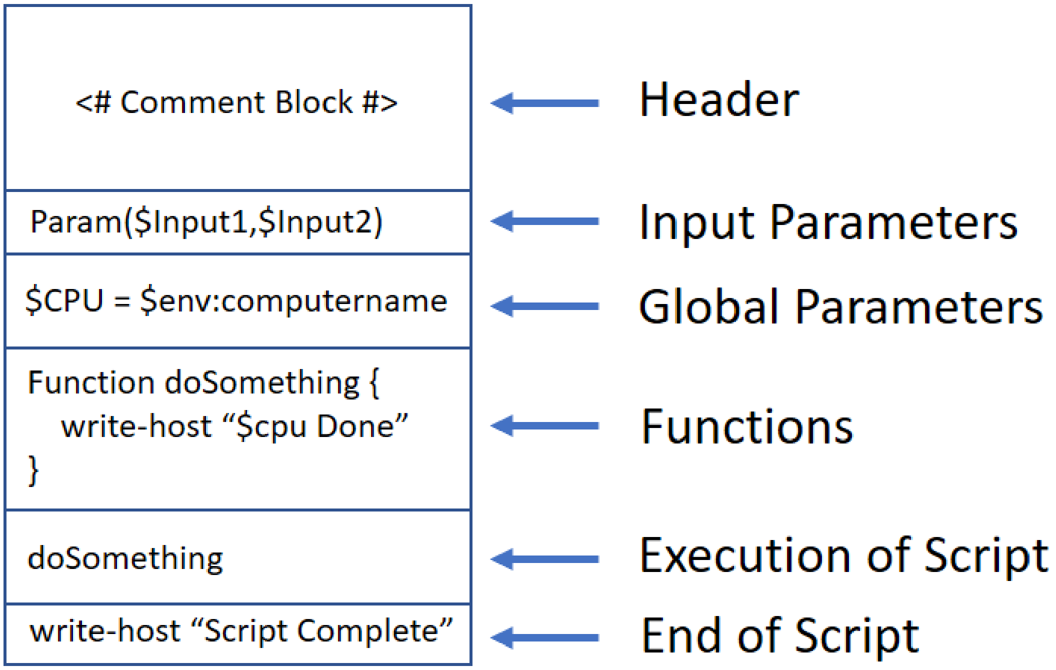
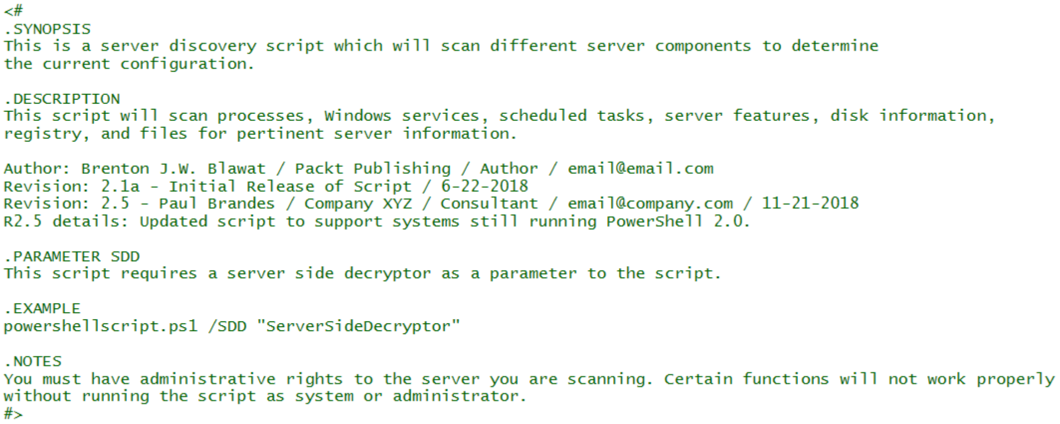
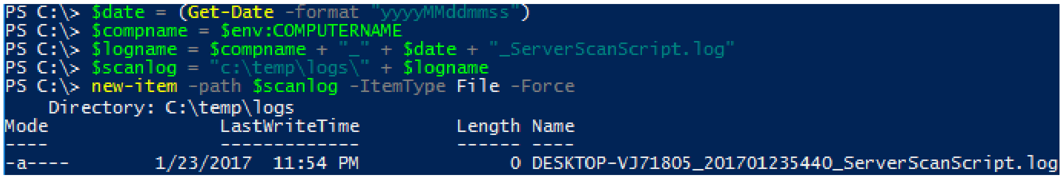
Chapter 2: Script Structure, Comment Blocks, and Script Logging



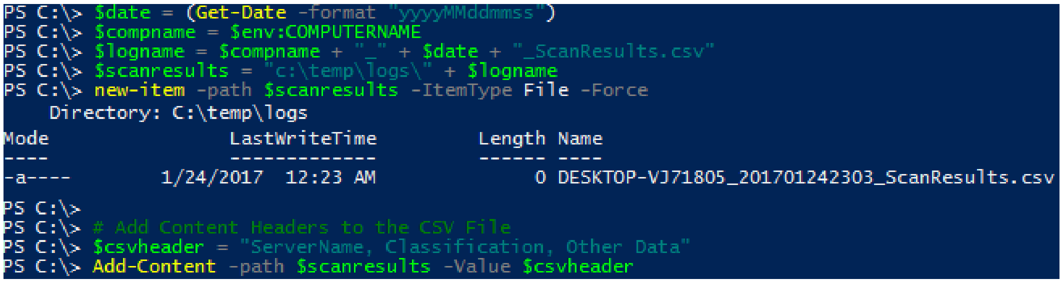
```
<# Comment Block #>
Param($Input1,$Input2)
$CPU = $env:computername
Function doSomething {
   write-host “$cpu Done”
}
doSomething
write-host “Script Complete”
```



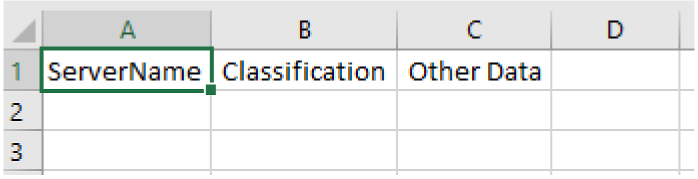
```
<#
.SYNOPSIS
This is a server discovery script which will scan different server components to determine
the current configuration.

.DESCRIPTION
This script will scan processes, Windows services, scheduled tasks, server features, disk information,
registry, and files for pertinent server information.

Author: Brenton J.W. Blawat / Packt Publishing / Author / email@email.com
Revision: 2.1a - Initial Release of Script / 6-22-2018
Revision: 2.5 - Paul Brandes / Company XYZ / Consultant / email@company.com / 11-21-2018
R2.5 details: Updated script to support systems still running PowerShell 2.0.

.PARAMETER SDD
This script requires a server side decryptor as a parameter to the script.

.EXAMPLE
powershellscript.ps1 /SDD "ServerSideDecryptor"

.NOTES
You must have administrative rights to the server you are scanning. Certain functions will not work properly
without running the script as system or administrator.
#>
```



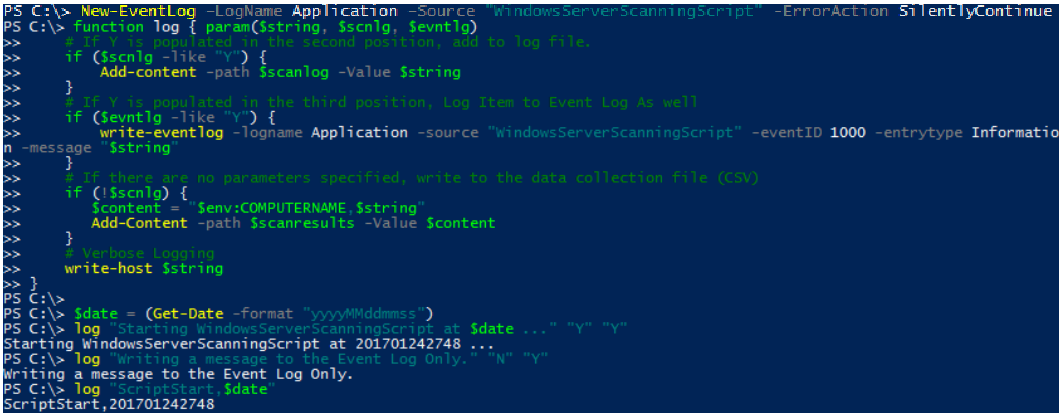
```
PS C:\> $date = (Get-Date -format "yyyyMMddmmss")
PS C:\> $compname = $env:COMPUTERNAME
PS C:\> $logname = $compname + "_" + $date + "_ServerScanScript.log"
PS C:\> $scanlog = "c:\temp\logs\" + $logname
PS C:\> new-item -path $scanlog -ItemType File -Force
    Directory: C:\temp\logs
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        1/23/2017  11:54 PM              0 DESKTOP-VJ71805_201701235440_ServerScanScript.log
```



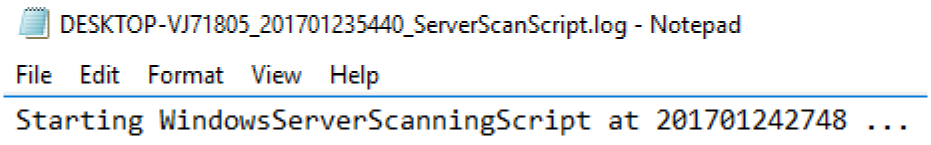
```
PS C:\> $date = (Get-Date -format "yyyyMMddmmss")
PS C:\> $compname = $env:COMPUTERNAME
PS C:\> $logname = $compname + "_" + $date + "_ScanResults.csv"
PS C:\> $scanresults = "c:\temp\logs\" + $logname
PS C:\> new-item -path $scanresults -ItemType File -Force
    Directory: C:\temp\logs
Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        1/24/2017  12:23 AM              0 DESKTOP-VJ71805_201701242303_ScanResults.csv

PS C:\>
PS C:\> # Add Content Headers to the CSV File
PS C:\> $csvheader = "ServerName, Classification, Other Data"
PS C:\> Add-Content -path $scanresults -Value $csvheader
```



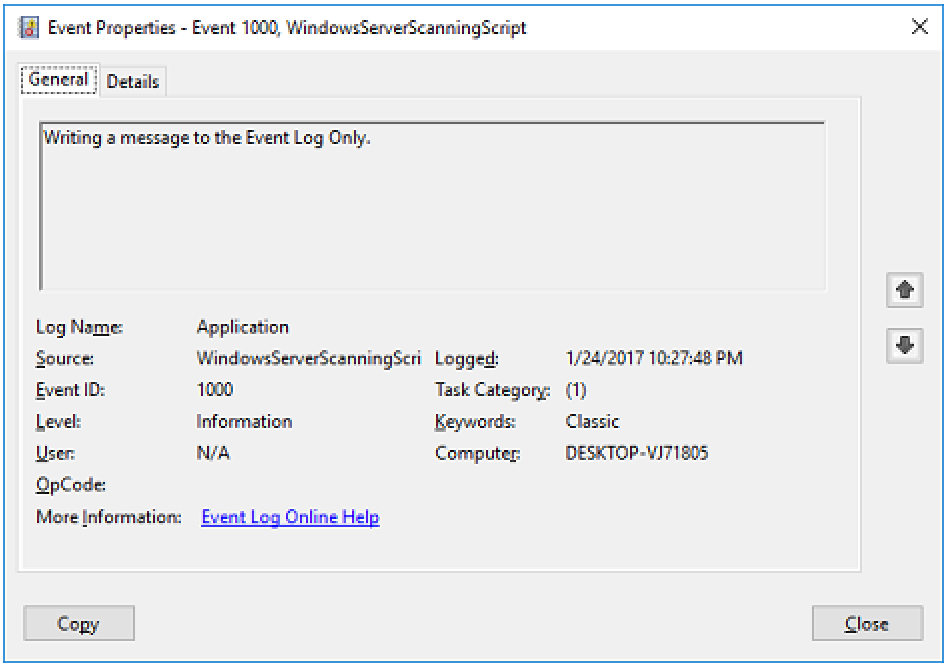
	A	B	C	D
1	ServerName	Classification	Other Data	
2				
3				



```
PS C:\> New-EventLog -LogName Application -Source "WindowsServerScanningScript" -ErrorAction SilentlyContinue
PS C:\> function log { param($string, $scnlg, $evntlg)
>>     # If Y is populated in the second position, add to log file.
>>     if ($scnlg -like "Y") {
>>         Add-content -path $scanlog -Value $string
>>     }
>>     # If Y is populated in the third position, Log Item to Event Log As well
>>     if ($evntlg -like "Y") {
>>         write-eventlog -logname Application -source "WindowsServerScanningScript" -eventID 1000 -entrytype Informatio
n -message "$string"
>>     }
>>     # If there are no parameters specified, write to the data collection file (CSV)
>>     if (!$scnlg) {
>>        $content = "$env:COMPUTERNAME,$string"
>>        Add-Content -path $scanresults -Value $content
>>     }
>>     # Verbose Logging
>>     write-host $string
>> }
PS C:\>
PS C:\> $date = (Get-Date -format "yyyyMMddmmss")
PS C:\> log "Starting WindowsServerScanningScript at $date ..." "Y" "Y"
Starting WindowsServerScanningScript at 201701242748 ...
PS C:\> log "Writing a message to the Event Log Only." "N" "Y"
Writing a message to the Event Log Only.
PS C:\> log "ScriptStart,$date"
ScriptStart,201701242748
```



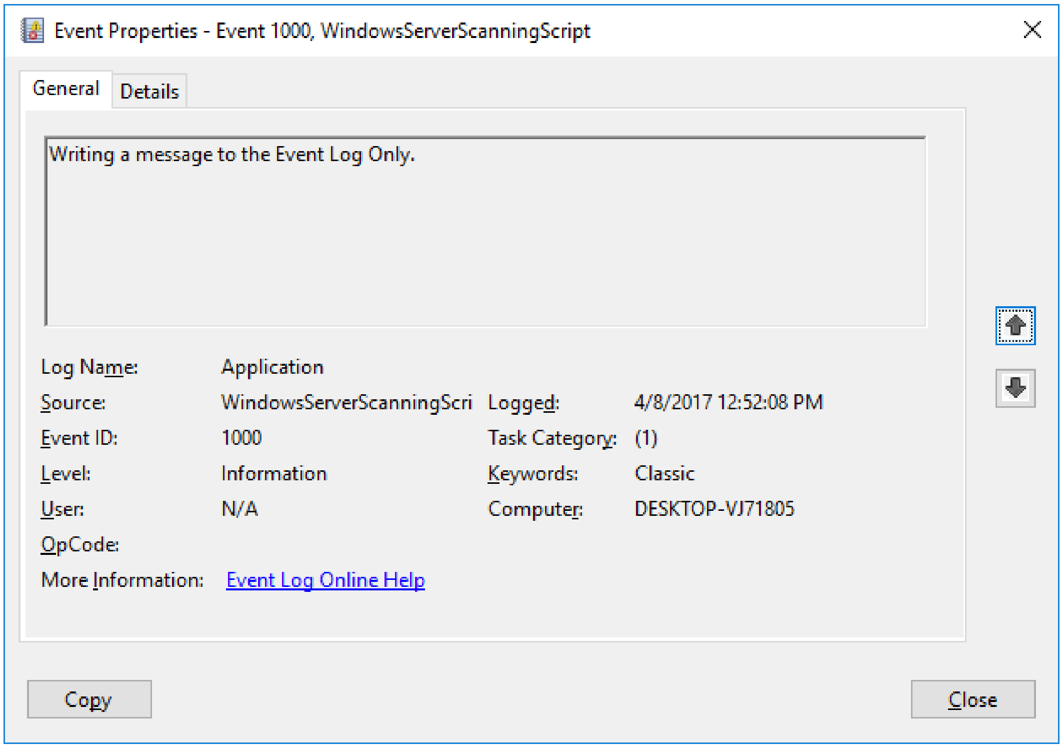
```
Starting WindowsServerScanningScript at 201701242748 ...
```



Event Properties - Event 1000, WindowsServerScanningScript

Writing a message to the Event Log Only.

