Advanced Scripting   
Modules

Last Updated: 4/18/2024 10:44 AM Version 1  
Document Prepared for: CYBER360 Student

# Name Click here to enter name ID Click here to enter id

# 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.

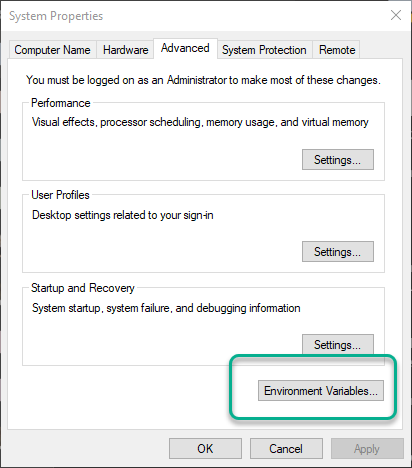
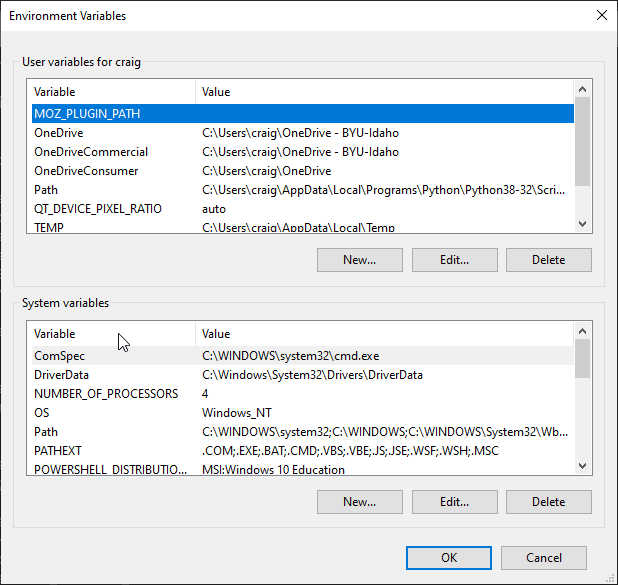
# Requirements

PowerShell on Windows (Desktop or Core) for tasks 1.   
PowerShell (any platform) for tasks 2 through 7.   
Class example files from the **psfiles.zip** archive.

# Setup

# Task 1—Exploring the Module System

## Steps

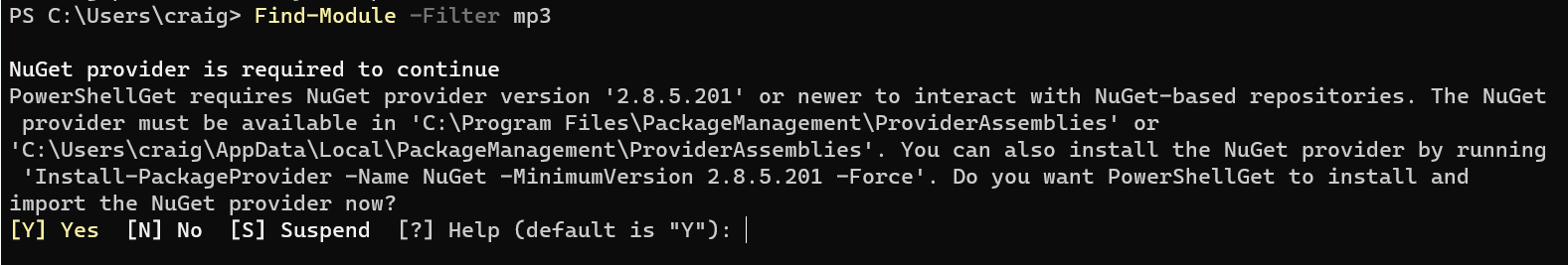
1. Open a new instance of PowerShell.
2. Get a list of modules that are currently loaded in PowerShell  
   Get-Module
   1. What modules are listed (one line, comma separated)? Click or tap here to enter text.   
      *By the way, did you know PowerShell on Windows can pipe output directly to the clipboard?*   
      Try this:   
      **Get-Module|% Name|Set-Clipboard** # *then paste into some other app*.   
      For a comma-separated list, try this:   
      **(get-module|% name) -join ', '|set-clipboard**
3. Load a module dynamically by entering one of a modules exported commands:   
   Get-WidowsUpdateLog   
   (By default, this cmdlet gathers and converts update logs, placing a report file on your desktop. It may take a minute or so to complete.)
4. Now get a list of loaded modules.   
   Get-Module
   1. What new module was loaded? Click or tap here to enter text.
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 (one line, comma separated)? Click or tap here to enter text. *Note: I sure hope you didn’t type all that in!*
6. The environment variable **PSModulePath** contains a list of directories separated by semicolons (or colons on Linux/macOS) that PowerShell will look in to find modules. View yours.   
   $env:PSModulePath -split ';'
   * 1. How many directories are in your module path? Click or tap here to enter text.. *Note: again, I hope you made PowerShell count them for you, and you didn’t count them by hand!*
7. Here’s how to change the **PSModulePath** environment variable:
   1. In Windows, environment variables are set at either the machine or the user level.
      1. You can set it from Control Panel. System Properties, [Environment Variables…] button. (Changes take effect next time you open a shell.)   
          
