket (type: list of strings).
log	Extra log information in the Run Script task details
return "OK"	This will communicate with the Run Script task that everything went fine. Other possibilities are return "Warning" and return "Error".

2. Save this code as an AppleScript text file in the Script Runner's AppleScript folder (default: / Library/Scripts/Esko/AppleScript) or in the Automation Engine AppleScript folder.

Save	As: MyHelloWorld.applescript	
	AppleScript	; a
▼ DEVICES Mac HD iDisk ► SHARED ▼ PLACES Desktop	ColorSync Esko Folder Action Scripts Folder Actions Font Book FontSync Scripts	AppleScript  Cathering  AppleScript  Cathering  Catheri
	File Format: Text	•
	Line Endings: Unix (LF)	\$
	Options: Run Only Startu	ip Screen
	Stay Open	
New Folder	)	Cancel Save

**Note:** Script Runner supports 'Text' format. Therefore it is essential to change the file format to 'Text'.

**3.** You can add following code to test this script locally in the AppleScript Editor. Save the file and click Run to execute the script.

# 5 ESKO 😔



Notice the 'Hello, World!' and 'OK' result in the event log. The Script Runner does not attend to the test code in your script. It will execute the contents of the main function and ignore the rest. Therefore, you can keep your test code for future local testing.

 Open the Automation Engine Pilot. Go to Files view where you can select a file and open a New Task. Choose the Run Script task, modify its settings and launch the task.

0 0	New Task – Default
E Launch	Priority: Normal 🗘 🗌 Hold Ticket: 🗃 - Tickets: 🚆 -
Settings for Step	b: Run Script
>_ Out	put in: BROWSE []
File	Name: []
Run on: Script Type: Script File:	My Mac Script Runner  AppleScript  MyHelloWorld.applescript
Script Parame	eters

#### Read more in Run Script

Note that the 'Hello, World!' in the task details and 'OK' state are corresponding with log "Hello, World!"' and `return "OK"' in the script.

00			Tasks (RunScri	ptDemo)				
무 않	RunScriptDemo	:	- 10			(	2~	
Mode	Job		New Job				Search	
Mode VIEWS	job Name RunScriptDemo Task Type R Started on Server b Started as 5 Process Time 0 The scripts and Hello, World1 The scripts error o <none></none>	File Name Blairon, PDF-pdf un Script egezingem (23) 12 3:56 PM (23) 12 3:56 PM :00:00 rd output stream: utput stream:	New Joo Task Type Run Script	Progress 100%	Phase	State	Search Launched ¥ 5/23/12 3:56 PM	Task ID 3452
Uueue Monitor Containers Lustomers Users Hot Folders	Relaunch Move to	) History Delete						

### 5.1.1 Using AppleScript On Script Runner: A Sample Case

In this example, we use AppleScript to copy every input file with a size smaller than the size specified in the script parameters to the output folder.

We can use inputs, outputFolder and params in the AppleScript to achieve our objective. First, we demonstrate how to duplicate files without the size restriction and then proceed with the actual case.

1. Open the AppleScript Editor and add the code given below. This code is aimed to iterate through the list of inputs. It enables you to handle the inputs one by one, via the 'input' variable.

# 5 ESKO 😚



2. You can modify the Script as given below to duplicate the files to a specified output folder without size restrictions. Save this code as an AppleScript text file in the default AppleScript folder of Script Runner(default: /Library/Scripts/Esko/AppleScript) or in the Automation Engine AppleScript folder.

00	🐑 Myl	Duplicate.applescript	$\bigcirc$
0	<u>ک</u>		-
AppleScript	* <no element="" selected=""></no>	•	Bundle Contents
on main(inputs Translati set output	s, outputFolder, params) e the output folder (UNIX FolderReference <b>to</b> <i>POSIX</i>	oath) into an AppleScript file ref file outputFolder	erence
repeat wi Trai set in	th input in inputs nslate the input (UNIX pat putReference to POSIX file	n) into an AppleScript file refere input	nce
Dup tell ap end repea	blicate input to output folde oplication "Finder" <b>to dupl</b> replacing at	er i <b>cate</b> inputReference to outputf	FolderReference with
return "O end main 	К"		
Events	Renlies Result	*	
"OK"	meprice Result		
	Descrip	tion Event Log	

**3.** Add the file size check in the code as given below. This will duplicate the file when the input file size is smaller than the maximum size from the script parameters. If this condition is not met it will add an entry in the log and there will be "Warning". Save the file.

