How to extend your EFT Server automation using PowerShell

KB Article Draft

Applies to: All EFT Versions

PowerShell it's a very powerful scripting language that allows you automate and perform many task, it does have great advantages over using VBScript it also can be reusable and easy to maintain.

There are many editors that can provide Intellisense and debugging capabilities in addition of a full community support.
http://www.serverwatch.com/server-tutorials/6-powerful-powershell-tools-and-utilities.html

This article doesn't describe how to use PowerShell however it describes how you can use it to extend EFT Automation capabilities.

Here is an example of a PowerShell script that takes one argument and displays it on the output.

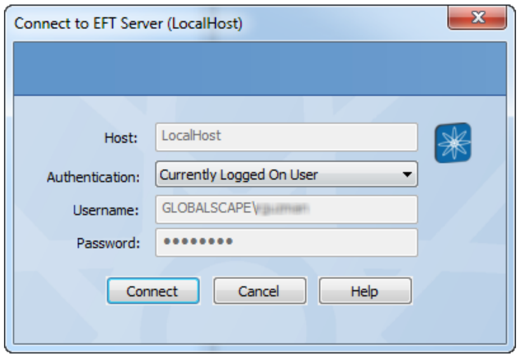
C:\temp\Hello.ps1

```
[CmdletBinding()]
Param(
        [String]$Name=""
)
Write-Host "Hello $Name from Powershell..."
```

You can save this in EFT.

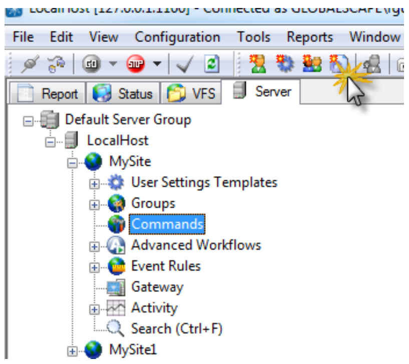
Now let's integrate this script with an EFT Event Rule.

1. Open the EFT administration interface.

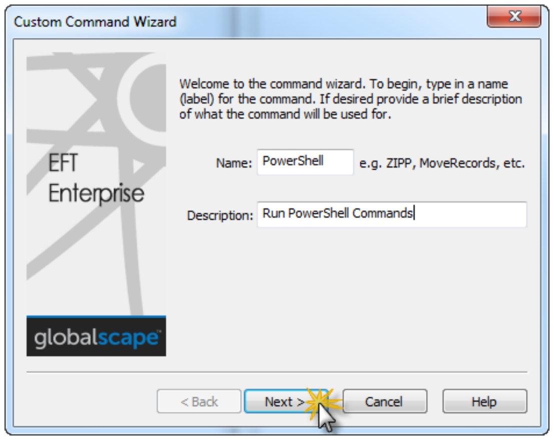
2. Login with your admin credentials.
   
   Connect to EFT Server (LocalHost)

   Host: LocalHost
   Authentication: Currently Logged On User
   Username: GLOBALSCAPE\
   Password: ••••••••

3. Create a new Custom Command.



4. A new Custom Command wizard will appear, enter the **Name** and the **Description** of the Custom Command and click **Next**.
   Name: PowerShell
   Description: Run PowerShell Commands

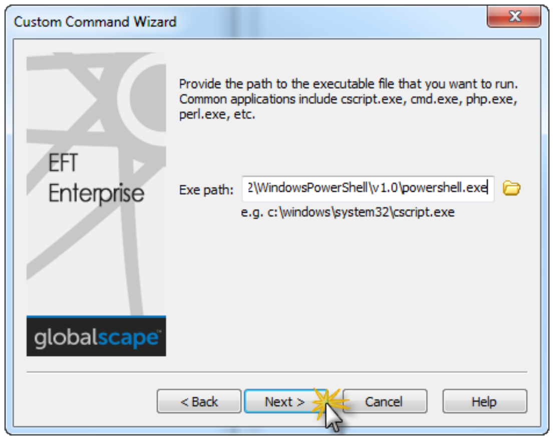


Welcome to the command wizard. To begin, type in a name (label) for the command. If desired provide a brief description of what the command will be used for.

Name: PowerShell e.g. ZIPP, MoveRecords, etc.

