Advanced Scripting   
Modules

Last Updated: 6/16/2022 11:03 PM Version 1  
Document Prepared for: CIT361 Student

# Name Daniel Harris ID 235868292

# Instructions

Save a copy of this document. Answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

PowerShell provides a module system to add libraries of functions to your shell. You can install modules manually or via a repository. You can also create your own repositories.

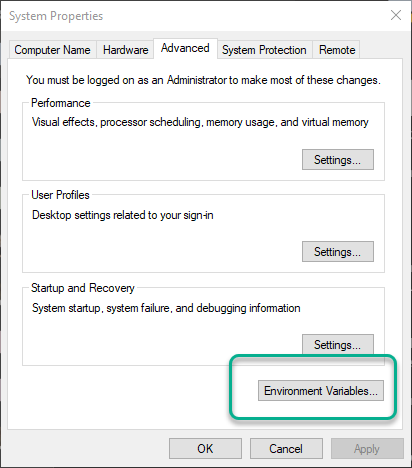
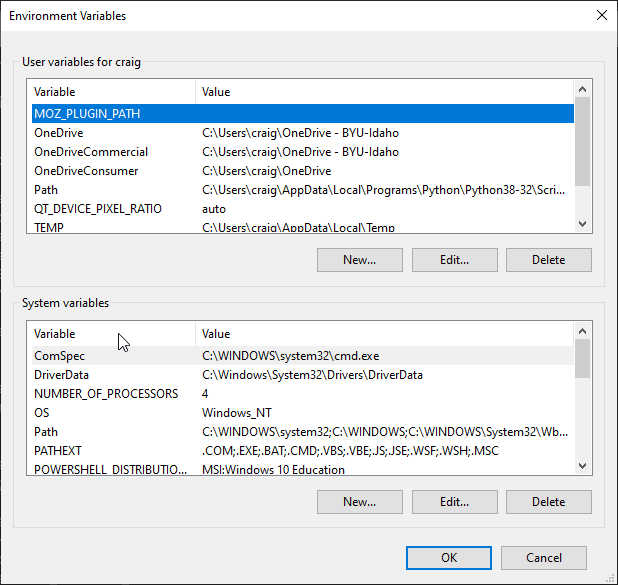
# Requirements