00	MyDupli 🛛	cate.applescript 💿
Record Stop Run	Compile	
AppleScript ‡ on ma	uin ‡	
on main(inputs, outpu set returnValue	tFolder, params) to "OK"	
Translate the or set outputFolderR	utput folder (UNIX p eference <b>to</b> <i>POSIX</i> a	ath) into an AppleScript file reference file outputFolder
Get the maxim megabyte to set maxSize to	mum size from the bytes (item 1 of params	e script parameters and convert from
Iterate through <b>repeat with</b> input Translate th <b>set</b> inputRefe	the list of inputs in inputs he input (UNIX path rence to POSIX file	) into an AppleScript file reference input
Get the s set inputSiz	ize of the input e to size of (in	fo for inputReference)
Check wet if inputSize Duplica	ther the input is < maxSize then te input to output for	OK to duplicate
tell applic	ation "Finder" to di outputFolderRefe	uplicate inputReference to erence with replacing
log inpu set retu	t & " is too big rnValue to "Warn:	to duplicate" ing"
end if end repeat		
<b>return</b> returnVal	ue	
end main		
	Description	Event Log

4. Open the Automation Engine Pilot. Go to Files view where you can select the files to be copied and open a New Task. Choose the Run Script task, modify its settings and launch. This modified ticket will duplicate every selected file which is smaller than 10MB to the current job's Script Output folder. In this example, we excuted this task for two files(Blairon.pdf: 22MB and Dansk Droge.pdf: <1MB).</p>





'Dansk Droge.pdf' is duplicated into the job's Script Output folder. 'Blairon.pdf' was too big to duplicate (> 10MB). Therefore, the task ended in 'Warning' state and added an entry in the task details.

000		Files (RunScript	Demo)					
무 않	RunScriptDemo	+				Q		
Mode	Job	New Job					Search	
VIEWS Files To Do List Pages Products Proofs	RunScriptDemo     Artwork     Gsko     ScriptOutput	Name	View <u>View</u>	Type PDF File	Size D 89KB 2	bate 2/10/10 2:16 PM	No Favori	tes
WebCenter WebCenter CDI Tasks Milestones Tickets TOOLS TOOLS Tig Devices Tig Queue Monitor	File Name Blairon.pdf,Dansk Droge.pdf Task Type: Run Script Started on Server: begezingem Started at 5/31/12 12:36 Finished at 5/31/12 12:36 Process Time: 0:00:00 The script's standard output st Volumes/MydbcOntainer/Run5	Task Type Prog Run Script 100 PM s PM eam: criptDemo/Artwork (Blairon.c	ress Phase % Details	duplicate	State	Launched 5/31/12 12:36	Task ID PM 3781	>.
Containers	New Task New Workflow Info	α φύσει το για του κράται οι τ <sub>ρ</sub>		uupicate				
	1 file				Serv	er: begezingem	User: admin	0 🛤

## 5.2 Using Shell Script on Script Runner

In this example, we use a Shell Script to copy every input file with a size smaller than the specified size in the script parameters to the output folder.

1. Open a text editor and add following code. When the Script Runner executes this code, \$1 (the script's first argument) will contain a string of input file paths separated by :. The code splits up the concatenated file paths into a real list. This helps to iterate through the list and handle the **Run Script** task's inputs one by one.



2. Write and save the code as below. This script copies the input to the output folder if the input's file size is smaller than the maximum size from the script parameters. If the size of the file is bigger, it adds an entry in the log and makes sure the task ends in 'Warning' state (via exit value '1'). Save this code as a text file to the Script Runner's Shell folder (default: /Library/Scripts/Esko/Shell) or to the Automation Engine Shell folder.