Description: Run PowerShell Commands

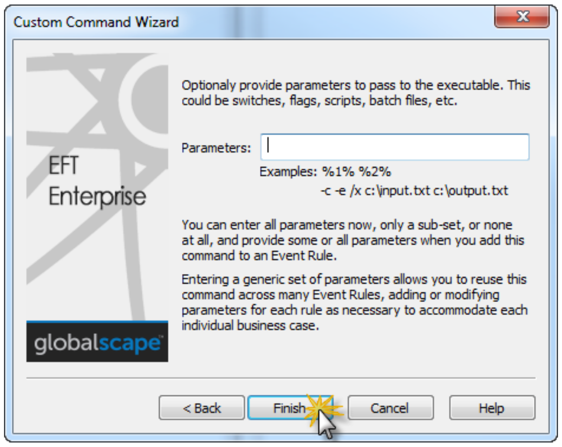
5. Enter the **Executable path** for powershell.exe and click **Next**.
   Exe path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe



Provide the path to the executable file that you want to run. Common applications include cscript.exe, cmd.exe, php.exe, perl.exe, etc.

Exe path: 2\WindowsPowerShell\v1.0\powershell.exe
e.g. c:\windows\system32\cscript.exe

6. Leave parameters empty and click **Finish**.



Custom Command Wizard

Optionaly provide parameters to pass to the executable. This could be switches, flags, scripts, batch files, etc.

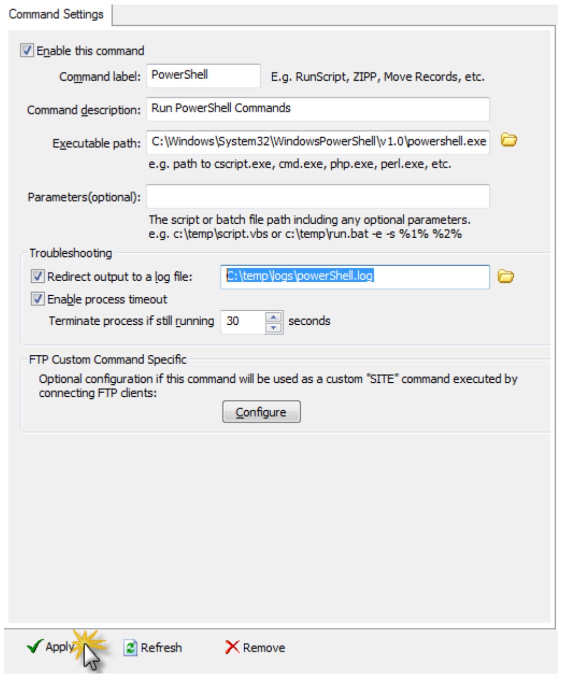
Parameters:

Examples: %1% %2%
-c -e /x c:\input.txt c:\output.txt

You can enter all parameters now, only a sub-set, or none at all, and provide some or all parameters when you add this command to an Event Rule.

Entering a generic set of parameters allows you to reuse this command across many Event Rules, adding or modifying parameters for each rule as necessary to accommodate each individual business case.

7. Configure **Troubleshooting Custom command** settings and click **Apply**.
Redirect output to a log File: C:\temp\logs\powerShell.log (must have this directory already created, or use different path for an existing folder)
Enable Process timeout: 30 secs



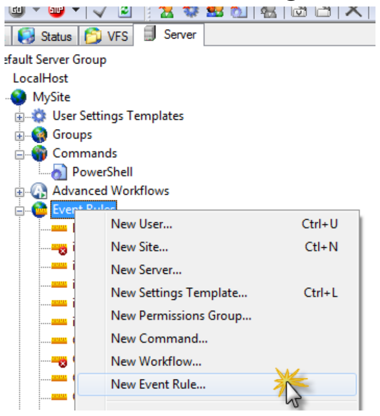
Command Settings

☑ Enable this command
Command label: PowerShell — E.g. RunScript, ZIPP, Move Records, etc.
Command description: Run PowerShell Commands
Executable path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
e.g. path to cscript.exe, cmd.exe, php.exe, perl.exe, etc.
Parameters(optional):
The script or batch file path including any optional parameters.
e.g. c:\temp\script.vbs or c:\temp\run.bat -e -s %1% %2%

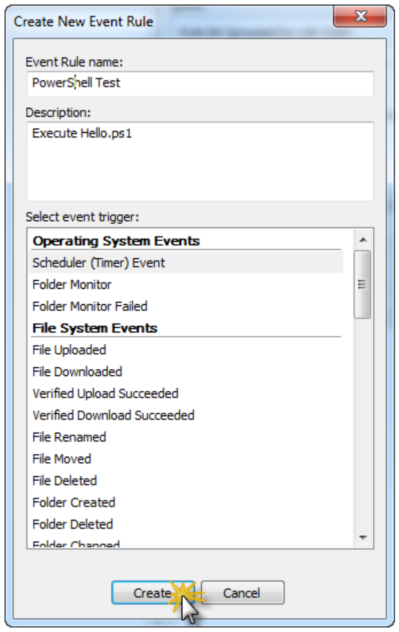
Troubleshooting
☑ Redirect output to a log file: C:\temp\logs\powerShell.log
☑ Enable process timeout
Terminate process if still running 30 seconds