PowerShell

# Setup

# Task 1—Exploring the Module System

## Steps

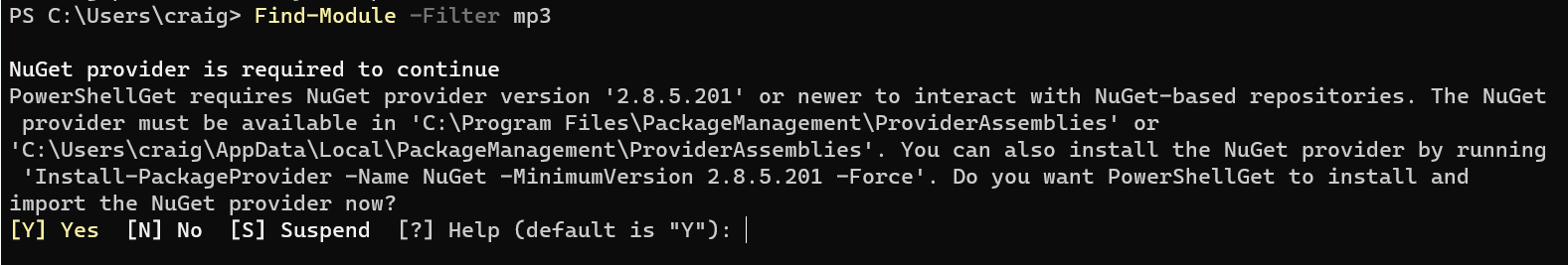
1. Open a new instance of PowerShell.
2. Get a list of modules that are currently loaded in PowerShell  
   Get-Module
   1. What modules are listed? Microsoft.PowerShell.Management Microsoft.PowerShell.Utility PSReadline
   2. BTW if you are typing this stuff in you don’t get it yet 😊 Make life easier with.
      1. (Get-Module|% name|Set-Clipboard) if you are not on Windows the Set-Clipboard part won’t work.
      2. Or even better if you want a comma separated list (Get-Module|% name) -join ', '|Set-Clipboard
3. Load a module dynamically. Enter the command  
   Get-Alias
4. Now get a list of loaded modules.  
   Get-Module
   1. What new module was loaded? no
5. List all the modules that are available to be dynamically loaded. This list changes as you add modules to the system.  
   get-module -ListAvailable
   1. Record all the available module names? powershell-yaml  
      Microsoft.PowerShell.Operation.Validation  
      PackageManagement  
      Pester  
      PowerShellGet  
      PSReadline  
      AppBackgroundTask  
      AppLocker  
      AppvClient  
      Appx  
      AssignedAccess  
      BitLocker  
      BitsTransfer  
      BranchCache  
      CimCmdlets  
      ConfigCI  
      ConfigDefender  
      ConfigDefenderPerformance  
      Defender  
      DeliveryOptimization  
      DirectAccessClientComponents  
      Dism  
      DnsClient  
      EventTracingManagement  
      HostNetworkingService  
      International  
      iSCSI  
      ISE  
      Kds  
      LanguagePackManagement  
      LAPS  
      Microsoft.PowerShell.Archive  
      Microsoft.PowerShell.Diagnostics  
      Microsoft.PowerShell.Host  
      Microsoft.PowerShell.LocalAccounts  
      Microsoft.PowerShell.Management  
      Microsoft.PowerShell.ODataUtils  
      Microsoft.PowerShell.Security  
      Microsoft.PowerShell.Utility  
      Microsoft.WSMan.Management  
      MMAgent  
      MsDtc  
      NetAdapter  
      NetConnection  
      NetEventPacketCapture  
      NetLbfo  
      NetNat  
      NetQos  
      NetSecurity  
      NetSwitchTeam  
      NetTCPIP  
      NetworkConnectivityStatus  
      NetworkSwitchManager  
      NetworkTransition  
      PcsvDevice  
      PersistentMemory  
      PKI  
      PnpDevice  
      PrintManagement  
      ProcessMitigations  
      Provisioning  
      PSDesiredStateConfiguration  
      PSDiagnostics  
      PSScheduledJob  
      PSWorkflow  
      PSWorkflowUtility  
      ScheduledTasks  
      SecureBoot  
      SmbShare  
      SmbWitness  
      StartLayout  
      Storage  
      StorageBusCache  
      TLS  
      TroubleshootingPack  
      TrustedPlatformModule  
      UEV  
      VpnClient  
      Wdac  
      Whea  
      WindowsDeveloperLicense  
      WindowsErrorReporting  
      WindowsSearch  
      WindowsUpdate
      1. I hope you didn’t type that in!
6. The environment variable PSModulePath contains a list of directories separated by semicolons (or colons on Linux) that PowerShell will look in to find modules. View yours.  
   $env:PSModulePath
   1. Now make it easier to read  
      $env:PSModulePath -split ';'
      1. Are you getting the hang of it yet?
      2. How many directories are in your module path. 2. I hope you didn’t count them by hand!
7. Changing the PSModulePath environment variable.
   1. In windows environment variables are set at either the machine or user level.
      1. You can set it from the control panel system properties Environment variables button. (takes effect next time you open a shell)  
          
      2. Using the Setx command. Use Setx /? for help (takes effect next time you open a shell)
      3. Modifying the registry keys from PowerShell (takes effect next time you open a shell)
      4. Modify the $env:PSModulePath variable from PoweShell (only takes effect in current session)
   2. In Linux
      1. Add export commands to .bashrc for persistent variables (takes effect next time you open a shell)
      2. Use the export command for current shell (only take effect in current session)
      3. Modify the $env:PSModulePath variable from PoweShell (only take effect in current session)

# Task 2—Finding and Installing Modules in a Repository

By default, PowerShell is configured to access the public PowerShell repository (PSGallery). As of this writing there are nearly 7000 packages available. The gallery is at <https://www.powershellgallery.com/>.

You can search for modules or commands in PowerShell or some repositories (like PSGallery) also provide a web based search tool.

## Steps

1. Finding modules using PowerShell. Just like discovering commands the key here is to think of a good keyword that describes what you are looking for. Once you have done that use the Find-Module command or Find-Command command to locate a module. Say you want to find a module that will allow you to work with mp3 tags. Use the find-module command to see what is there.  
   Find-Module -Filter mp3
2. NOTE: The first time you use the find or install-module cmdlets you will be prompted to install the NuGet provider. This is necessary to work with repositories. Enter **Y**
3. This returns a list of all the modules that contain ‘mp3’. Notice the module named id3 install it for the current user  
   Install-Module id3 -Scope CurrentUser
   1. What commands does this module contain?
4. Use the command(s) in the module to discover the following information about the mp3 file **psfiles\files\Amazing\_Grace.mp3**

|  |  |
| --- | --- |
| Tag | Value |
| Title | Amazing Grace 2011 |
| Album | Classical Sampler |
| BeatsPerMinute | 79 |
| FirstArtist | Kevin MacLeod |
| FirstGenre | Classical |

1. Find a module of interest to you, install it, answer the following questions
   1. What is the module name? GetBios
   2. What interested you about this module? It was in the top 100 packages that wasn’t azure related
   3. What commands are in this module separate commands with a comma? Get-bios
2. Visit the PowerShell repository at <https://www.powershellgallery.com/>. Get familiar with the site.
   1. What is the post popular download? PackageManagement

# Task 3— Saving a Module from the Repository.

Since the repository is public there are varying qualities of code in the repository. If you are in doubt about the quality of a module you can download it before you install it.