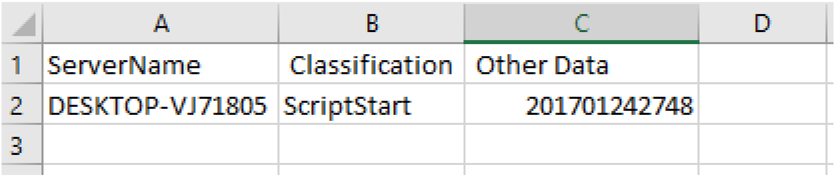
Log Name: Application
Source: WindowsServerScanningScri
Logged: 1/24/2017 10:27:48 PM
Event ID: 1000
Task Category: (1)
Level: Information
Keywords: Classic
User: N/A
Computer: DESKTOP-VJ71805
OpCode:
More Information: Event Log Online Help



Event Properties - Event 1000, WindowsServerScanningScript

Writing a message to the Event Log Only.

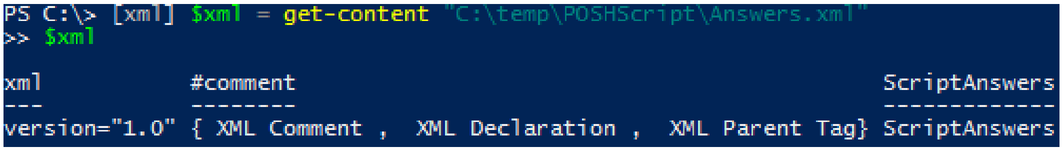
Log Name: Application
Source: WindowsServerScanningScri
Logged: 4/8/2017 12:52:08 PM
Event ID: 1000
Task Category: (1)
Level: Information
Keywords: Classic
User: N/A
Computer: DESKTOP-VJ71805
OpCode:
More Information: Event Log Online Help



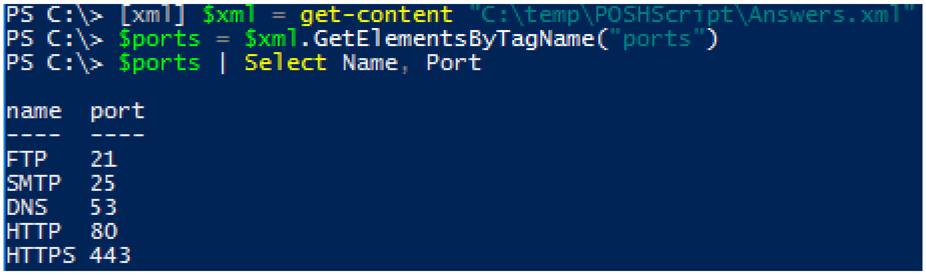
	A	B	C	D
1	ServerName	Classification	Other Data	
2	DESKTOP-VJ71805	ScriptStart	201701242748	
3				

Chapter 3: Working with Answer Files

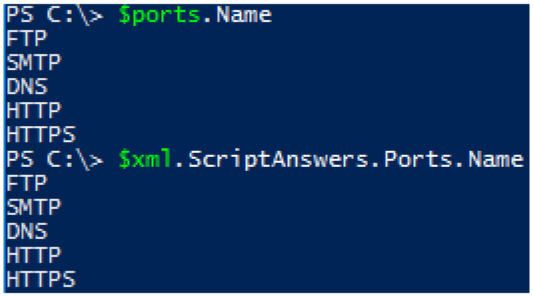
```
<?xml version="1.0"?>
<!-- XML Comment -->
<!-- XML Declaration -->


<!-- XML Parent Tag-->
<ScriptAnswers>

  <!-- XML Child Tag-->
  <ports port="21" name="FTP"></ports>

  <!-- XML Sibling Tags-->
  <ports port="25" name="SMTP"></ports>
  <ports port="53" name="DNS"></ports>
  <ports port="80" name="HTTP"></ports>
  <ports port="443" name="HTTPS"></ports>

<!-- XML Parent Closing Tag-->
</ScriptAnswers>
```



```
PS C:\> [xml] $xml = get-content "C:\temp\POSHScript\Answers.xml"
>> $xml

xml           #comment                                           ScriptAnswers
---           --------                                           -------------
version="1.0" { XML Comment ,  XML Declaration ,  XML Parent Tag} ScriptAnswers
```

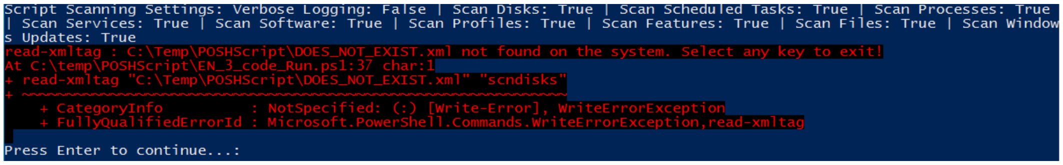


```
PS C:\> [xml] $xml = get-content "C:\temp\POSHScript\Answers.xml"
PS C:\> $ports = $xml.GetElementsByTagName("ports")
PS C:\> $ports | Select Name, Port

name  port
----  ----
FTP   21
SMTP  25
DNS   53
HTTP  80
HTTPS 443
```



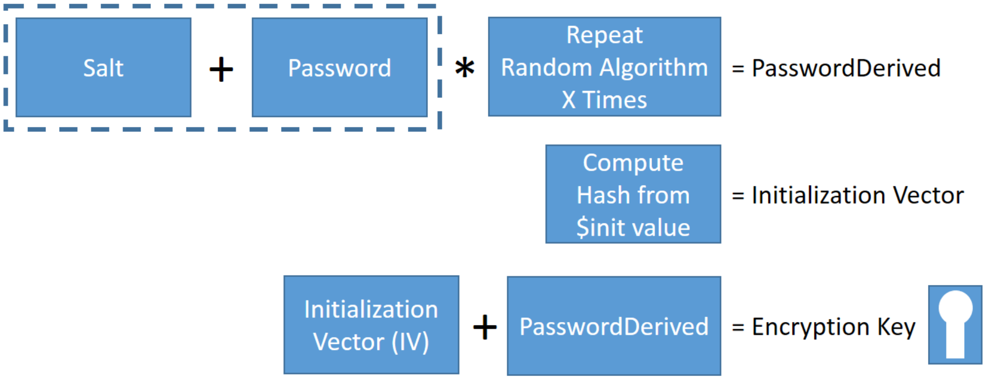
```
PS C:\> $ports.Name
FTP
SMTP
DNS
HTTP
HTTPS
PS C:\> $xml.ScriptAnswers.Ports.Name
FTP
SMTP
DNS
HTTP
HTTPS
```

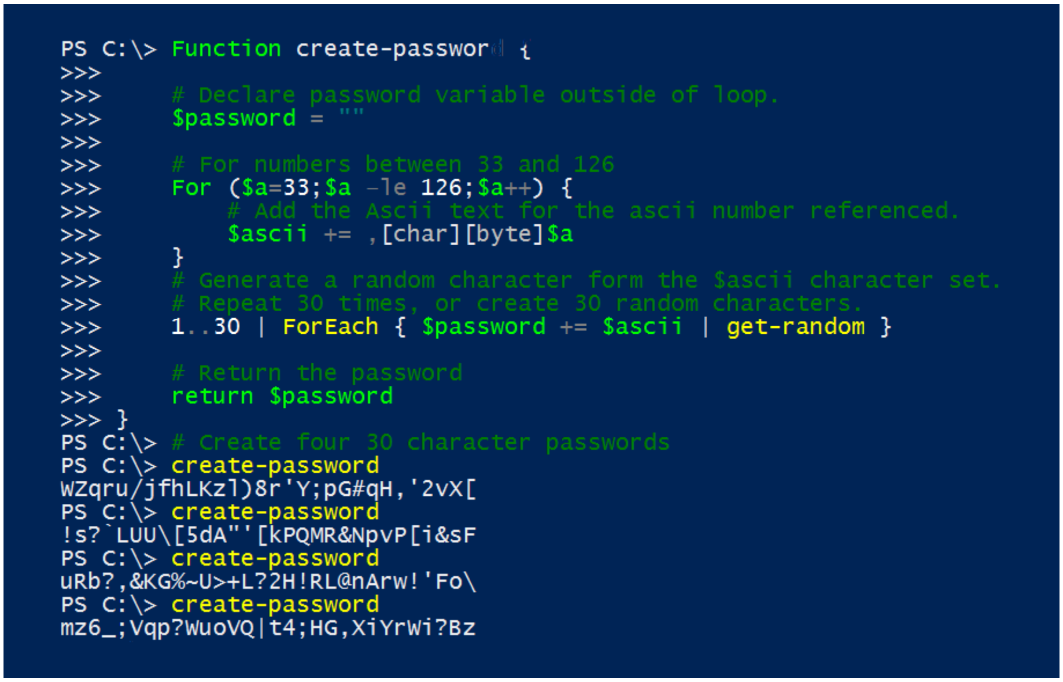
```
<?xml version="1.0"?>

<ScriptAnswers>

  <!-- This section enables and disables features in the script -->
  <!-- To Enable Logging For Each Step, Set To True. To Disable Logging, Set to False -->
  <verboselog id="False"></verboselog>
  <!-- Scan Disks -->
  <scndisks id="True"></scndisks>
  <!-- Scan Scheduled Tasks -->
  <scnschtsks id="True"></scnschtsks>
  <!-- Scan Processes -->
  <scnproc id="True"></scnproc>
  <!-- Scan Services -->
  <scnsvcs id="True"></scnsvcs>
  <!-- Scan Software -->
  <scnsoft id="True"></scnsoft>
  <!-- Scan User Profiles -->
  <scnuprof id="True"></scnuprof>
  <!-- Scan Windows Features -->
  <scnwfeat id="True"></scnwfeat>
  <!-- Scan Files -->
  <scnfls id="True"></scnfls>
  <!-- Scan Windows Updates -->
  <scnwupd id="True"></scnwupd>

</ScriptAnswers>
```



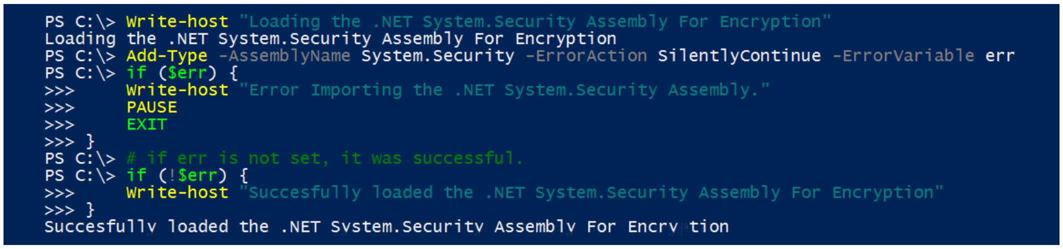
```
Script Scanning Settings: Verbose Logging: False | Scan Disks: True | Scan Scheduled Tasks: True | Scan Processes: True
| Scan Services: True | Scan Software: True | Scan Profiles: True | Scan Features: True | Scan Files: True | Scan Window
s Updates: True
read-xmltag : C:\Temp\POSHScript\DOES_NOT_EXIST.xml not found on the system. Select any key to exit!
At C:\temp\POSHScript\EN_3_code_Run.ps1:37 char:1
+ read-xmltag "C:\Temp\POSHScript\DOES_NOT_EXIST.xml" "scndisks"
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
    + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,read-xmltag

Press Enter to continue...:
```

Chapter 4: String Encryption and Decryption

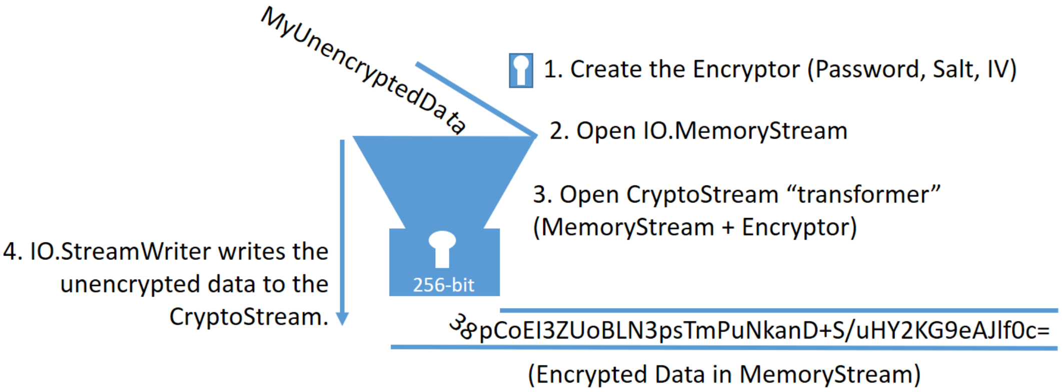

Salt
+
Password
*
Repeat Random Algorithm X Times
= PasswordDerived
Compute Hash from $init value
= Initialization Vector
Initialization Vector (IV)
+
PasswordDerived
= Encryption Key