FTP Custom Command Specific
Optional configuration if this command will be used as a custom "SITE" command executed by connecting FTP clients:

8. Create an Event Rule, right click on Event Rules and then click **New Event Rule**.



9. Enter the name of the event Rule and then provide the Event Rule Name: PowerShell Test and Description: Execute Hello.ps1

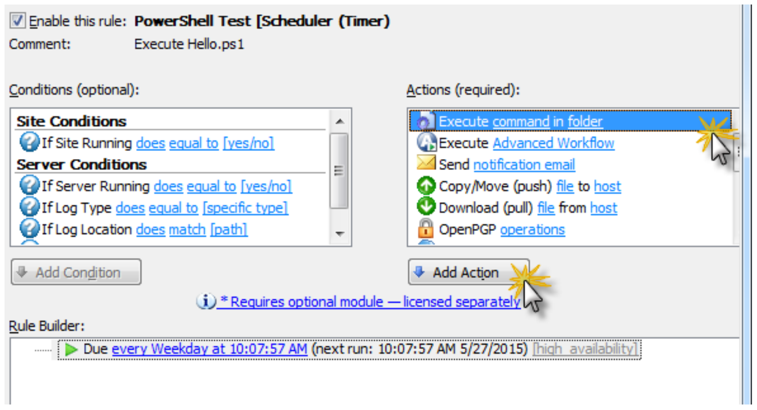


Event Rule name:
PowerShell Test

Description:
Execute Hello.ps1

Select event trigger:
Operating System Events
Scheduler (Timer) Event
Folder Monitor
Folder Monitor Failed
File System Events
File Uploaded
File Downloaded
Verified Upload Succeeded
Verified Download Succeeded
File Renamed
File Moved
File Deleted
Folder Created
Folder Deleted
Folder Changed

10. Select the **Execute Command in folder** action and then click **Add Action**.



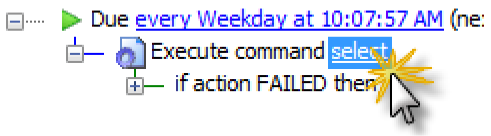
Enable this rule: **PowerShell Test [Scheduler (Timer)**
Comment: Execute Hello.ps1

Conditions (optional):
Site Conditions
If Site Running does equal to [yes/no]
Server Conditions
If Server Running does equal to [yes/no]
If Log Type does equal to [specific type]
If Log Location does match [path]

Actions (required):
Execute command in folder
Execute Advanced Workflow
Send notification email
Copy/Move (push) file to host
Download (pull) file from host
OpenPGP operations

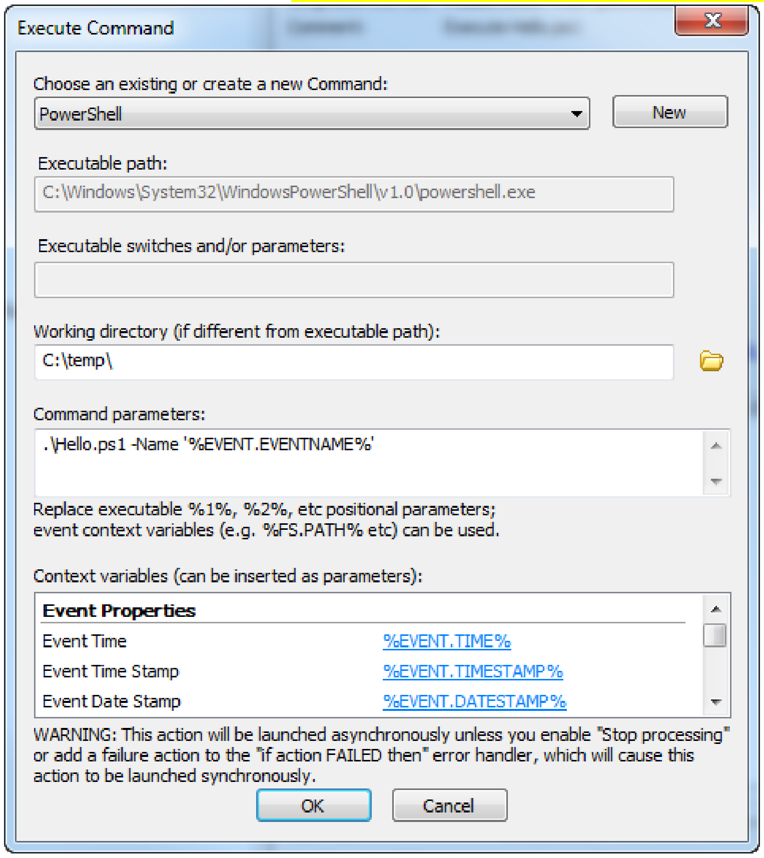
Rule Builder:
Due every Weekday at 10:07:57 AM (next run: 10:07:57 AM 5/27/2015) [high availability]