## Steps

1. Save a module from the repository. Let’s look at the code of the ID3 module.
2. Change to you psfiles directory.
3. Save the module  
   Save-Module -Name id3 -Path .
4. This will create a folder for the module’s files named ID3. Change to that folder the get a directory listing.
   1. What directory(ies) do you see? ID3, version number
5. Change to the highest numbered directory. Get a directory.
   1. What files to you see? 2 Powershell relate files and a dll
6. .psm1 files are module scripts, .psd1 files are PowerShell data files, .ddl are external libraries that are needed for the module.
7. View the contents of the .psm1 file. You should see the functions that you saw when you listed the commands from the module earlier.

# Task 4— Manual Module Loading and Unloading

You can load a module manually without installing it.

## Steps

1. Manually install the crypto library. There is a file named crypto.zip in the psfiles\files folder. You will extract it now. Change to your psfiles\files folder.  
   Expand-Archive .\crypto.zip .
2. You should see a folder named crypto which has a subfolder named 1.0 that contains two files crypto.psm1 and crypto.psd1. The .psd1 is the module manifest(more on that later), the .psm1 is the actual module file. You can use the Import-Module cmdlet to load a module. If the module has a .psd1 file load it, if not load the psm1 file. Load the module  
   Import-Module .\files\crypto\1.0\crypto.psd1
   1. Notice you will get a warning about unapproved verbs. This means that some of the functions have names that don’t follow the PowerShell naming conventions. You can override the warning by adding the -DisableNameChecking parameter.
3. List the modules, you should see the crypto module.  
   Get-Module
4. What commands are in the module? ConvertTo-Hex  
   Decrypt-String  
   Encrypt-String  
   Get-CaesarCypher  
   Get-DesKey  
   Get-MD5  
   Get-StringHash
5. Use the Get-MD5 to get the MD5 hash of the word foo  
   Get-MD5 foo
   1. What is the MD5 has of foo? ACBD18DB4CC2F85CEDEF654FCCC4A4D8
6. You can also unload a module. Unload crypto with the Remove-Module command  
   Remove-Module crypto
7. Try running getting a hash of foo again.  
   Get-MD5 foo
   1. What was the result? It didn’t work
8. Get a list of the loaded modules. Is crypto there? It is not

# Task 5—Manual Module Installation and Removal

You can also install a module manually. Since crypto is such a great utility you can install it to either your profile or the entire system. All you need to do is copy the entire module folder to one of the directories in your PsModulesPath.

## Steps

1. List your module paths (remember if you are on Linux or OSX use a : rather than ;.   
   $env:PSModulePath -split ';'
2. Find a path that is in your home directory. This is your local module path. Don’t tell me you looked through the list. Let PowerShell find it for you.   
   $env:PSModulePath -split ';'|? {$\_ -like "$(Resolve-Path ~)\*"}
3. Copy the contents of the file crypto.zip to your modules folder however you would like, but this will work if you are still in the psfiles\files directory.
4. First store the modules path in a variable  
   $mp=$env:PSModulePath -split ';'|? {$\_ -like "$(Resolve-Path ~)\*"}
5. Then make sure the path exists  
   Test-Path $mp
   1. If this returns false, you need to create the path.  
      New-Item -Type Directory $mp
6. Now expand the new folder  
   Expand-Archive .\crypto.zip $mp
7. Try it out and see if it works  
   Get-md5 foo
8. Now get rid of the module manually. Delete the crypto folder from your modules directory. You should still have your module directory in the $mp variable  
   Remove-Item $mp -Force
9. Try Get-MD5 foo again. Did it work? It was successful Why or why not? Because its still loaded in memory for this session
10. Open a new PowerShell Shell, try Get-MD5 foo again. It didn’t’w work What does this tell you? Session based module loading

# Task 7—Registering a module repository

We have a private module repository for this class. In this exercise you will register the repository so you can search, install, and publish to it.

## Steps

1. Register the class repository (this is all on one line)  
   Register-PSRepository -Name CIT361   
    -SourceLocation http://cit361-lab.cit.byui.edu:8624/nuget/CIT361/   
    -PublishLocation http://cit361-lab.cit.byui.edu:8624/nuget/CIT361/   
    -InstallationPolicy Trusted
2. List all the packages in the repository   
   Find-Module -Repository cit361
   1. What version is the module LindyConfig? 1.0.0.5

# Deliverable

Upload this document with completed answers to i-learn.