\$1	First shell script argument: the Run Script task's inputs. A string of input file paths, separated by ':'.
\$2	Second shell script argument or output folder: This is the folder where Automation Engine expects the script's result files. AE will continue the flow with the files you write in this folder. If you leave this folder empty, AE will continue the flow with the inputs of the Run Script task.
\$3, \$4, \$5,	Remaining shell script arguments: additional script parameters which you can inject into the script via the Run Script ticket.
exitValue	Ending Status of the task
0	ОК
1	Warning

3. Open the Automation Engine Pilot. Go to Files view where you can select a file and open a New Task. Choose the Run Script task, modify its settings and launch. This modified ticket will duplicate every selected file which is smaller than 10MB to the current job's Script Output folder. In this example, we excuted this task for two files(Blairon.pdf: 22MB and Dansk Droge.pdf: <1MB).</p>

Error

0 0		New T	ask – Defa	ult		
Launch	n 📼	Priority: Normal	<b>;</b>	🗌 Hold	Ticket: 👼 🕶	Tickets: 🛅 🕇
Settings for S	tep: Run S	cript				
>_ 0	utput in:	[Job URL]/ScriptOutpu	ıt			BROWSE []
A FI	ile Name:					BROWSE []
Run o Script Typ Script Fil	n: My M e: Shell e: MyCop	lac Script Runner				[]
Script Para 10 + –	umeters					

'Dansk Droge.pdf' is duplicated into the job's Script Output folder. 'Blairon.pdf' was too big to duplicate (> 10MB). Therefore, the task ended in 'Warning' state (due to exitValue=1 in the code) and added an entry in the task details.

000		Files (R	unScriptDemo	)					
무 않	RunScriptDemo	÷ • •					(Q,		
Mode	Job	New Job						Search	
VIEWS	Comparison of the second	Name	.pdf	View View	Type PDF File	Size 89KB	Date 2/10/10 2:16 PM	No Favori	ites
WebCenter	File Name	Task Type	Progress	Phase	Kuriser	Stat	e Launched	Task ID	
Plates	Blairon.pdf.Dansk Droge.pdf	Run Script	100%	Filade		Jul	5/31/12 12:3	6 PM 3781	
CDI	00			Details					
TICKETS	Task Type: Run Script Started on Server: begezingem Started at: 5/31/12 12:3 Finished at: 5/31/12 12:3 Process Time: 0:00:00	5 PM 5 PM							<u>&gt;</u> .
Queue Monitor Containers Customers	The script's standard output st /Volumes/MyJobContainer/Run The script's error output stream <none></none>	ream: ScriptDemo/Artwork, 1:	Blairon.pdf is 1	oo big ti	duplicate				
🔛 Hot Folders	New Task New Workflow	)							

# 6. Scripting for Script Runner on Windows

A Script Runner on Windows supports Windows Script and Batch File. We recommend Windows Script for its scripting abilities comparable to batch files, its wider range of supported features and the simpler syntax. Windows Script is plain-text VBScript or JScript which is interpreted and run by the Windows Script Host.

## 6.1 Using Windows Script on Script Runner

1. Open a text editor and add the following code:



Function Main	The function that will be called by the Script Runner. Script Runner executes only the code in this main function.
inputs	First argument of main function: a list of input file paths(type: list of strings).
outputFolder	Second argument of the main function: the folder where AE expects the script's result files. AE will continue the flow with the files you write in this folder. If you leave this folder empty, AE will continue the flow with the inputs of the Run Script task (type: string).
params	Third argument of the main function: additional script parameters injected into the script via the Run Script ticket (type: list of strings).
WScript.Echo	This puts some extra log info in the Run Script task details and log. This call prints text to the Console and adds a newline character without Script Runner context.

Main = "OK"	This will communicate with the Run Script
	task that everything went fine. Other
	possibilities are Main = "Warning" and
	Main = "Error".

2. You can test this script locally by adding the following code: Save this file. Open command prompt. Change the directory to the script's parent directory. Run command 'cscript MyHelloWorld.vbs'.



This will produce the output 'Hello, World!' to the console. The Script Runner does not attend to the test code in your script. It will execute the contents of the main function and ignore the rest. Therefore, you can keep your test code for future local testing.

**3.** Open the Automation Engine Pilot. Go to **Files** view where you can select a file and open a **New Task**. Choose the **Run Script** task, modify its settings and launch the task.

# 6 ESKD 😌

00	New Task – Default	
E Launch	Priority: Normal 🗘 🗌 Hold Ticket: 🗃 -	Tickets: 🚟 🕶
Settings for Step	p: Run Script	
>_ Out	put in:	BROWSE []
🔔 File	Name:	BROWSE []
Run on:	My Win Script Runner	
Script Type:	Windows Script	
Script File:	MyHelloWorld.vbs	CI 💽
Script Parame	eters	
+ -		

Note that the 'Hello, World!' in the task details and 'OK' state are corresponding with WScript.Echo "Hello, World!" and Main = "OK" in the script.