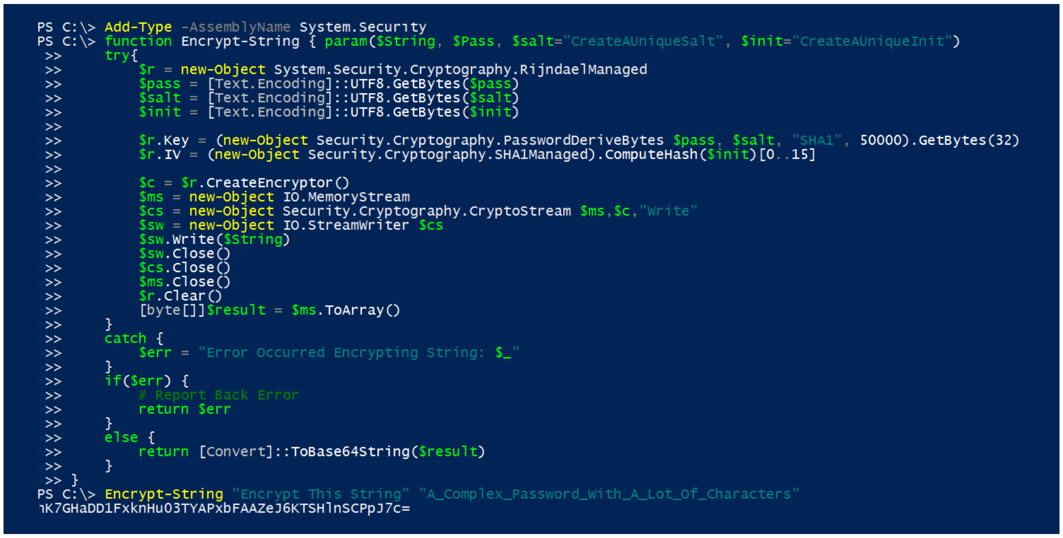
```
PS C:\> Function create-password {
>>>
>>>     # Declare password variable outside of loop.
>>>     $password = ""
>>>
>>>     # For numbers between 33 and 126
>>>     For ($a=33;$a -le 126;$a++) {
>>>         # Add the Ascii text for the ascii number referenced.
>>>         $ascii += ,[char][byte]$a
>>>     }
>>>     # Generate a random character form the $ascii character set.
>>>     # Repeat 30 times, or create 30 random characters.
>>>     1..30 | ForEach { $password += $ascii | get-random }
>>>
>>>     # Return the password
>>>     return $password
>>> }
PS C:\> # Create four 30 character passwords
PS C:\> create-password
WZqru/jfhLKzl)8r'Y;pG#qH,'2vX[
PS C:\> create-password
!s?`LUU\[5dA"'[kPQMR&NpvP[i&sF
PS C:\> create-password
uRb?,&KG%~U>+L?2H!RL@nArw!'Fo\
PS C:\> create-password
mz6_;Vqp?WuoVQ|t4;HG,XiYrWi?Bz
```



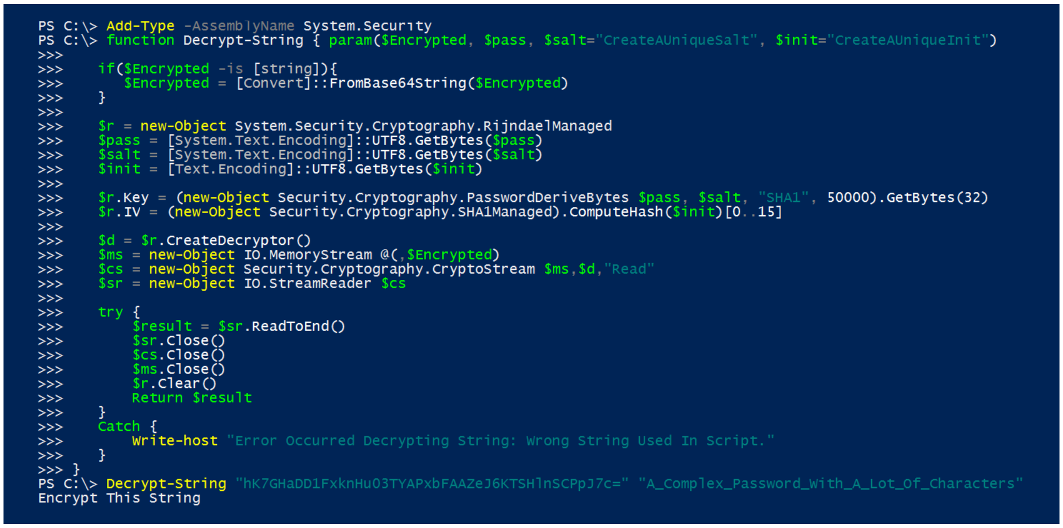
```
PS C:\> Write-host "Loading the .NET System.Security Assembly For Encryption"
Loading the .NET System.Security Assembly For Encryption
PS C:\> Add-Type -AssemblyName System.Security -ErrorAction SilentlyContinue -ErrorVariable err
PS C:\> if ($err) {
>>>     Write-host "Error Importing the .NET System.Security Assembly."
>>>     PAUSE
>>>     EXIT
>>> }
PS C:\> # if err is not set, it was successful.
PS C:\> if (!$err) {
>>>     Write-host "Succesfully loaded the .NET System.Security Assembly For Encryption"
>>> }
Succesfully loaded the .NET System.Security Assembly For Encryption
```



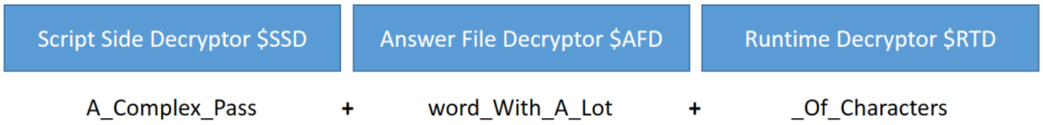
MyUnencryptedData
1. Create the Encryptor (Password, Salt, IV)
2. Open IO.MemoryStream
3. Open CryptoStream "transformer" (MemoryStream + Encryptor)
4. IO.StreamWriter writes the unencrypted data to the CryptoStream.
256-bit
38pCoEI3ZUoBLN3psTmPuNkanD+S/uHY2KG9eAJlf0c=
(Encrypted Data in MemoryStream)



```
PS C:\> Add-Type -AssemblyName System.Security
PS C:\> function Encrypt-String { param($String, $Pass, $salt="CreateAUniqueSalt", $init="CreateAUniqueInit")
>>      try{
>>          $r = new-Object System.Security.Cryptography.RijndaelManaged
>>          $pass = [Text.Encoding]::UTF8.GetBytes($pass)
>>          $salt = [Text.Encoding]::UTF8.GetBytes($salt)
>>          $init = [Text.Encoding]::UTF8.GetBytes($init)
>>
>>          $r.Key = (new-Object Security.Cryptography.PasswordDeriveBytes $pass, $salt, "SHA1", 50000).GetBytes(32)
>>          $r.IV = (new-Object Security.Cryptography.SHA1Managed).ComputeHash($init)[0..15]
>>
>>          $c = $r.CreateEncryptor()
>>          $ms = new-Object IO.MemoryStream
>>          $cs = new-Object Security.Cryptography.CryptoStream $ms,$c,"Write"
>>          $sw = new-Object IO.StreamWriter $cs
>>          $sw.Write($String)
>>          $sw.Close()
>>          $cs.Close()
>>          $ms.Close()
>>          $r.Clear()
>>          [byte[]]$result = $ms.ToArray()
>>      }
>>      catch {
>>          $err = "Error Occurred Encrypting String: $_"
>>      }
>>      if($err) {
>>          # Report Back Error
>>          return $err
>>      }
>>      else {
>>          return [Convert]::ToBase64String($result)
>>      }
>> }
PS C:\> Encrypt-String "Encrypt This String" "A_Complex_Password_With_A_Lot_Of_Characters"
ıK7GHaDD1FxknHu03TYAPxbFAAZeJ6KTSHlnSCPpJ7c=
```

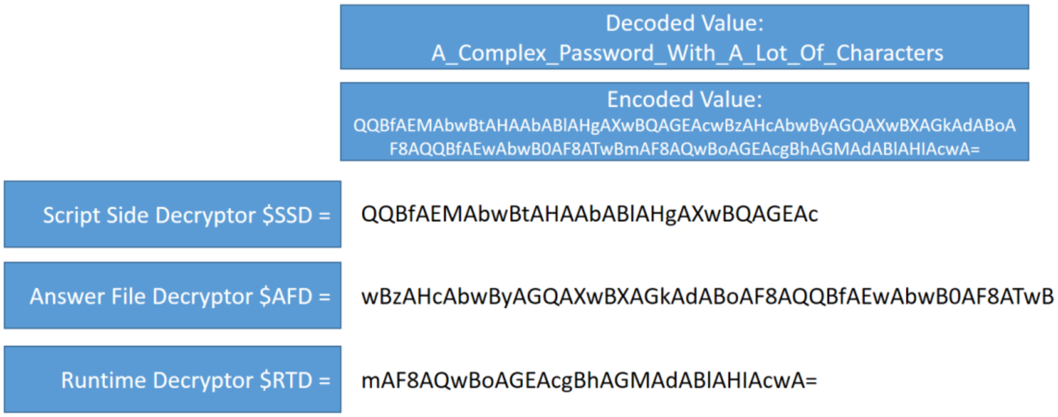


```
PS C:\> Add-Type -AssemblyName System.Security
PS C:\> function Decrypt-String { param($Encrypted, $pass, $salt="CreateAUniqueSalt", $init="CreateAUniqueInit")
>>>
>>>     if($Encrypted -is [string]){
>>>        $Encrypted = [Convert]::FromBase64String($Encrypted)
>>>     }
>>>
>>>     $r = new-Object System.Security.Cryptography.RijndaelManaged
>>>     $pass = [System.Text.Encoding]::UTF8.GetBytes($pass)
>>>     $salt = [System.Text.Encoding]::UTF8.GetBytes($salt)
>>>     $init = [Text.Encoding]::UTF8.GetBytes($init)
>>>
>>>     $r.Key = (new-Object Security.Cryptography.PasswordDeriveBytes $pass, $salt, "SHA1", 50000).GetBytes(32)
>>>     $r.IV = (new-Object Security.Cryptography.SHA1Managed).ComputeHash($init)[0..15]
>>>
>>>     $d = $r.CreateDecryptor()
>>>     $ms = new-Object IO.MemoryStream @(,$Encrypted)
>>>     $cs = new-Object Security.Cryptography.CryptoStream $ms,$d,"Read"
>>>     $sr = new-Object IO.StreamReader $cs
>>>
>>>     try {
>>>         $result = $sr.ReadToEnd()
>>>         $sr.Close()
>>>         $cs.Close()
>>>         $ms.Close()
>>>         $r.Clear()
>>>         Return $result
>>>     }
>>>     Catch {
>>>         Write-host "Error Occurred Decrypting String: Wrong String Used In Script."
>>>     }
>>> }
PS C:\> Decrypt-String "hK7GHaDD1FxknHu03TYAPxbFAAZeJ6KTSHlnSCPpJ7c=" "A_Complex_Password_With_A_Lot_Of_Characters"
Encrypt This String
```



Script Side Decryptor $SSD | Answer File Decryptor $AFD | Runtime Decryptor $RTD

A_Complex_Pass + word_With_A_Lot + _Of_Characters



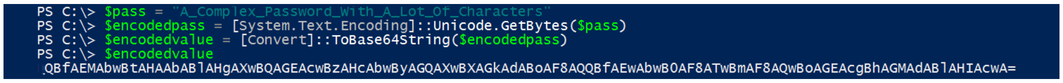
Decoded Value:
A_Complex_Password_With_A_Lot_Of_Characters

Encoded Value:
QQBfAEMAbwBtAHAAbABlAHgAXwBQAGEAcwBzAHcAbwByAGQAXwBXAGkAdABoA
F8AQQBfAEwAbwB0AF8ATwBmAF8AQwBoAGEAcgBhAGMAdABlAHIAcwA=

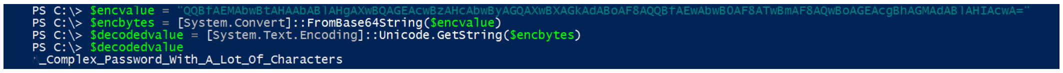
Script Side Decryptor $SSD = QQBfAEMAbwBtAHAAbABlAHgAXwBQAGEAc

Answer File Decryptor $AFD = wBzAHcAbwByAGQAXwBXAGkAdABoAF8AQQBfAEwAbwB0AF8ATwB

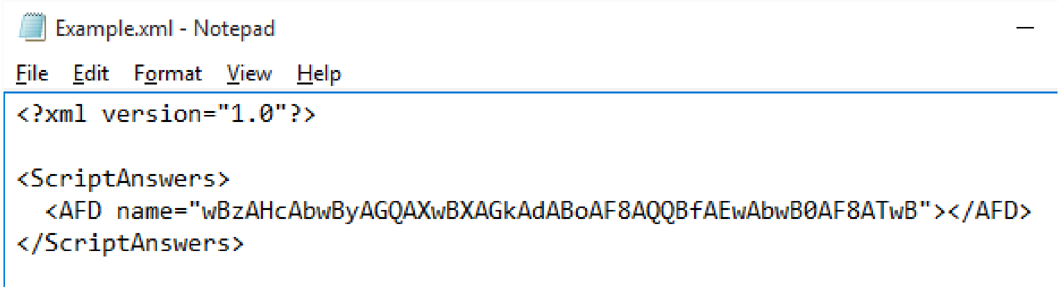
Runtime Decryptor $RTD = mAF8AQwBoAGEAcgBhAGMAdABlAHIAcwA=



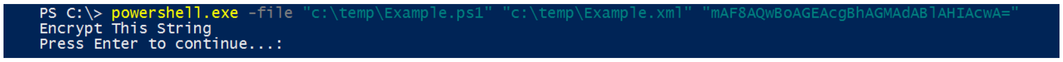
```
PS C:\> $pass = "A_Complex_Password_With_A_Lot_Of_Characters"
PS C:\> $encodedpass = [System.Text.Encoding]::Unicode.GetBytes($pass)
PS C:\> $encodedvalue = [Convert]::ToBase64String($encodedpass)
PS C:\> $encodedvalue
QBfAEMAbwBtAHAAbABlAHgAXwBQAGEAcwBzAHcAbwByAGQAXwBXAGkAdABoAF8AQQBfAEwAbwB0AF8ATwBmAF8AQwBoAGEAcgBhAGMAdABlAHIAcwA=
```



```
PS C:\> $encvalue = "QQBfAEMAbwBtAHAAbABlAHgAXwBQAGEAcwBzAHcAbwByAGQAXwBXAGkAdABoAF8AQQBfAEwAbwB0AF8ATwBmAF8AQwBoAGEAcgBhAGMAdABlAHIAcwA="
PS C:\> $encbytes = [System.Convert]::FromBase64String($encvalue)
PS C:\> $decodedvalue = [System.Text.Encoding]::Unicode.GetString($encbytes)
PS C:\> $decodedvalue
_Complex_Password_With_A_Lot_Of_Characters
```