      2. Or, use the **setx** command. Enter **setx /?** for help. (Changes take effect next time you open a shell.)
      3. Or, modify registry keys from PowerShell. (Not shown here. Changes take effect next time you open a shell.)
      4. Or, modify the **$env:PSModulePath** variable in PowerShell. (Changes only affect current session!)
   2. In Linux:
      1. Add suitable **export** commands to a login startup script, such as **.bashrc**, for persistent environment variables. (Changes take effect next time you open a shell.)
      2. Or, use **export** command in current shell (Only affects current shell session.)
      3. Or, modify the **$env:PSModulePath** variable from PowerShell. (Only affects current **pwsh** session.)

# Task 2—Finding and Installing Modules from 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. 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** or **Find-Command** cmdlets to locate a module. Suppose 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-Module 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? Click or tap here to enter text.
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 | Click or tap here to enter text. |
| Album | Click or tap here to enter text. |
| BeatsPerMinute | Click or tap here to enter text. |
| FirstArtist | Click or tap here to enter text. |
| FirstGenre | Click or tap here to enter text. |

1. Find a module of interest to you. Install it, then answer the following questions:
   1. What is the module’s name? Click or tap here to enter text.
   2. What interested you about this module? Click or tap here to enter text.
   3. What commands are in this module? (one line, comma separated) Click or tap here to enter text.
2. Visit the PowerShell repository at <https://www.powershellgallery.com/>. Get familiar with the site.
   1. Which module is currently the most popular download? Click or tap here to enter text.

# 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 and examine it before you install it.

## Steps

1. You can save a module from the repository. Let’s look at the code of the ID3 module.
2. Change to your psfiles directory.
3. Save the module:   
   **Save-Module -Name id3 -Path .**
4. This will create a subfolder for the module’s files named **ID3**. Change to that subfolder, then get a directory listing.
   1. What subdirectory(ies) do you see? Click or tap here to enter text.
5. Change to the highest numbered subdirectory. Get a directory listing there.
   1. What files do you see? Click or tap here to enter text.
   2. **.psm1** files are module scripts. **.psd1** files are PowerShell data files. **.dll** files have “dynamic link library” external shared code objects which are needed for the module.
6. View the contents of the **.psm1** file. You should see definitions for the functions that you saw when you listed the commands from the module in the previous task.

# 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 .   
   Note: this archive does NOT contain malware, but a few antivirus software packages are known to quarantine this file’s contents, likely because they see *crypto* in the name. You may need to manually adjust your security software for some steps in this task.)
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 manifests later.) The **.psm1** file 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, just load the psm1 file. Manually load the crypto module:   
   Import-Module .\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 suggested PowerShell naming conventions You can override the warning by adding the **-DisableNameChecking** switch parameter.
3. List the modules. You should see the crypto module.   
   Get-Module
4. What commands are in the module? (one line, comma separated) Click or tap here to enter text.
5. Use the Get-MD5 cmdlet to get the MD5 hash of the word **foo**:   
   Get-MD5 foo
   1. What is the MD5 has of foo? Click or tap here to enter text.
6. You can also unload a module. Unload **crypto** with the **Remove-Module** cmdlet:   
   Remove-Module crypto
7. Now try running getting a hash of **foo** again.   
   Get-MD5 foo
   1. What was the result? Click or tap here to enter text.
8. Get a list of the loaded modules. Is crypto there? Click or tap here to enter text.

# 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 to 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 using Linux or macOS, use **:** rather than **;**.)   
   $env:PSModulePath -split ';'
2. Find a path that is in your home directory. This is your local module path. Don’t just look through the list; let PowerShell find it for you!   
   $env:PSModulePath -split ';'|? {$\_ -like "$(Resolve-Path ~)\*"}
3. Copy the contents of the **crypto.zip** archive to your modules folder. Do so however you would like; the following steps 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, so try this:   
   Remove-Item $mp -Force
9. Try Get-MD5 foo again.   
   Did it work? Click or tap here to enter text.   
   Why or why not? Click or tap here to enter text.
10. Open a new PowerShell session. Try **Get-MD5 foo** again.   
    Did it work? Click or tap here to enter text.   
    Explain: Click or tap here to enter text.

# Task 7—Registering another 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. Enter (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? Click or tap here to enter text.

We will use this Repository in an upcoming lab.

# Deliverable

Upload this document with completed answers to I-Learn Canvas.