### 6.1.1 Using Windows Script On Script Runner: A Sample Case

In this example, we use Windows Script to copy every input file with a size smaller than the size specified in the script parameters to the output folder.

We can use inputs, outputFolder and params in the AppleScript to achieve our objective. First, we demonstrate how to duplicate files without the size restriction and then proceed with the actual use case.

1. Open a text editor and add the code given below. This code is aimed to iterate through the list of inputs. It enables you to handle the inputs one by one, via the 'input' variable.



2. You can modify the Script as given below to duplicate the files to a specified output folder without size restrictions. Save this code as a text file with '.vbs' extension (VBScript) in the Windows



Script folder of Script Runner (default: C:\Esko\bg\_data\_fastserverscrrunnt\_v100\Scripts \WindowsScript) or in the Automation Engine WindowsScript folder.



**3.** Add the file size check in the code as given below. This will duplicate the file when the input file size is smaller than the maximum size from the script parameters. If this condition is not met it will add an entry in the log and there will be "Warning". Save the file.

```
*C:\Esko\bg_data_fastserverscrrunnt_v100\Scripts\WindowsScript\MyCopy.vbs - Notepad++
<u>File Edit Search View Encoding Language Settings Macro Run Plugins Window ?</u>
                                                                                   Х
 | 2 🚽 🗄 🕒 12 👘 10 | 2 🕊 | # 🎍 | 3 3 4 | 12 🔄 1 🗐 🖉 | ● 🗉 🕨 >
MyCopy.vbs
      - Function Main(inputs, outputFolder, params)
   3
            Dim returnValue
            returnValue = "OK"
   4
            ' Get the maximum size from the script parameters
   6
            ' and convert from megabyte to bytes.
   7
   8
            Dim maxSize
   9
            maxSize = params(0) * 1000000
            ' Create the file system object that is going to do the copying.
            Dim filesys
  13
            Set filesys = CreateObject("Scripting.FileSystemObject")
 14
            WScript.Echo "File system object created"
            ' Iterate through the list of inputs.
 16
 17
            For Each input In inputs
 18
                ' Turn the input path into a file object.
 19
                Dim inputReference
 20
               Set inputReference = filesys.GetFile(input)
 21
 22
                ' Check wether the input is OK to copy.
 23
                If inputReference.Size < maxSize Then</pre>
                    ' Copy input to output folder.
 24
 25
                    Dim outputFile
 26
                    outputFile = outputFolder & "\" & filesys.GetFileName(input)
 27
                    filesys.CopyFile input, outputFile
 28
                Else
                   WScript.Echo input & " is too big to copy"
 29
 30
                    returnValue = "Warning"
 31
                End If
 32
            Next
 33
            ' Set the function's return value.
 34
 35
            Main = returnValue
 36
       End Function
 37
 •
                                Ш
                                                                               INS
length: 1046 lines: 37
                   Ln:31 Col:15 Sel:0
                                                 Dos\Windows
                                                               ANSI
```

4. Open the Automation Engine Pilot. Go to Files view where you can select the files to be copied and open a New Task. Choose the Run Script task, modify its settings and launch. This modified ticket will duplicate every selected file which is smaller than 10MB to the current job's Script Output folder. In this example, we excuted this task for two files(Blairon.pdf: 22MB and Dansk Droge.pdf: <1MB).</p>

# 6 ESKD 😌

0 0	New Task – Default	
E Launch	Priority: Normal 🗘 🗌 Hold Ticket: 📷 🗸	Tickets: 🗮 🕶
Settings for Step	: Run Script	
>_ Out	out in: [Job URL]/ScriptOutput	BROWSE []
A File	Name:	BROWSE []
Run on: Script Type:	My Win Script Runner	
Script File:	MyCopy.vbs	[] 🔽
Script Parame	ters	
+ -		

'Dansk Droge.pdf' is duplicated into the job's Script Output folder. 'Blairon.pdf' was too big to duplicate (> 10MB). Therefore, the task ended in 'Warning' state and added an entry in the task details.