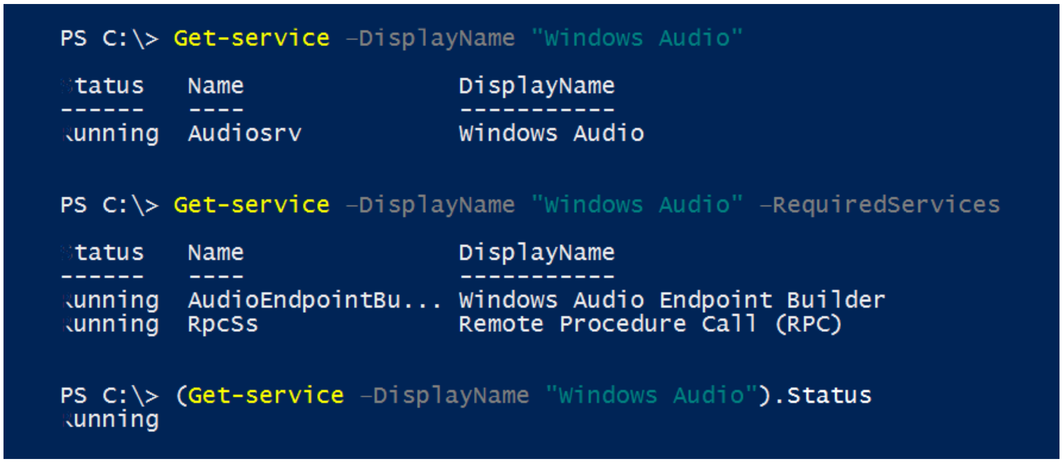


```
<?xml version="1.0"?>

<ScriptAnswers>
  <AFD name="wBzAHcAbwByAGQAXwBXAGkAdABoAF8AQQBfAEwAbwB0AF8ATwB"></AFD>
</ScriptAnswers>
```

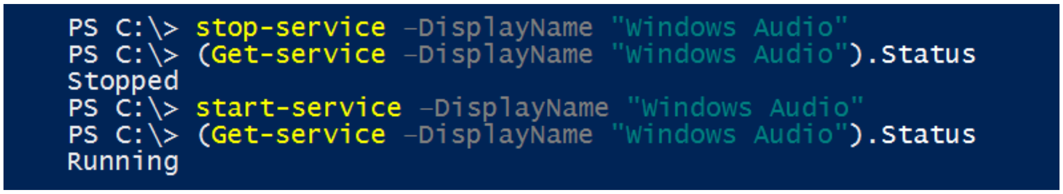


```
PS C:\> powershell.exe -file "c:\temp\Example.ps1" "c:\temp\Example.xml" "mAF8AQwBoAGEAcgBhAGMAdABlAHIAcwA="
Encrypt This String
Press Enter to continue...:
```

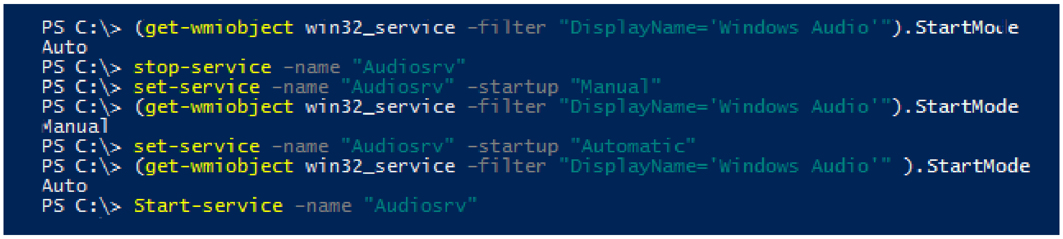
Chapter 5: Interacting with Services, Processes, Profiles, and Logged on Users



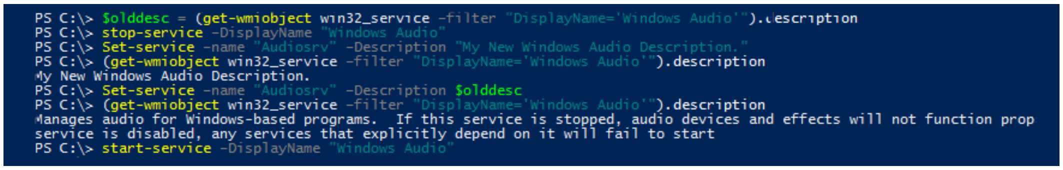
```
PS C:\> Get-service -DisplayName "Windows Audio"

Status   Name               DisplayName
------   ----               -----------
Running  Audiosrv           Windows Audio


PS C:\> Get-service -DisplayName "Windows Audio" -RequiredServices

Status   Name               DisplayName
------   ----               -----------
Running  AudioEndpointBu... Windows Audio Endpoint Builder
Running  RpcSs              Remote Procedure Call (RPC)


PS C:\> (Get-service -DisplayName "Windows Audio").Status
Running
```



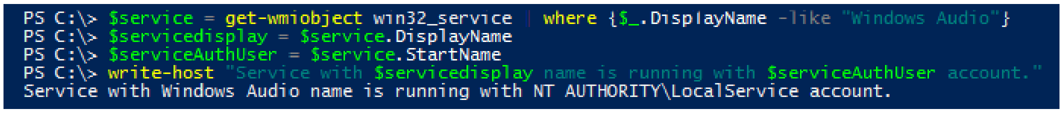
```
PS C:\> stop-service -DisplayName "Windows Audio"
PS C:\> (Get-service -DisplayName "Windows Audio").Status
Stopped
PS C:\> start-service -DisplayName "Windows Audio"
PS C:\> (Get-service -DisplayName "Windows Audio").Status
Running
```



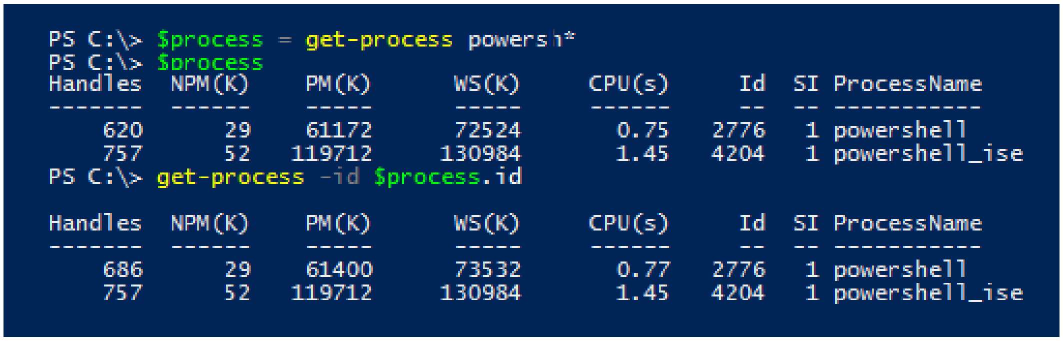
```
PS C:\> (get-wmiobject win32_service -filter "DisplayName='Windows Audio'").StartMode
Auto
PS C:\> stop-service -name "Audiosrv"
PS C:\> set-service -name "Audiosrv" -startup "Manual"
PS C:\> (get-wmiobject win32_service -filter "DisplayName='Windows Audio'").StartMode
Manual
PS C:\> set-service -name "Audiosrv" -startup "Automatic"
PS C:\> (get-wmiobject win32_service -filter "DisplayName='Windows Audio'" ).StartMode
Auto
PS C:\> Start-service -name "Audiosrv"
```



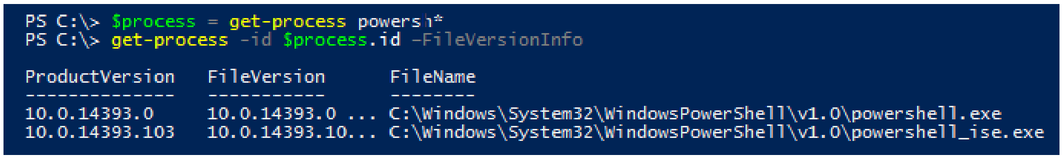
```
PS C:\> $olddesc = (get-wmiobject win32_service -filter "DisplayName='Windows Audio'").description
PS C:\> stop-service -DisplayName "Windows Audio"
PS C:\> Set-service -name "Audiosrv" -Description "My New Windows Audio Description."
PS C:\> (get-wmiobject win32_service -filter "DisplayName='Windows Audio'").description
My New Windows Audio Description.
PS C:\> Set-service -name "Audiosrv" -Description $olddesc
PS C:\> (get-wmiobject win32_service -filter "DisplayName='Windows Audio'").description
Manages audio for Windows-based programs.  If this service is stopped, audio devices and effects will not function prop
service is disabled, any services that explicitly depend on it will fail to start
PS C:\> start-service -DisplayName "Windows Audio"
```



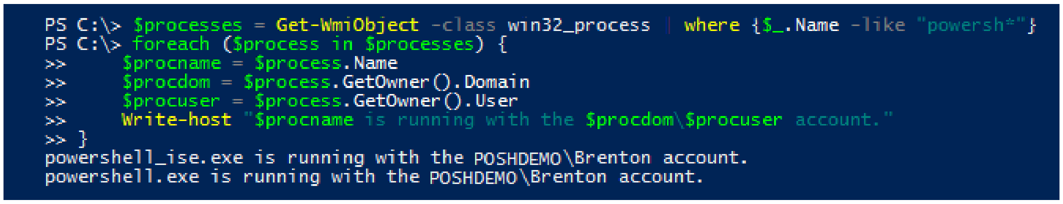
```
PS C:\> $service = get-wmiobject win32_service | where {$_.DisplayName -like "Windows Audio"}
PS C:\> $servicedisplay = $service.DisplayName
PS C:\> $serviceAuthUser = $service.StartName
PS C:\> write-host "Service with $servicedisplay name is running with $serviceAuthUser account."
Service with Windows Audio name is running with NT AUTHORITY\LocalService account.
```



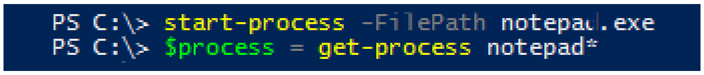
```
PS C:\> $process = get-process powersh*
PS C:\> $process
Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    620      29    61172      72524       0.75   2776   1 powershell
    757      52   119712     130984       1.45   4204   1 powershell_ise
PS C:\> get-process -id $process.id

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    686      29    61400      73532       0.77   2776   1 powershell
    757      52   119712     130984       1.45   4204   1 powershell_ise
```



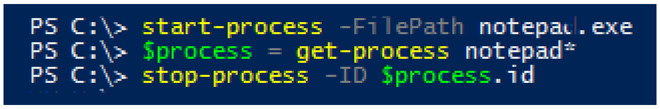
```
PS C:\> $process = get-process powersh*
PS C:\> get-process -id $process.id -FileVersionInfo

ProductVersion   FileVersion      FileName
--------------   -----------      --------
10.0.14393.0     10.0.14393.0 ... C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
10.0.14393.103   10.0.14393.10... C:\Windows\System32\WindowsPowerShell\v1.0\powershell_ise.exe
```



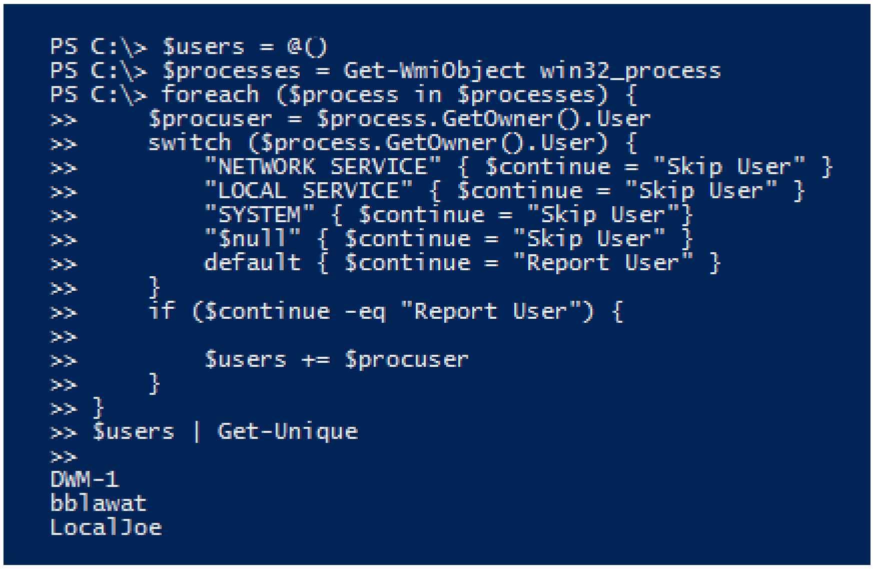
```
PS C:\> $processes = Get-WmiObject -class win32_process | where {$_.Name -like "powersh*"}
PS C:\> foreach ($process in $processes) {
>>     $procname = $process.Name
>>     $procdom = $process.GetOwner().Domain
>>     $procuser = $process.GetOwner().User
>>     Write-host "$procname is running with the $procdom\$procuser account."
>> }
powershell_ise.exe is running with the POSHDEMO\Brenton account.
powershell.exe is running with the POSHDEMO\Brenton account.
```



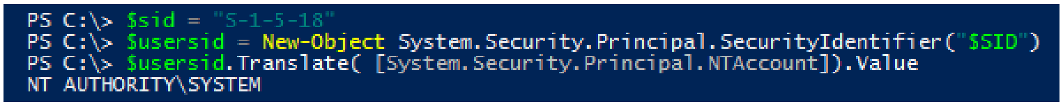
```
PS C:\> start-process -FilePath notepad.exe
PS C:\> $process = get-process notepad*
```



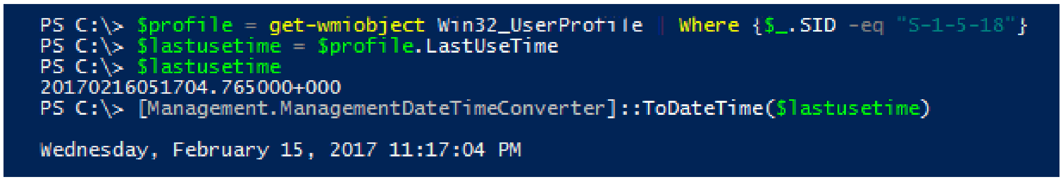
```
PS C:\> start-process -FilePath notepad.exe
PS C:\> $process = get-process notepad*
PS C:\> stop-process -ID $process.id
```