11. Click **Select**.



Due every Weekday at 10:07:57 AM (ne:
Execute command select
if action FAILED then

12. Choose "PowerShell" Custom Command and add a context variable name parameter.
Custom Command: PowerShell
Working Directory: C:\temp\
Command Parameters: .\Hello.ps1 -Name '%EVENT.EVENTNAME%'



Execute Command

Choose an existing or create a new Command: PowerShell

Executable path: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Executable switches and/or parameters:

Working directory (if different from executable path): C:\temp\

Command parameters:

```
.\Hello.ps1 -Name '%EVENT.EVENTNAME%'
```

Replace executable %1%, %2%, etc positional parameters;
event context variables (e.g. %FS.PATH% etc) can be used.

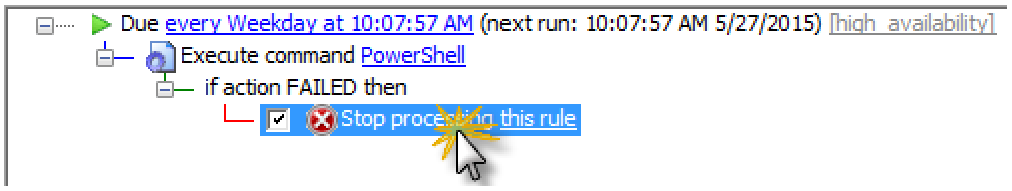
Context variables (can be inserted as parameters):

Event Properties	
Event Time	%EVENT.TIME%
Event Time Stamp	%EVENT.TIMESTAMP%
Event Date Stamp	%EVENT.DATESTAMP%

WARNING: This action will be launched asynchronously unless you enable "Stop processing" or add a failure action to the "if action FAILED then" error handler, which will cause this action to be launched synchronously.

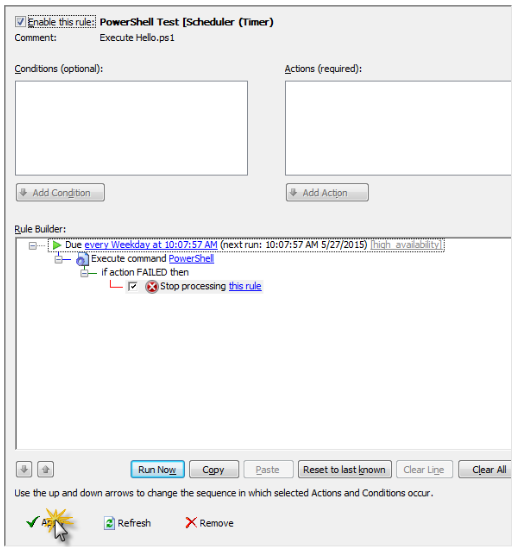
NOTE: *It's important to notice that there are **single quotes** between the %EVENT.EVENTNAME% parameter; this is needed in case the Event Rule name has spaces so the script can deal with them correctly.*

13. Select the **Stop processing this rule** check box. This forces EFT to wait until the process is complete to return and process any subsequent action within the Event Rule, or leave cleared if you want to run this process execution asynchronously.



Due every Weekday at 10:07:57 AM (next run: 10:07:57 AM 5/27/2015) [high availability]
Execute command PowerShell
if action FAILED then
[✓] Stop processing this rule

14. Click **Apply**.



Enable this rule: **PowerShell Test [Scheduler (Timer)**

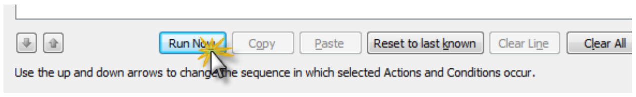
Comment: Execute Hello.ps1

Rule Builder:

```
Due every Weekday at 10:07:57 AM (next run: 10:07:57 AM 5/27/2015) [high availability]
    Execute command PowerShell
        if action FAILED then
            Stop processing this rule
```

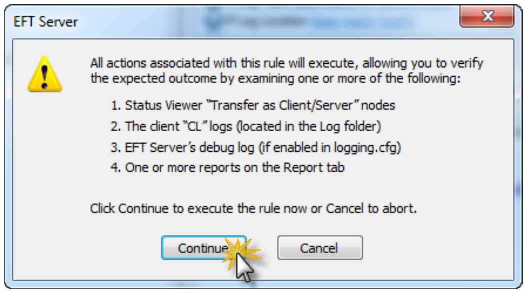
Use the up and down arrows to change the sequence in which selected Actions and Conditions occur.