00		Files (RunScriptDe	mo)		
日 28 「	RunScriptDemo	; +		Q	
Mode	Job	Nev	Job		Search
VIEWS Files To Do List Pages	RunScriptDemo     Artwork     Esko     ScriptOutput	Name	View T View P	Size Date 89KB 2/10/	No Favorites
Products	File Name Task	Type Progress	Phase	State Launche	ed Task ID
C WebCenter		cript 100%	Details	6/6/12	4:2 3800
CDI Tasks	Task Type: Run S Started on Server: begez Started at: 6/6/1 Finished at: 6/6/1 Process Time: 0:00:0	cript ingem 2 4:27 PM 2 4:27 PM 10			>_
▼ TOOLS	The script's standard ou Microsoft (R) Windows S Copyright (C) Microsoft ( File system object creat \\begezingem\MyJobCc The script's error outpu <none></none>	utput stream: cript Host Version 5.8 Corporation. All rights n ed ontainer\RunScriptDemo t stream:	eserved. \Artwork\Blairo	n.pdf is too big to	сору

## 6.2 Using Batch File for Scripting on Windows

In this example, we use Batch File to copy every input file with a size smaller than the size specified in the script parameters to the output folder.

1. Open a text editor and add the code given below. This code is aimed to iterate through the list of inputs. It enables you to handle the inputs one by one, via the command %1 (the script's first argument) will contain a string of input file paths, separated by ';'.



2. You can modify the Script as given below to duplicate the files to a specified output folder with a size check. Save this code as a text file with '. bat ' extension in the Script Runner Batch File Folder (default: C:\Esko\bg\_data\_fastserverscrrunnt\_v100\Scripts\BatchFile) or in the Automation Engine BatchFile folder.

ESKO 🕄

```
- • ×
C:\Esko\bg_data_fastserverscrrunnt_v100\Scripts\Batch\myCopy.bat - Notepad++
<u>File Edit Search View Encoding Language Settings Macro Run Plugins Window ?</u>
                                                                                    Х
 - 🕞 🖶 🕒 🕞 🕞 😓 | 🗶 🐚 🜔 | Ə 🙋 📾 🍢 | 🔍 👒 | 🖳 🔂 = 1 🗐 🗐 | ● 🗉 🕨 🚱 🔂
📄 myCopy.bat
  1
     echo off
  2 rem First script argument contains the input file paths
  3 set inputs=81
  4 rem Trim surrounding quotes
  5
     for /f "usebackq tokens=*" <mark>%%a</mark> in ('<mark>%inputs%</mark>') do set inputs=<mark>%%~a</mark>
  6
  7 rem Get the output folder
  8 set outputFolder=82
  9
 10 rem Get the maximum size from the script parameters
 11 rem and convert from megabyte to bytes
 12 set /a maxSize=83*1000000
 13
 14
     set exitValue=0
 15
 16 rem Iterate through the input file paths which are separated by ';'
 17 : ITERATE_INPUTS
 18 if "%inputs%"=="" goto EXIT
 19 for /f "tokens=1 delims=;" %%a in ("%inputs%") do call :HANDLE_INPUT "%%a"
 20 for /f "tokens=1* delims=;" %%a in ("%inputs%") do set inputs=%%b
 21 goto ITERATE_INPUTS
 22
 23
     HANDLE INPUT
 24 rem Get the size of the input
 25 for %%? in (%1) do set inputSize=%%~z?
 26 rem Check wether the input is OK to copy
 27 if %inputSize% lss %maxSize% (
         rem Copy input to output folder
 28
 29
         copy /y %1 %outputFolder%
 30 ) else (
        echo <mark>%1</mark> is too big to copy
 31
         set exitValue=1
 32
 33
 34 goto :eof
 35
 36 :EXIT
     exit %exitValue%
 37
 38
                                                                                INS
Bate length : 973 lines : 38
                          Ln:38 Col:1 Sel:0
                                                     Dos\Windows
                                                                  ANSI
```

81	First batch file argument: the Run Script task's inputs. A string of input file paths, separated by ';'.
%2	Second batch file argument or output folder: the folder where AE expects the script's result files. AE will continue the flow with the files you write in this folder. If you leave this folder empty, AE will continue the flow with the inputs of the Run Script task.
%3, %4,%5	Remaining batch file arguments: additional script parameters, injected into the script via the <b>Run Script</b> ticket.