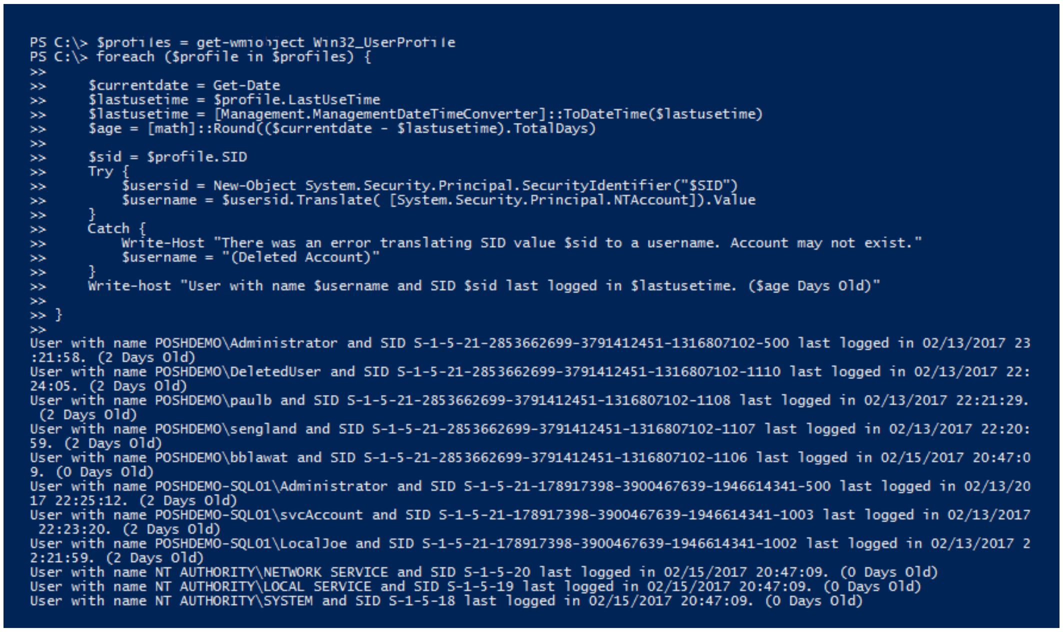
```
PS C:\> $users = @()
PS C:\> $processes = Get-WmiObject win32_process
PS C:\> foreach ($process in $processes) {
>>     $procuser = $process.GetOwner().User
>>     switch ($process.GetOwner().User) {
>>         "NETWORK SERVICE" { $continue = "Skip User" }
>>         "LOCAL SERVICE" { $continue = "Skip User" }
>>         "SYSTEM" { $continue = "Skip User"}
>>         "$null" { $continue = "Skip User" }
>>         default { $continue = "Report User" }
>>     }
>>     if ($continue -eq "Report User") {
>>
>>         $users += $procuser
>>     }
>> }
>> $users | Get-Unique
>>
DWM-1
bblawat
LocalJoe
```



```
PS C:\> $sid = "S-1-5-18"
PS C:\> $usersid = New-Object System.Security.Principal.SecurityIdentifier("$SID")
PS C:\> $usersid.Translate( [System.Security.Principal.NTAccount]).Value
NT AUTHORITY\SYSTEM
```

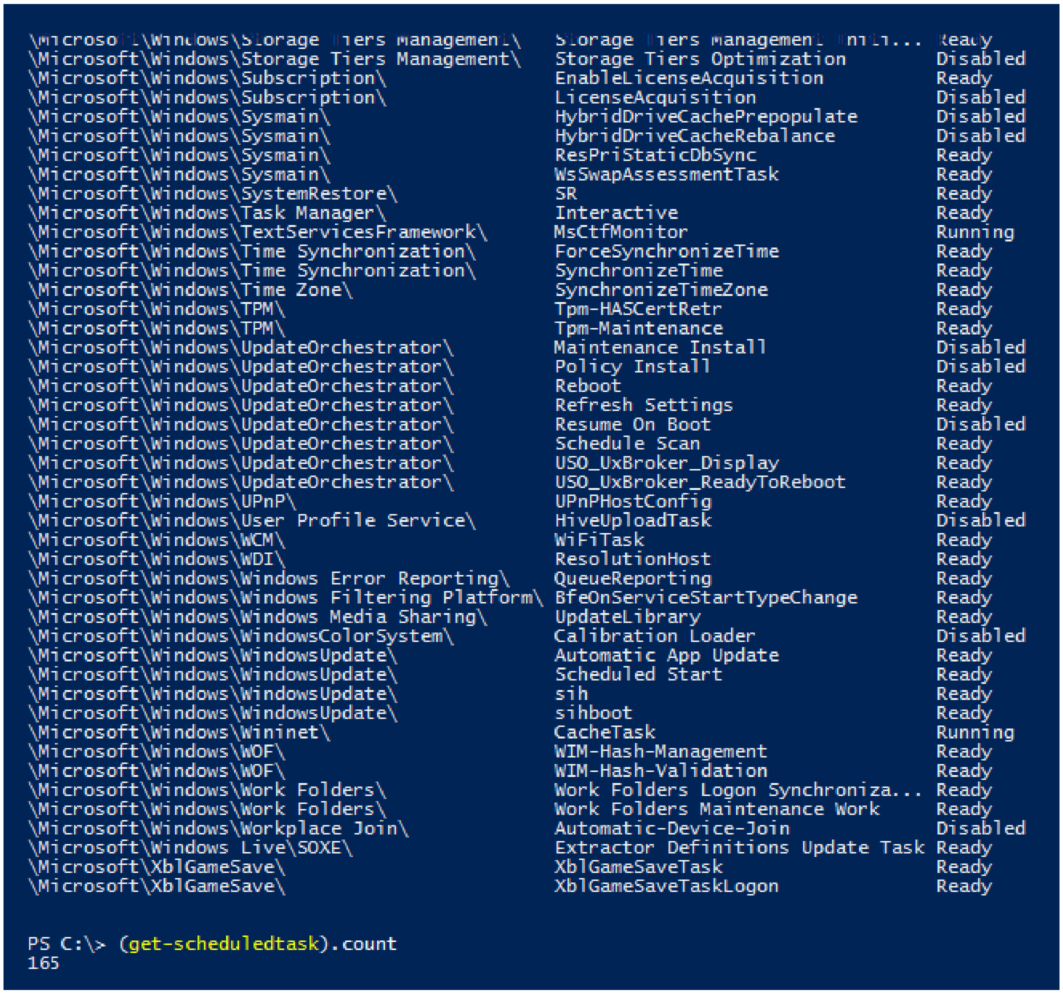


```
PS C:\> $profile = get-wmiobject Win32_UserProfile | Where {$_.SID -eq "S-1-5-18"}
PS C:\> $lastusetime = $profile.LastUseTime
PS C:\> $lastusetime
20170216051704.765000+000
PS C:\> [Management.ManagementDateTimeConverter]::ToDateTime($lastusetime)

Wednesday, February 15, 2017 11:17:04 PM
```

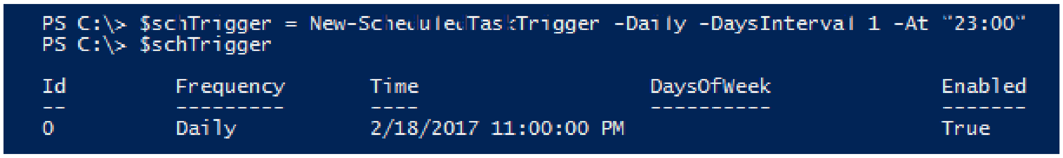


```
PS C:\> $profiles = get-wmiobject Win32_UserProfile
PS C:\> foreach ($profile in $profiles) {
>>
>>     $currentdate = Get-Date
>>     $lastusetime = $profile.LastUseTime
>>     $lastusetime = [Management.ManagementDateTimeConverter]::ToDateTime($lastusetime)
>>     $age = [math]::Round(($currentdate - $lastusetime).TotalDays)
>>
>>     $sid = $profile.SID
>>     Try {
>>         $usersid = New-Object System.Security.Principal.SecurityIdentifier("$SID")
>>         $username = $usersid.Translate( [System.Security.Principal.NTAccount]).Value
>>     }
>>     Catch {
>>         Write-Host "There was an error translating SID value $sid to a username. Account may not exist."
>>         $username = "(Deleted Account)"
>>     }
>>     Write-host "User with name $username and SID $sid last logged in $lastusetime. ($age Days Old)"
>>
>> }
>>
User with name POSHDEMO\Administrator and SID S-1-5-21-2853662699-3791412451-1316807102-500 last logged in 02/13/2017 23
:21:58. (2 Days Old)
User with name POSHDEMO\DeletedUser and SID S-1-5-21-2853662699-3791412451-1316807102-1110 last logged in 02/13/2017 22:
24:05. (2 Days Old)
User with name POSHDEMO\paulb and SID S-1-5-21-2853662699-3791412451-1316807102-1108 last logged in 02/13/2017 22:21:29.
 (2 Days Old)
User with name POSHDEMO\sengland and SID S-1-5-21-2853662699-3791412451-1316807102-1107 last logged in 02/13/2017 22:20:
59. (2 Days Old)
User with name POSHDEMO\bblawat and SID S-1-5-21-2853662699-3791412451-1316807102-1106 last logged in 02/15/2017 20:47:0
9. (0 Days Old)
User with name POSHDEMO-SQL01\Administrator and SID S-1-5-21-178917398-3900467639-1946614341-500 last logged in 02/13/20
17 22:25:12. (2 Days Old)
User with name POSHDEMO-SQL01\svcAccount and SID S-1-5-21-178917398-3900467639-1946614341-1003 last logged in 02/13/2017
 22:23:20. (2 Days Old)
User with name POSHDEMO-SQL01\LocalJoe and SID S-1-5-21-178917398-3900467639-1946614341-1002 last logged in 02/13/2017 2
2:21:59. (2 Days Old)
User with name NT AUTHORITY\NETWORK SERVICE and SID S-1-5-20 last logged in 02/15/2017 20:47:09. (0 Days Old)
User with name NT AUTHORITY\LOCAL SERVICE and SID S-1-5-19 last logged in 02/15/2017 20:47:09. (0 Days Old)
User with name NT AUTHORITY\SYSTEM and SID S-1-5-18 last logged in 02/15/2017 20:47:09. (0 Days Old)
```

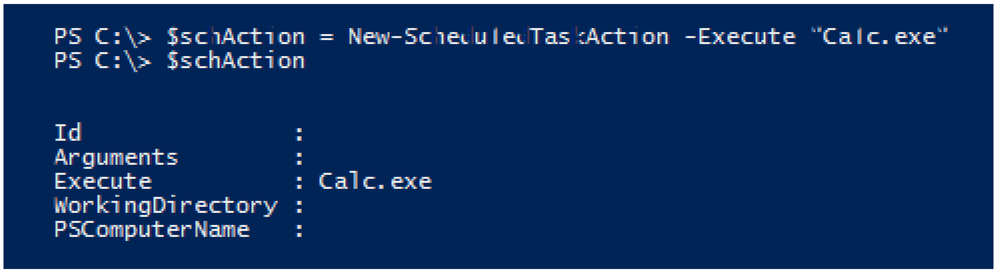
Chapter 6: Evaluating Scheduled Tasks



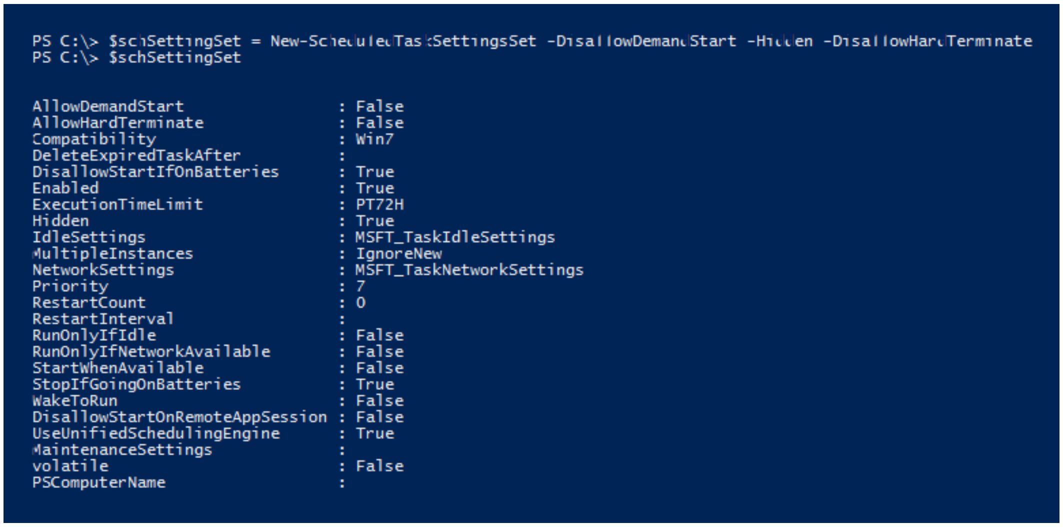
```
\Microsoft\Windows\Storage Tiers Management\   Storage Tiers Management Initi... Ready
\Microsoft\Windows\Storage Tiers Management\   Storage Tiers Optimization        Disabled
\Microsoft\Windows\Subscription\               EnableLicenseAcquisition          Ready
\Microsoft\Windows\Subscription\               LicenseAcquisition                Disabled
\Microsoft\Windows\Sysmain\                    HybridDriveCachePrepopulate       Disabled
\Microsoft\Windows\Sysmain\                    HybridDriveCacheRebalance         Disabled
\Microsoft\Windows\Sysmain\                    ResPriStaticDbSync                Ready
\Microsoft\Windows\Sysmain\                    WsSwapAssessmentTask              Ready
\Microsoft\Windows\SystemRestore\              SR                                Ready
\Microsoft\Windows\Task Manager\               Interactive                       Ready
\Microsoft\Windows\TextServicesFramework\      MsCtfMonitor                      Running
\Microsoft\Windows\Time Synchronization\       ForceSynchronizeTime              Ready
\Microsoft\Windows\Time Synchronization\       SynchronizeTime                   Ready
\Microsoft\Windows\Time Zone\                  SynchronizeTimeZone               Ready
\Microsoft\Windows\TPM\                        Tpm-HASCertRetr                   Ready
\Microsoft\Windows\TPM\                        Tpm-Maintenance                   Ready
\Microsoft\Windows\UpdateOrchestrator\         Maintenance Install               Disabled
\Microsoft\Windows\UpdateOrchestrator\         Policy Install                    Disabled
\Microsoft\Windows\UpdateOrchestrator\         Reboot                            Ready
\Microsoft\Windows\UpdateOrchestrator\         Refresh Settings                  Ready
\Microsoft\Windows\UpdateOrchestrator\         Resume On Boot                    Disabled
\Microsoft\Windows\UpdateOrchestrator\         Schedule Scan                     Ready
\Microsoft\Windows\UpdateOrchestrator\         USO_UxBroker_Display              Ready
\Microsoft\Windows\UpdateOrchestrator\         USO_UxBroker_ReadyToReboot        Ready
\Microsoft\Windows\UPnP\                       UPnPHostConfig                    Ready
\Microsoft\Windows\User Profile Service\       HiveUploadTask                    Disabled
\Microsoft\Windows\WCM\                        WiFiTask                          Ready
\Microsoft\Windows\WDI\                        ResolutionHost                    Ready
\Microsoft\Windows\Windows Error Reporting\    QueueReporting                    Ready
\Microsoft\Windows\Windows Filtering Platform\ BfeOnServiceStartTypeChange       Ready
\Microsoft\Windows\Windows Media Sharing\      UpdateLibrary                     Ready
\Microsoft\Windows\WindowsColorSystem\         Calibration Loader                Disabled
\Microsoft\Windows\WindowsUpdate\              Automatic App Update              Ready
\Microsoft\Windows\WindowsUpdate\              Scheduled Start                   Ready
\Microsoft\Windows\WindowsUpdate\              sih                               Ready
\Microsoft\Windows\WindowsUpdate\              sihboot                           Ready
\Microsoft\Windows\Wininet\                    CacheTask                         Running
\Microsoft\Windows\WOF\                        WIM-Hash-Management               Ready
\Microsoft\Windows\WOF\                        WIM-Hash-Validation               Ready
\Microsoft\Windows\Work Folders\               Work Folders Logon Synchroniza... Ready
\Microsoft\Windows\Work Folders\               Work Folders Maintenance Work     Ready
\Microsoft\Windows\Workplace Join\             Automatic-Device-Join             Disabled
\Microsoft\Windows Live\SOXE\                  Extractor Definitions Update Task Ready
\Microsoft\XblGameSave\                        XblGameSaveTask                   Ready
\Microsoft\XblGameSave\                        XblGameSaveTaskLogon              Ready


PS C:\> (get-scheduledtask).count
165
```



