IN719 Systems Administration
Introduction to PowerShell

March 13, 2013

Introduction

When a sysadmin needs to execute a repetitive task, the right thing to do is to write a script. It saves time and ensures that the task is done accurately. In the past, the scripting options available on Windows were very limited, but with *PowerShell* it is now possible to write scripts that allow us to automate even very complex Active Directory tasks.

1 Basic PowerShell

Start PowerShell by clicking the blue command-prompt icon on the taskbar. From the command prompt you can invoke Windows executables and special PowerShell commands, or *cmdlets* (Pronounced “command-lets”). For example, try typing `notepad` at the prompt. To try something more interesting, type `Get-Process` to see information about running processes on the server.

You can also save results to variables. For example, try `$foo = Get-Process`. Now `$foo` contains an array where each entry is a line of information about one process. Type `$foo[11]` to see one of the entries.

2 Script files

PowerShell scripts can be stored in plain text files that end in the `.ps1` extension. Write a script the contains only the `Get-Process` line and save it in `C:Users\your_username\sample.ps1`. Before you can invoke it, you may need to enter `Set-ExecutionPolicy RemoteSigned` in your shell. Now Execute your script by entering `.\sample.ps1`.

Now consider the script below (also available at
`I:\COURSES\AITEIT3\BITY3\IN719 Systems Administration\Week4\sample.ps1`):

```
1 $wmi = Get-WmiObject Win32_Service
2 foreach ($s in $wmi) {
3   if ($s.name -eq "DNS") {
4     Write-Host "Found DNS"
5    break
6  } else {
7    Write-Host "Nope"
8  }
9 }
```

On line 1 we get a list of objects representing the running services on the server and assign it to the variable `$wmi`. From line 2 until line 9 we loop over that list, assigning each service object instance to `$s`. On line 3 we check the name property of `$s` to see if its value is "DNS". If it is, we write "Found DNS" to the console and break out of the loop on the next line. Otherwise, we hit the else clause on line 6 and write "Nope" to the console.

This example shows variables, loops, and conditionals. This should cover about 80% of your scripting needs. What remains is to consult PowerShell documents to learn how to access and manipulate Active Directory objects.