Exit value	Ending Status of the task
0	OK
1	Warning
2	Error

3. Open the Automation Engine Pilot. Go to Files view where you can select the files to be copied and open a New Task. Choose the Run Script task, modify its settings and launch. This modified ticket will duplicate every selected file which is smaller than 10MB to the current job's Script Output folder. In this example, we excuted this task for two files(Blairon.pdf: 22MB and Dansk Droge.pdf: <1MB).</p>

0 0	New Task – Default	
Launch	Priority: Normal 🗘 🗌 Hold Ticket: 📅	• Tickets: 🚟 •
Settings for Step	2: Run Script	
Dut:	put in: [Job URL]/ScriptOutput	BROWSE []
File	Name:	BROWSE []
Run on: Script Type:	My Win Script Runner	
Script File:	myCopy.bat	E1 💌
Script Parame	eters	

'Dansk Droge.pdf' is duplicated into the job's Script Output folder. 'Blairon.pdf' was too big to duplicate (> 10MB). Therefore, the task ended in 'Warning' state and added an entry in the task details.





## 7. Using ExtendScript on Script Runner(On MacOS/Windows)

A Script Runner supports direct interpretation of ExtendScript by the Adobe Creative Suite (CS) application selected in the **Run Script** ticket.

ExtendScript is JavaScript extended for Adobe CS applications. Adobe provides a complete integrated development environment (IDE) for programming ExtendScript which is the ExtendScript Toolkit (ESTK). Latest versions of the ESTK are available with the Creative Suite. For more info and Adobe scripting resources visit the Adobe Scripting Center.

 Open the ExtendScript Toolkit and add following code. Save this code in the Script Runner's ExtendScript folder. The default location is : /Library/Scripts/Esko/ExtendScript for a Script Runner on Mac OS or C:\Esko\bg\_data\_fastserverscrrunnt\_v100\Scripts\ExtendScript on Windows. Alternatively, you can save them in the ExtendScript folder of Automation Engine.



params	Third argument of main function: additional script parameters, injected into the script via the <b>Run Script</b> ticket. (type: list of strings)
\$.writeln	This writes extra log information in the Run Script task details and log. Without Script Runner context this call prints text to the Console, and adds a newline character.
alert	This registers some extra log info in the <b>Run</b> <b>Script</b> task details and log. Without Script Runner context this call displays an alert box.
return "OK";	This will communicate with the <b>Run Script</b> task that everything went fine. Other possibilities are Return = "Warning" and Return = "Error".

2. To test the script locally in the ExtendScript Toolkit, add following code, save and run.



Note that the alert box pops up and the 'Hello, World!' and 'OK' result in the Console.





The Script Runner does not attend to the test code in your script. It will execute only the contents of the main function and ignore the rest. Therefore, you can keep your test code for future local testing.

**3.** Open the Automation Engine Pilot. Go to **Files** view where you can select a file and open a **New Task**. Choose the **Run Script** task, modify its settings and launch the task.

00	New Task - Default	
E Launch	Priority: Normal 🗘 🗌 Hold Ticket: 🐼 - Tickets: 🚆	•
Settings for Step	Run Script	
>_ Outp	BROWSE []	
File	Name:	
Run on: Script Type: Application: Script File:	My Mac Script Runner	
Script Parame	eters	

Note that the 'Hello, World!' in the task details and 'OK' state are corresponding with \$.writeln("Hello World!") and Return = "OK" in the script.

00	Tasks (RunScriptDemo)						
무 않 🔽	RunScriptDemo	<b>‡</b> +		Q			
Mode	Job	New	New Job		Search		
▼ VIEWS	Job Name File N	ame Task Type	Progress	Phase State	Launched 🔻	Task ID	
🗋 Files	RunScriptDemo Blair	n.pdf Run Script	100%		6/4/12 2:11 PM	3788	
🔁 To Do List		-	0				
Pages	Task Type: Run Started on Server: bege	cript ingem				>_	
Products	Started at: 6/4/	2 2:11 PM					
Proofs	Finished at: 6/4/	2 2:11 PM					
O WebCenter	Process Time: 0:00:	)1					
Plates	The script's standard output stream:						
迷 CDI	Another way to say: He	lo, World!					
📙 Tasks	The script's error output	t stream:					
📻 Milestones	<none></none>						
🔚 Tickets							
TOOLS							
Devices	Relaunch Move to His	ory Delete					
	1 of 1 selected		Se	rver: begezinge	m User: admin	<b>V</b> 0	

## 7.1 Using ExtendScript on Script Runner: A Sample Case.

In the following example we are going to use ExtendScript to **Open** a file in Illustrator and print it using a **Print Preset**.