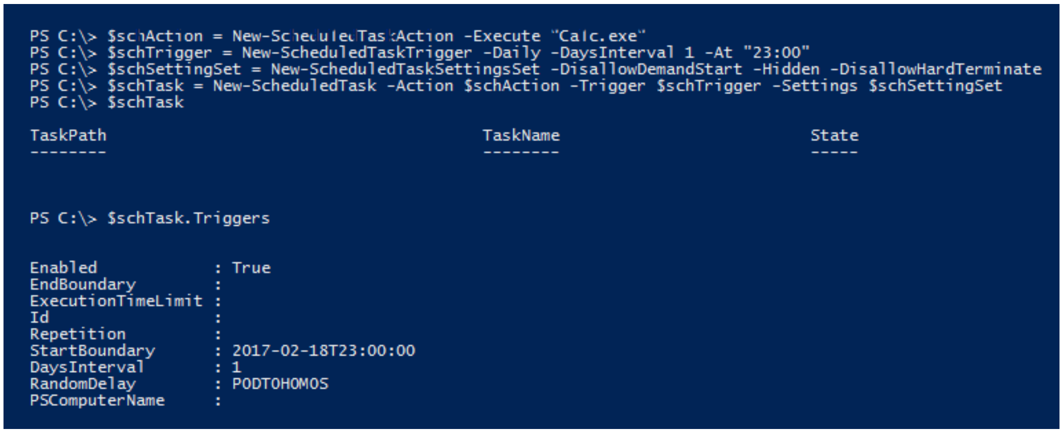
```
PS C:\> $schTrigger = New-ScheduledTaskTrigger -Daily -DaysInterval 1 -At "23:00"
PS C:\> $schTrigger

Id         Frequency       Time                   DaysOfWeek              Enabled
--         ---------       ----                   ----------              -------
0          Daily           2/18/2017 11:00:00 PM                          True
```



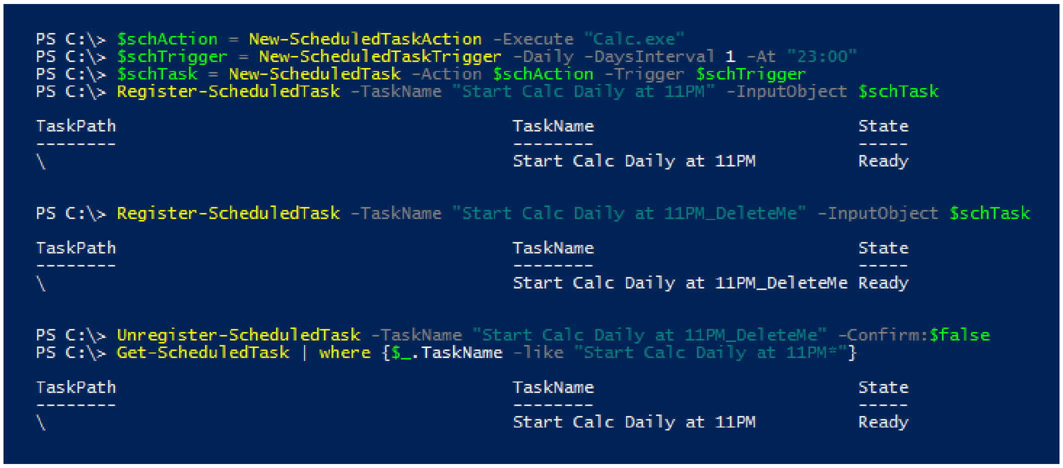
```
PS C:\> $schAction = New-ScheduledTaskAction -Execute “Calc.exe”
PS C:\> $schAction


Id               :
Arguments        :
Execute          : Calc.exe
WorkingDirectory :
PSComputerName   :
```



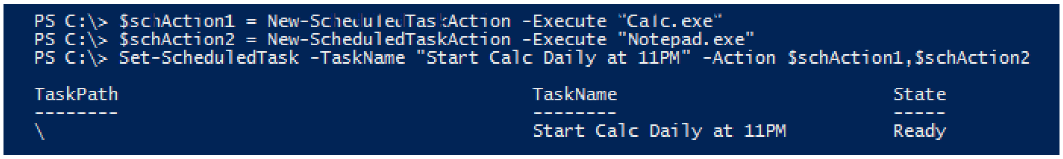
```
PS C:\> $schSettingSet = New-ScheduledTaskSettingsSet -DisallowDemandStart -Hidden -DisallowHardTerminate
PS C:\> $schSettingSet


AllowDemandStart                : False
AllowHardTerminate              : False
Compatibility                   : Win7
DeleteExpiredTaskAfter          :
DisallowStartIfOnBatteries      : True
Enabled                         : True
ExecutionTimeLimit              : PT72H
Hidden                          : True
IdleSettings                    : MSFT_TaskIdleSettings
MultipleInstances               : IgnoreNew
NetworkSettings                 : MSFT_TaskNetworkSettings
Priority                        : 7
RestartCount                    : 0
RestartInterval                 :
RunOnlyIfIdle                   : False
RunOnlyIfNetworkAvailable       : False
StartWhenAvailable              : False
StopIfGoingOnBatteries          : True
WakeToRun                       : False
DisallowStartOnRemoteAppSession : False
UseUnifiedSchedulingEngine      : True
MaintenanceSettings             :
volatile                        : False
PSComputerName                  :
```



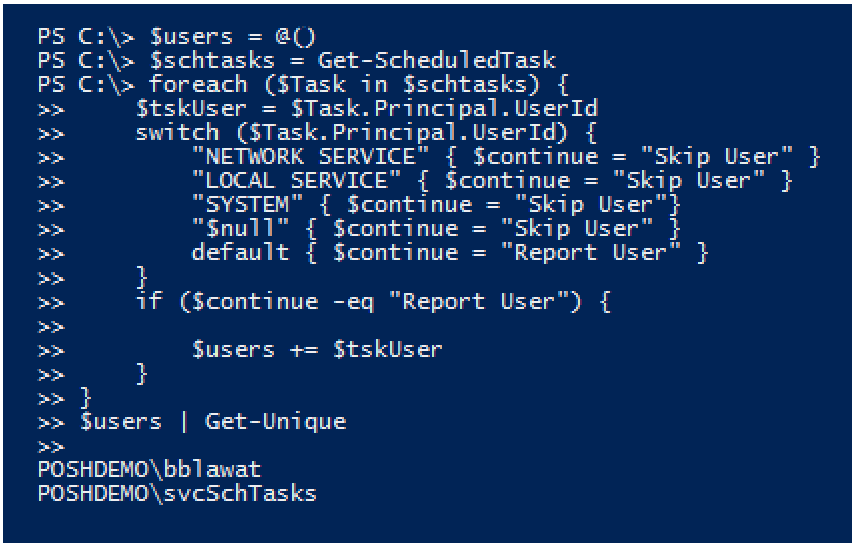
```
PS C:\> $schAction = New-ScheduledTaskAction -Execute “Calc.exe”
PS C:\> $schTrigger = New-ScheduledTaskTrigger -Daily -DaysInterval 1 -At "23:00"
PS C:\> $schSettingSet = New-ScheduledTaskSettingsSet -DisallowDemandStart -Hidden -DisallowHardTerminate
PS C:\> $schTask = New-ScheduledTask -Action $schAction -Trigger $schTrigger -Settings $schSettingSet
PS C:\> $schTask

TaskPath                                       TaskName                          State
--------                                       --------                          -----


PS C:\> $schTask.Triggers


Enabled            : True
EndBoundary        :
ExecutionTimeLimit :
Id                 :
Repetition         :
StartBoundary      : 2017-02-18T23:00:00
DaysInterval       : 1
RandomDelay        : P0DT0H0M0S
PSComputerName     :
```



```
PS C:\> $schAction = New-ScheduledTaskAction -Execute "Calc.exe"
PS C:\> $schTrigger = New-ScheduledTaskTrigger -Daily -DaysInterval 1 -At "23:00"
PS C:\> $schTask = New-ScheduledTask -Action $schAction -Trigger $schTrigger
PS C:\> Register-ScheduledTask -TaskName "Start Calc Daily at 11PM" -InputObject $schTask

TaskPath                                       TaskName                          State
--------                                       --------                          -----
\                                              Start Calc Daily at 11PM          Ready


PS C:\> Register-ScheduledTask -TaskName "Start Calc Daily at 11PM_DeleteMe" -InputObject $schTask

TaskPath                                       TaskName                          State
--------                                       --------                          -----
\                                              Start Calc Daily at 11PM_DeleteMe Ready


PS C:\> Unregister-ScheduledTask -TaskName "Start Calc Daily at 11PM_DeleteMe" -Confirm:$false
PS C:\> Get-ScheduledTask | where {$_.TaskName -like "Start Calc Daily at 11PM*"}

TaskPath                                       TaskName                          State
--------                                       --------                          -----
\                                              Start Calc Daily at 11PM          Ready
```

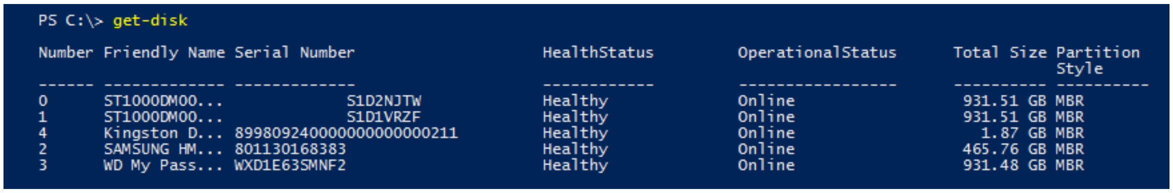


```
PS C:\> $schAction1 = New-ScheduledTaskAction -Execute "Calc.exe"
PS C:\> $schAction2 = New-ScheduledTaskAction -Execute "Notepad.exe"
PS C:\> Set-ScheduledTask -TaskName "Start Calc Daily at 11PM" -Action $schAction1,$schAction2

TaskPath                                       TaskName                          State
--------                                       --------                          -----
\                                              Start Calc Daily at 11PM          Ready
```

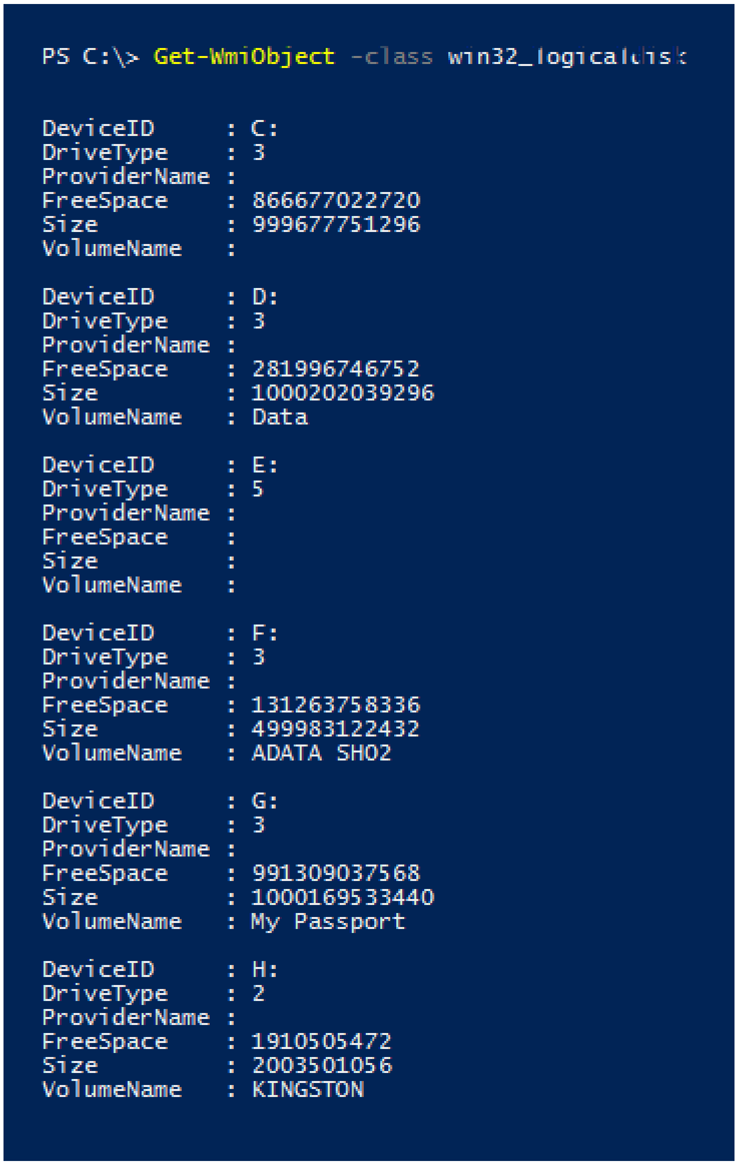


```
PS C:\> $users = @()
PS C:\> $schtasks = Get-ScheduledTask
PS C:\> foreach ($Task in $schtasks) {
>>     $tskUser = $Task.Principal.UserId
>>     switch ($Task.Principal.UserId) {
>>         "NETWORK SERVICE" { $continue = "Skip User" }
>>         "LOCAL SERVICE" { $continue = "Skip User" }
>>         "SYSTEM" { $continue = "Skip User"}
>>         "$null" { $continue = "Skip User" }
>>         default { $continue = "Report User" }
>>     }
>>     if ($continue -eq "Report User") {
>>
>>         $users += $tskUser
>>     }
>> }
>> $users | Get-Unique
>>
POSHDEMO\bblawat
POSHDEMO\svcSchTasks
```

