The Experts Conference

Explore the Outer Limits.

Integrating Powershell with Legacy Environments

Sean Kearney – “The Energized Tech”

MCSE,MCTS,Microsoft MVP:Powershell

Member of Springboard Technical Experts Program

Presented By: QUEST SOFTWARE Sponsored By: Microsoft

GET-MYTH | INVOKE-FACT

Beliefs

- “You can’t work with vbScript from Powershell easily.”
- “I have to redo everything just to use Powershell”
- “You can’t pass data and return values to Powershell from Console or vbScript”
- “Powershell can’t return a status to vbScript or Console”

Facts

- CSCRIPT.EXE
- CMD.EXE
- POWERSHELL.EXE
- All are just applications that can send/receive parameters and return status

Launching our Scripts

GET-LEGACY | INVOKE-POWERSHELL

CMD.EXE

POWERSHELL.EXE –file *Script.PS1* –executionpolicy *RemoteSigned* **Param1**

Results Stored in ***%ERRORLEVEL%***

Default (***0 = Good*, *1 = Bad)*** or Values can be returned with “**RETURN**” or “**EXIT**” in Powershell

Launching our Scripts

GET-LEGACY | INVOKE-POWERSHELL

VBSCRIPT

DIM OBJSHELL=Wscript.NewObject(“Wscript.Shell”)

APP=“**POWERSHELL.EXE –executionpolicy RemoteSigned –file** “

Params=“**Param1**”

Result=OBJSHELL.RUN(***APP & Params***)

Result stored in ***“Result”*** variable

Default (***0 = Good***, ***1 = Bad)*** or Values can be returned with “**RETURN**” or “**EXIT**” in Powershell

Launching our Scripts

GET-POWERSHELL | INVOKE-LEGACY

CONSOLE

```
CMD.EXE /C Script.CMD $Param1
& ‘C:\ApplicationFolder\ApplicationName.EXE‘ $Param1
```

VBSCRIPT

```
CSCRIPT.EXE Script.vbs $Param1
```

Results stored in **$LASTEXITCODE**

Default (0 = Good – 1 = Bad) or Values can be returned with **“WSCRIPT.QUIT”** in VBSCRIPT or **EXIT** from the CONSOLE (CMD.EXE)

Passing Parameters

Parameters passed to Powershell from **CMD.EXE** or **vbScript** will ALWAYS be received as **[STRING]** in Powershell

Parameters passed to **VBSCRIPT** and **CMD.EXE** from Powershell must be “Sanitized” back to **[STRING]** to ensure the data is what you *WANT*. Remember it works in Objects

In both scenarios “**EXIT XX**” can return a Numeric Status code

Everything is an Object!

Other option not realized by many. All output in Powershell is an **OBJECT**

(*including Console Applications*)

This means data **CAPTURED** from a Console Application (CMD.EXE or VBSCRIPT) can be **manipulated** and used by Powershell as well as sending back **results** to the original application