**Note:** When you use ExtendScript on Windows, you can avoid troubles while accessing your user specific settings such as Adobe applications' **Presets**, **Actions** etc by stopping the Script Runner service and running it as an application for the logged in user (who also defined Adobe settings).

Open Start > All Programs > Esko > Automation Engine Script Runner > Preferences .

Stop the Script Runner and uncheck Start at login (which actually means 'Start as service') and Close Preferences.

Start as a console application by double clicking its executable: Script Runner's program folder> \bin\_ix86\egscrrun.exe (e.g.C:\Esko\bg\_prog\_fastserverscrrunnt\_v120\bin\_ix86\egscrrun.exe)

**Note:** To prevent having to start up the Script Runner every time you log in, add its executable to your user's/system's Startup Items.

- 1. Define a My Print Preset in your Adobe Illustrator application.
- 2. Open the ExtendScript Toolkit and add following code. This code is aimed to iterate through the list of inputs. It enables you to handle the inputs one by one, via the input variable.



**3.** To print every input file using a **Print Preset** from the script parameters of the **Run Script** task, write the code as below. Save this code in the default ExtendScript folder(of Script Runner or Automation Engine).





 Open the Automation Engine Pilot. Go to Files view where you can select a file and open a New Task. Choose the Run Script task, modify its settings and launch the task.

0 0	New Task – Default				
Eaunch	Priority: Normal 🗘 🗌 Hold Ticket: 🗟 - Tickets: 🗮 -				
Settings for Step: Run Script					
Dut:	but in: BROWSE []				
File	Name: BROWSE []				
Run on: Script Type: Application:	My Mac Script Runner				
Script File: Script Parame	MyPrint.jsx []				
My Print Pres	et				

Launching this ticket will result in the selected Illustrator files being printed using the Print Preset specified as script parameters ( $M_Y$  Print Preset) in the task.

## 8. Appendix : Script Samples

#### Note:

Sample scripts are provided as-is with no warranty of fitness for a particular purpose. These scripts are solely intended to demonstrate techniques for accomplishing common tasks. Additional script logic and error-handling may need to be added to achieve the desired results in your specific environment.

It is up to the user to verify that his intended use of the offered automation functionality is compliant with any third party license agreement and/or other restrictions applicable to any non-Esko products.

### 8.1 AppleScript Code Samples

#### MyHelloWorld.applescript

```
on main(inputs, outputFolder, params)
log "Hello, World!"
return "OK"
end main
on run
-- Our main function is not using any of its arguments, so provide empty ones
set inputs to {} -- empty list
set outputFolder to "" -- empty string
set params to {} -- empty list
main(inputs, outputFolder, params)
end run
```

MyDuplicate.applescript

```
on main(inputs, outputFolder, params)
set returnValue to "OK"
-- Translate the output folder (UNIX path) into an AppleScript file
reference
set outputFolderReference to POSIX file outputFolder
-- Get the maximum size from the script parameters and convert from megabyte
to bytes
set maxSize to (item 1 of params) * 1000000
-- Iterate through the list of inputs
repeat with input in inputs
-- Translate the input (UNIX path) into an AppleScript file reference
set inputReference to POSIX file input
-- Get the size of the input
set inputSize to size of (info for inputReference)
-- Check wether the input is OK to duplicate
```



if inputSize < maxSize then -- Duplicate input to output folder tell application "Finder" to duplicate inputReference to outputFolderReference with replacing else log input & " is too big to duplicate" set returnValue to "Warning" end if end repeat return returnValue end main

## 8.2 Shell Script Code Sample

#### MyCopy.sh

#### #!/bin/bash

```
# Split up the input file paths into a list
# via the Internal Field Separator (IFS)
OLDIFS=$IFS # Always keep the original IFS
IFS=":" # Now set it to ':'
# Split up '$1' into the 'inputs' variable, with ':' as separator
inputs=( $1 )
IFS=$OLDIFS # Restore the original IFS
inputCount=${#inputs[@]}
# Get the output folder
outputFolder=$2
# Get the maximum size from the script parameters
# and convert from megabyte to bytes
maxSize=`expr $3 \* 1000000`
exitValue=0
# Iterate through the list of inputs
for (( i=0; i<${inputCount}; i++ ));</pre>
do
input=${inputs[$i]}
 # Get the size of the input
 inputSize=`ls -l "$input" | awk '{print $5}'`
 # Check wether the input is OK to copy
 if [ $inputSize -lt $maxSize ]; then
 # Copy input to output folder
 cp -f "$input" "$outputFolder"
 else
 echo "$input is too big to duplicate"
 exitValue=1
 fi
done
exit $exitValue
```