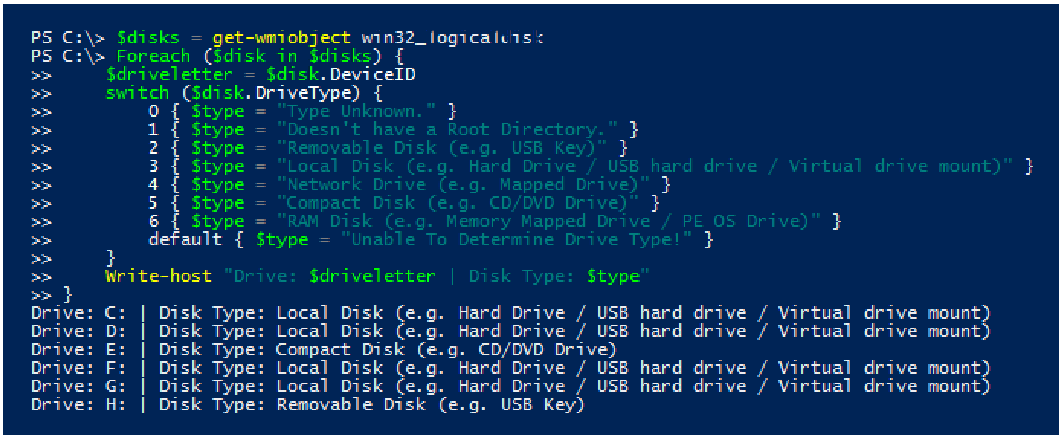
Chapter 7: Determining Disk Statistics



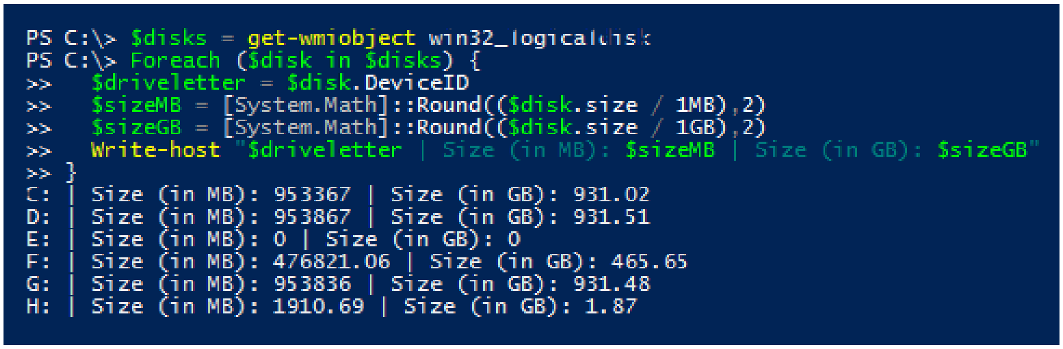
```
PS C:\> get-disk

Number Friendly Name Serial Number                    HealthStatus         OperationalStatus      Total Size Partition
                                                                                                             Style
------ ------------- -------------                    ------------         -----------------      ---------- ----------
0      ST1000DM00...             S1D2NJTW             Healthy              Online                  931.51 GB MBR
1      ST1000DM00...             S1D1VRZF             Healthy              Online                  931.51 GB MBR
4      Kingston D... 899809240000000000000211         Healthy              Online                    1.87 GB MBR
2      SAMSUNG HM... 801130168383                     Healthy              Online                  465.76 GB MBR
3      WD My Pass... WXD1E63SMNF2                     Healthy              Online                  931.48 GB MBR
```



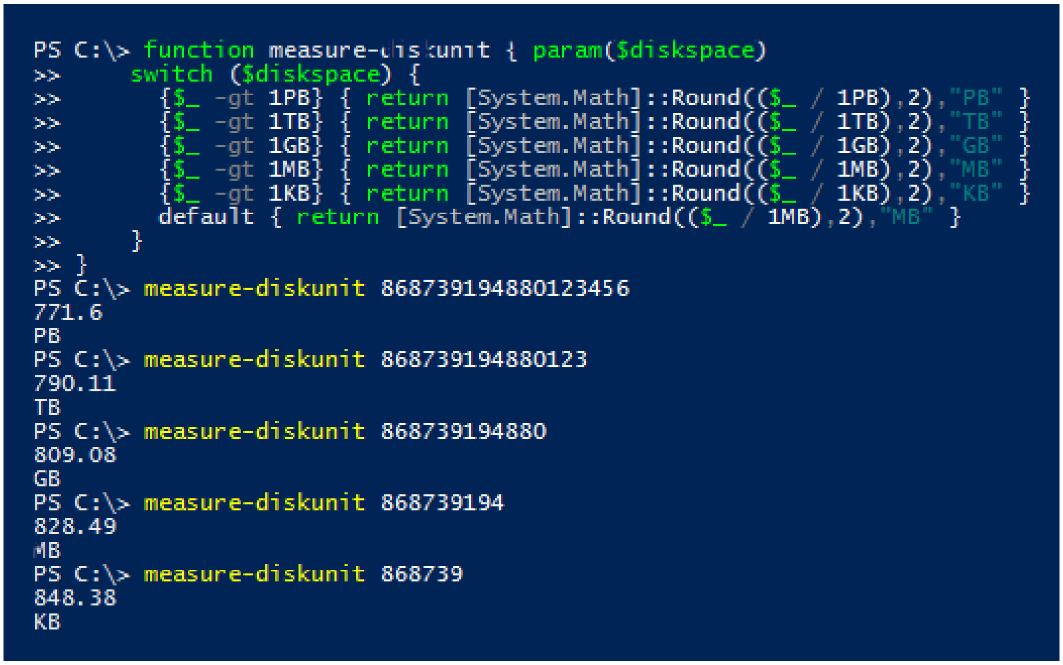
```
PS C:\> Get-WmiObject -class win32_logicaldisk


DeviceID     : C:
DriveType    : 3
ProviderName :
FreeSpace    : 866677022720
Size         : 999677751296
VolumeName   :

DeviceID     : D:
DriveType    : 3
ProviderName :
FreeSpace    : 281996746752
Size         : 1000202039296
VolumeName   : Data

DeviceID     : E:
DriveType    : 5
ProviderName :
FreeSpace    :
Size         :
VolumeName   :

DeviceID     : F:
DriveType    : 3
ProviderName :
FreeSpace    : 131263758336
Size         : 499983122432
VolumeName   : ADATA SH02

DeviceID     : G:
DriveType    : 3
ProviderName :
FreeSpace    : 991309037568
Size         : 1000169533440
VolumeName   : My Passport

DeviceID     : H:
DriveType    : 2
ProviderName :
FreeSpace    : 1910505472
Size         : 2003501056
VolumeName   : KINGSTON
```



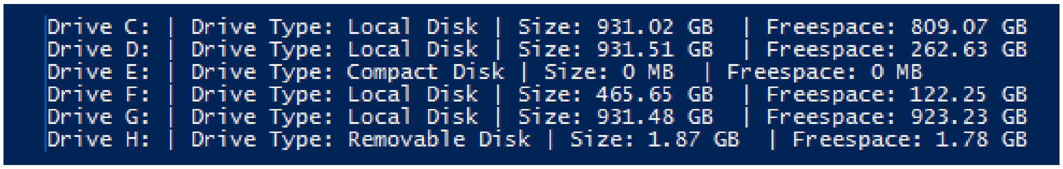
```
PS C:\> $disks = get-wmiobject win32_logicaldisk
PS C:\> Foreach ($disk in $disks) {
>>     $driveletter = $disk.DeviceID
>>     switch ($disk.DriveType) {
>>         0 { $type = "Type Unknown." }
>>         1 { $type = "Doesn't have a Root Directory." }
>>         2 { $type = "Removable Disk (e.g. USB Key)" }
>>         3 { $type = "Local Disk (e.g. Hard Drive / USB hard drive / Virtual drive mount)" }
>>         4 { $type = "Network Drive (e.g. Mapped Drive)" }
>>         5 { $type = "Compact Disk (e.g. CD/DVD Drive)" }
>>         6 { $type = "RAM Disk (e.g. Memory Mapped Drive / PE OS Drive)" }
>>         default { $type = "Unable To Determine Drive Type!" }
>>     }
>>     Write-host "Drive: $driveletter | Disk Type: $type"
>> }
Drive: C: | Disk Type: Local Disk (e.g. Hard Drive / USB hard drive / Virtual drive mount)
Drive: D: | Disk Type: Local Disk (e.g. Hard Drive / USB hard drive / Virtual drive mount)
Drive: E: | Disk Type: Compact Disk (e.g. CD/DVD Drive)
Drive: F: | Disk Type: Local Disk (e.g. Hard Drive / USB hard drive / Virtual drive mount)
Drive: G: | Disk Type: Local Disk (e.g. Hard Drive / USB hard drive / Virtual drive mount)
Drive: H: | Disk Type: Removable Disk (e.g. USB Key)
```



```
PS C:\> $disks = get-wmiobject win32_logicaldisk
PS C:\> Foreach ($disk in $disks) {
>>   $driveletter = $disk.DeviceID
>>   $sizeMB = [System.Math]::Round(($disk.size / 1MB),2)
>>   $sizeGB = [System.Math]::Round(($disk.size / 1GB),2)
>>   Write-host "$driveletter | Size (in MB): $sizeMB | Size (in GB): $sizeGB"
>> }
C: | Size (in MB): 953367 | Size (in GB): 931.02
D: | Size (in MB): 953867 | Size (in GB): 931.51
E: | Size (in MB): 0 | Size (in GB): 0
F: | Size (in MB): 476821.06 | Size (in GB): 465.65
G: | Size (in MB): 953836 | Size (in GB): 931.48
H: | Size (in MB): 1910.69 | Size (in GB): 1.87
```

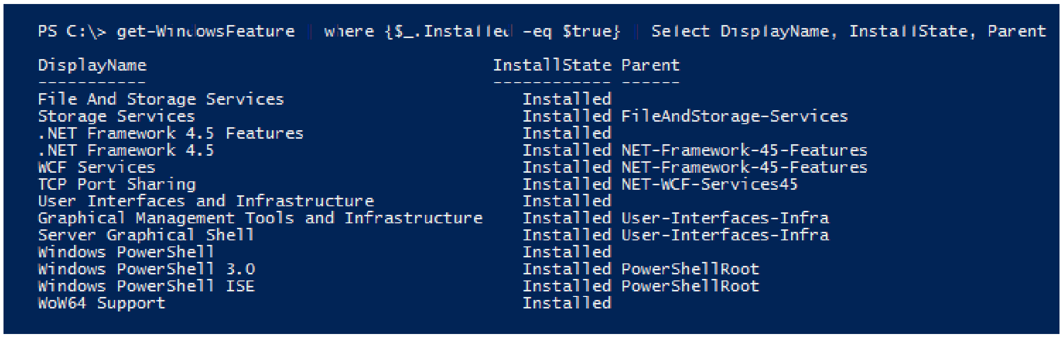


```
PS C:\> function measure-diskunit { param($diskspace)
>>     switch ($diskspace) {
>>       {$_ -gt 1PB} { return [System.Math]::Round(($_ / 1PB),2),"PB" }
>>       {$_ -gt 1TB} { return [System.Math]::Round(($_ / 1TB),2),"TB" }
>>       {$_ -gt 1GB} { return [System.Math]::Round(($_ / 1GB),2),"GB" }
>>       {$_ -gt 1MB} { return [System.Math]::Round(($_ / 1MB),2),"MB" }
>>       {$_ -gt 1KB} { return [System.Math]::Round(($_ / 1KB),2),"KB" }
>>       default { return [System.Math]::Round(($_ / 1MB),2),"MB" }
>>     }
>> }
PS C:\> measure-diskunit 868739194880123456
771.6
PB
PS C:\> measure-diskunit 868739194880123
790.11
TB
PS C:\> measure-diskunit 868739194880
809.08
GB
PS C:\> measure-diskunit 868739194
828.49
MB
PS C:\> measure-diskunit 868739
848.38
KB
```

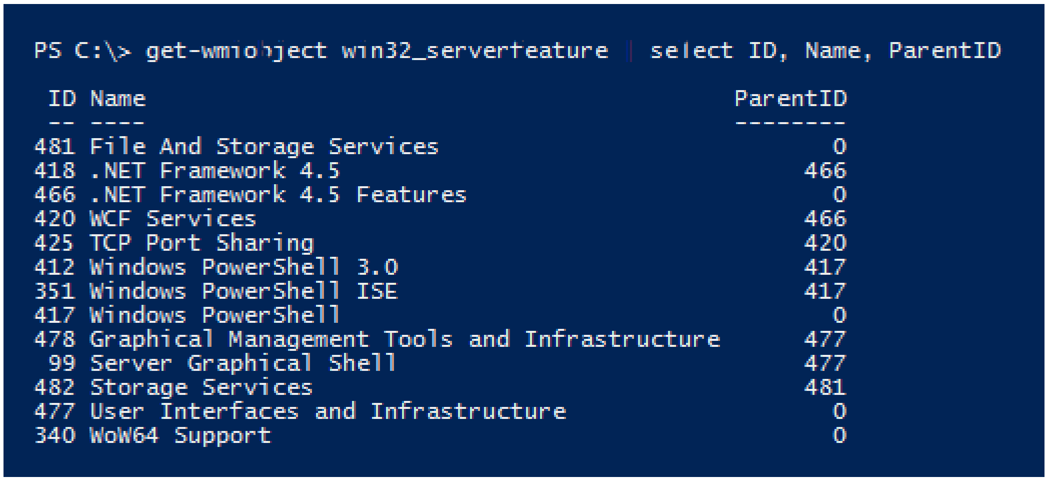


```
Drive C: | Drive Type: Local Disk | Size: 931.02 GB  | Freespace: 809.07 GB
Drive D: | Drive Type: Local Disk | Size: 931.51 GB  | Freespace: 262.63 GB
Drive E: | Drive Type: Compact Disk | Size: 0 MB  | Freespace: 0 MB
Drive F: | Drive Type: Local Disk | Size: 465.65 GB  | Freespace: 122.25 GB
Drive G: | Drive Type: Local Disk | Size: 931.48 GB  | Freespace: 923.23 GB
Drive H: | Drive Type: Removable Disk | Size: 1.87 GB  | Freespace: 1.78 GB
```

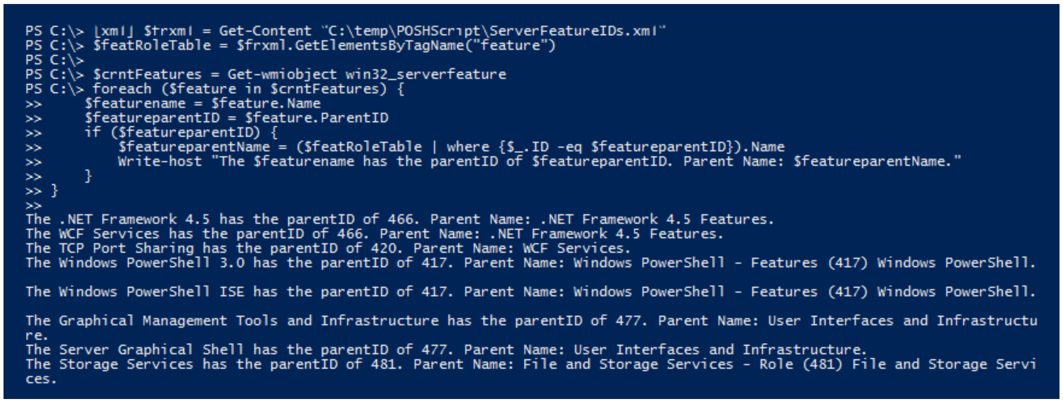
Chapter 8: Windows Features and Installed Software Detection