15. To test your event rule, click **Run Now**.



Use the up and down arrows to change the sequence in which selected Actions and Conditions occur.

16. Click **Continue**.



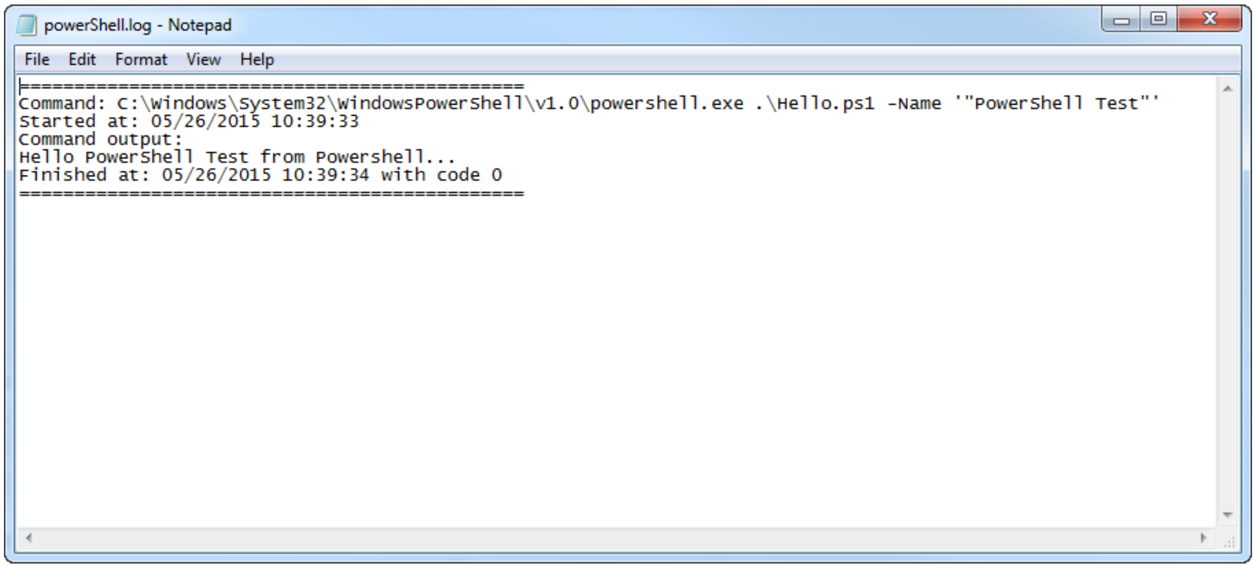
EFT Server

All actions associated with this rule will execute, allowing you to verify the expected outcome by examining one or more of the following:

1. Status Viewer "Transfer as Client/Server" nodes
2. The client "CL" logs (located in the Log folder)
3. EFT Server's debug log (if enabled in logging.cfg)
4. One or more reports on the Report tab

Click Continue to execute the rule now or Cancel to abort.

17. Verify your PowerShell has executed correctly. Open your output log file from C:\temp\logs\powerShell.log using Notepad



```
==============================================
Command: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe .\Hello.ps1 -Name '"PowerShell Test"'
Started at: 05/26/2015 10:39:33
Command output:
Hello PowerShell Test from Powershell...
Finished at: 05/26/2015 10:39:34 with code 0
==============================================
```

Remember you can pass any context variable to the script so in cases where you have FileUploads or Folder Monitor you can process files. PowerShell has built-in a very large number of CmdLets already available that you can use.

See https://technet.microsoft.com/en-us/library/dd772285.aspx

Some of the popular cmdlets and some applications are:

- Conditional statements, If, switch,
- File Folder Paths Manipulation
- Do while, while, do until, for, foreach
- File and Folder operations: Copy, Mode, Delete, Move folders, replicate, File/Folder Exist, File Attributes
- Database query
- Windows PowerShell Utility Cmdlets
    - Full support for XML, Text Files, CSV, etc.
    - Send Emails
    - Invoke Web Request
- Write Windows Event Log
- Get Process