## 8.3 Windows Script Code Samples

#### MyHelloWorld.vbs

```
Function Main(inputs, outputFolder, params)
WScript.Echo "Hello, World!"
' Set the function's return value.
Main = "OK"
End Function
' Our main function is not using any of its arguments,
' so provide empty ones.
Dim inputs()
Dim outputFolder
Dim params()
Main inputs, outputFolder, params
```

#### MyCopy.vbs

```
Function Main(inputs, outputFolder, params)
Dim returnValue
returnValue = "OK"
' Get the maximum size from the script parameters
' and convert from megabyte to bytes.
Dim maxSize
maxSize = params(0) * 1000000
' Create the file system object that is going to do the copying.
Dim filesys
Set filesys = CreateObject("Scripting.FileSystemObject")
WScript.Echo "File system object created"
 ' Iterate through the list of inputs.
For Each input In inputs
  ' Turn the input path into a file object.
 Dim inputReference
 Set inputReference = filesys.GetFile(input)
  ' Check wether the input is OK to copy.
 If inputReference.Size < maxSize Then</pre>
  ' Copy input to output folder.
  Dim outputFile
  outputFile = outputFolder & "\" & filesys.GetFileName(input)
  filesys.CopyFile input, outputFile
 Else
  WScript.Echo input & " is too big to copy"
  returnValue = "Warning"
 End If
Next
' Set the function's return value.
Main = returnValue
End Function
```



## 8.4 Batch File Code Sample

### MyCopy.bat

```
@echo off
rem First script argument contains the input file paths
set inputs=%1
rem Trim surrounding quotes
for /f "usebackq tokens=*" %%a in ('%inputs%') do set inputs=%%~a
rem Get the output folder
set outputFolder=%2
rem Get the maximum size from the script parameters
rem and convert from megabyte to bytes
set /a maxSize=%3*1000000
set exitValue=0
rem Iterate through the input file paths which are separated by ';'
:ITERATE_INPUTS
if "%inputs%"=="" goto EXIT
for /f "tokens=1 delims=;" %%a in ("%inputs%") do call :HANDLE_INPUT "%%a"
for /f "tokens=1* delims=;" %%a in ("%inputs%") do set inputs=%%b
goto ITERATE_INPUTS
:HANDLE_INPUT
rem Get the size of the input
for %%? in (%1) do set inputSize=%%~z?
rem Check wether the input is OK to copy
if %inputSize% lss %maxSize% (
rem Copy input to output folder
copy /y %1 %outputFolder%
) else (
echo %1 is too big to copy
set exitValue=1
goto :eof
: EXIT
exit %exitValue%
```

### 8.5 ExtendScript Code Samples

#### MyHelloWorld.jsx

```
function main (inputs, outputFolder, params) {
    $.writeln ("Hello, World!");
    alert ("Another way to say: Hello, World!");
    return "OK";
}
// Our main function is not using any of its arguments, so you can test
main without arguments
main ();
```

#### MyPrint.jsx

```
function main(inputs, outputFolder, params) {
    // First and only script parameter is the print preset.
    var printPreset = params[0];
    var returnValue = "OK";
    // Iterate through the list of inputs.
    for (i=0; i<inputs.length; i++)</pre>
    {
        var input = inputs[i];
        try
        {
            // Open the input file.
          var myDocument = app.open(File(input), DocumentColorSpace.CMYK);
            // Create print options and set the print preset.
            var options = new PrintOptions();
            options.printPreset = printPreset;
            // Print the document.
            myDocument.print(options);
            // Close the document without saving.
            myDocument.close(SaveOptions.DONOTSAVECHANGES);
        }
        catch (e)
        ł
            // Log problem info and set the return value to "Warning".
            $.writeln("Problem while printing " + input);
            $.writeln("Name: " + e.name);
$.writeln("Message: " + e.message);
            returnValue = "Warning";
        }
    }
    return returnValue;
```