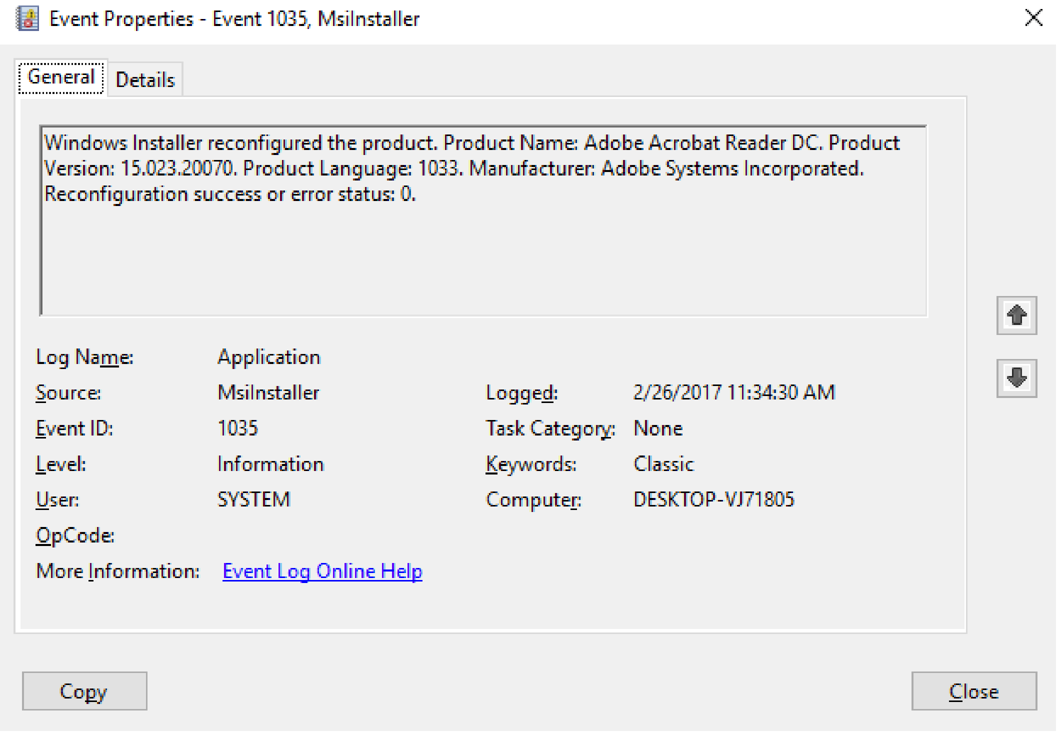
```
PS C:\> get-WindowsFeature | where {$_.Installed -eq $true} | Select DisplayName, InstallState, Parent

DisplayName                                  InstallState Parent
-----------                                  ------------ ------
File And Storage Services                       Installed
Storage Services                                Installed FileAndStorage-Services
.NET Framework 4.5 Features                     Installed
.NET Framework 4.5                              Installed NET-Framework-45-Features
WCF Services                                    Installed NET-Framework-45-Features
TCP Port Sharing                                Installed NET-WCF-Services45
User Interfaces and Infrastructure              Installed
Graphical Management Tools and Infrastructure   Installed User-Interfaces-Infra
Server Graphical Shell                          Installed User-Interfaces-Infra
Windows PowerShell                              Installed
Windows PowerShell 3.0                          Installed PowerShellRoot
Windows PowerShell ISE                          Installed PowerShellRoot
WoW64 Support                                   Installed
```



```
PS C:\> get-wmiobject win32_serverfeature | select ID, Name, ParentID

 ID Name                                               ParentID
 -- ----                                               --------
481 File And Storage Services                                 0
418 .NET Framework 4.5                                      466
466 .NET Framework 4.5 Features                               0
420 WCF Services                                            466
425 TCP Port Sharing                                        420
412 Windows PowerShell 3.0                                  417
351 Windows PowerShell ISE                                  417
417 Windows PowerShell                                        0
478 Graphical Management Tools and Infrastructure           477
 99 Server Graphical Shell                                  477
482 Storage Services                                        481
477 User Interfaces and Infrastructure                        0
340 WoW64 Support                                             0
```



```
PS C:\> [xml] $frxml = Get-Content "C:\temp\POSHScript\ServerFeatureIDs.xml"
PS C:\> $featRoleTable = $frxml.GetElementsByTagName("feature")
PS C:\>
PS C:\> $crntFeatures = Get-wmiobject win32_serverfeature
PS C:\> foreach ($feature in $crntFeatures) {
>>     $featurename = $feature.Name
>>     $featureparentID = $feature.ParentID
>>     if ($featureparentID) {
>>         $featureparentName = ($featRoleTable | where {$_.ID -eq $featureparentID}).Name
>>         Write-host "The $featurename has the parentID of $featureparentID. Parent Name: $featureparentName."
>>     }
>> }
>>
The .NET Framework 4.5 has the parentID of 466. Parent Name: .NET Framework 4.5 Features.
The WCF Services has the parentID of 466. Parent Name: .NET Framework 4.5 Features.
The TCP Port Sharing has the parentID of 420. Parent Name: WCF Services.
The Windows PowerShell 3.0 has the parentID of 417. Parent Name: Windows PowerShell - Features (417) Windows PowerShell.

The Windows PowerShell ISE has the parentID of 417. Parent Name: Windows PowerShell - Features (417) Windows PowerShell.

The Graphical Management Tools and Infrastructure has the parentID of 477. Parent Name: User Interfaces and Infrastructu
re.
The Server Graphical Shell has the parentID of 477. Parent Name: User Interfaces and Infrastructure.
The Storage Services has the parentID of 481. Parent Name: File and Storage Services - Role (481) File and Storage Servi
ces.
```



Event Properties - Event 1035, MsiInstaller

Windows Installer reconfigured the product. Product Name: Adobe Acrobat Reader DC. Product Version: 15.023.20070. Product Language: 1033. Manufacturer: Adobe Systems Incorporated. Reconfiguration success or error status: 0.

Log Name: Application

Source: MsiInstaller
Logged: 2/26/2017 11:34:30 AM

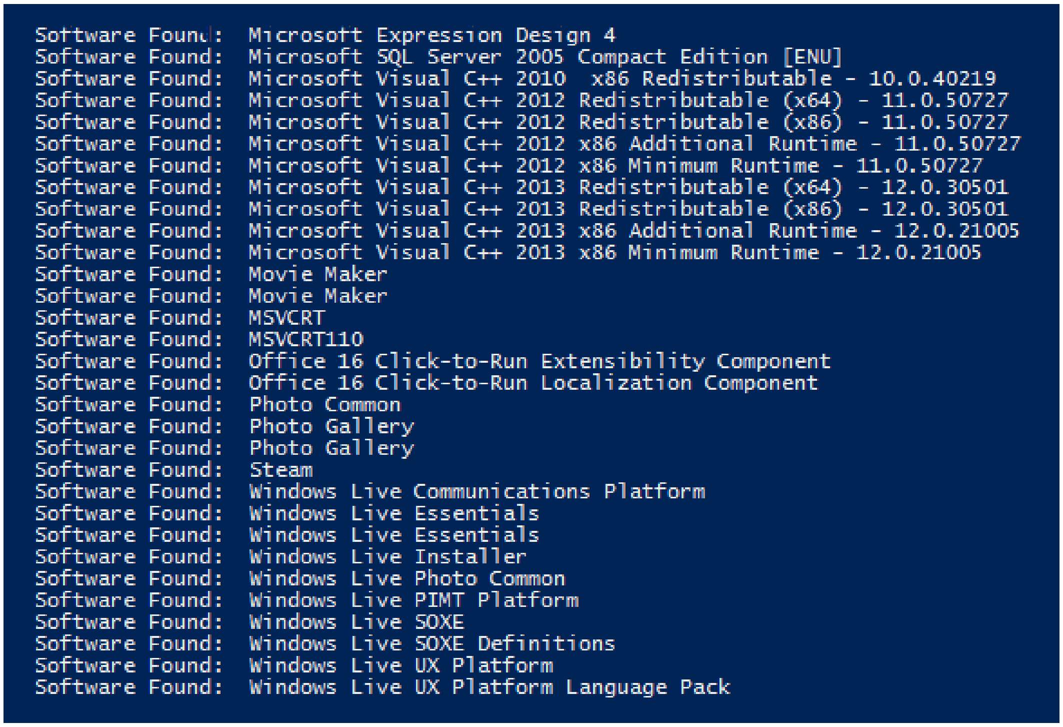
Event ID: 1035
Task Category: None

Level: Information
Keywords: Classic

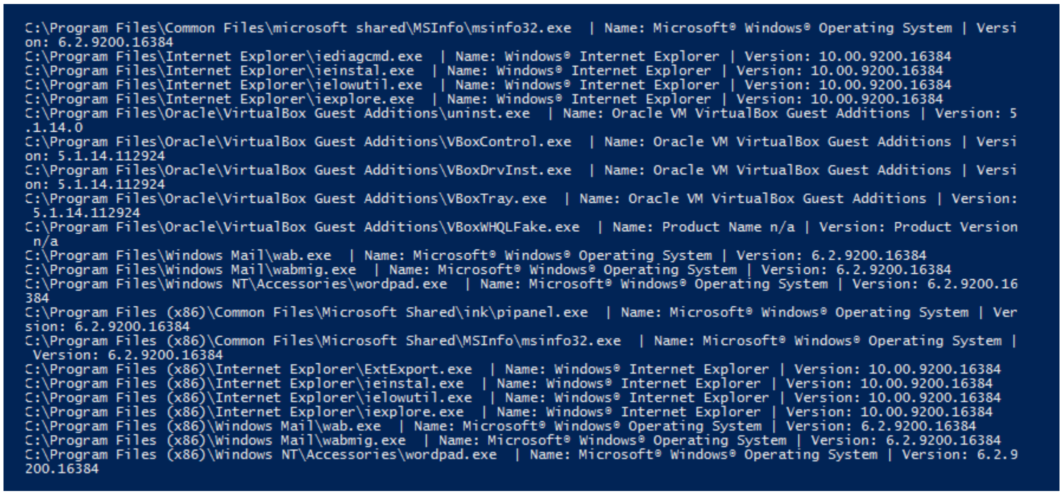
User: SYSTEM
Computer: DESKTOP-VJ71805

OpCode:

More Information: Event Log Online Help

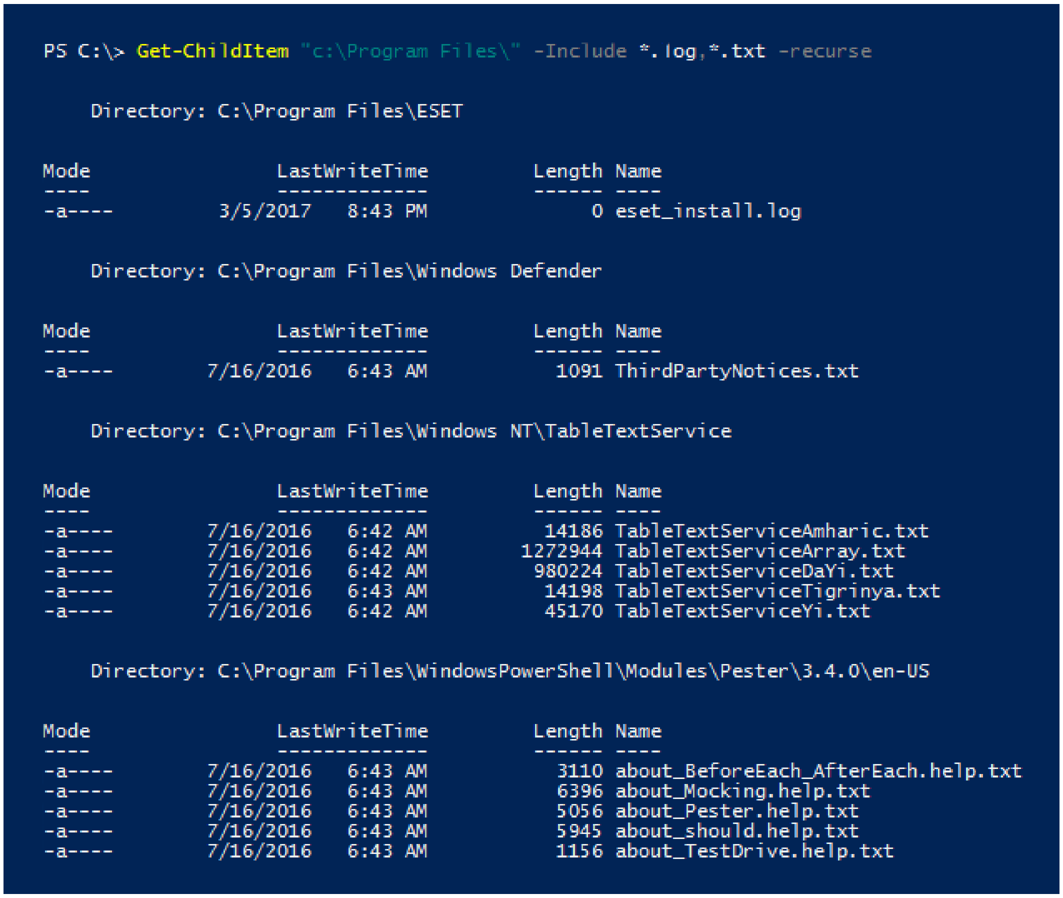


```
Software Found:  Microsoft Expression Design 4
Software Found:  Microsoft SQL Server 2005 Compact Edition [ENU]
Software Found:  Microsoft Visual C++ 2010  x86 Redistributable - 10.0.40219
Software Found:  Microsoft Visual C++ 2012 Redistributable (x64) - 11.0.50727
Software Found:  Microsoft Visual C++ 2012 Redistributable (x86) - 11.0.50727
Software Found:  Microsoft Visual C++ 2012 x86 Additional Runtime - 11.0.50727
Software Found:  Microsoft Visual C++ 2012 x86 Minimum Runtime - 11.0.50727
Software Found:  Microsoft Visual C++ 2013 Redistributable (x64) - 12.0.30501
Software Found:  Microsoft Visual C++ 2013 Redistributable (x86) - 12.0.30501
Software Found:  Microsoft Visual C++ 2013 x86 Additional Runtime - 12.0.21005
Software Found:  Microsoft Visual C++ 2013 x86 Minimum Runtime - 12.0.21005
Software Found:  Movie Maker
Software Found:  Movie Maker
Software Found:  MSVCRT
Software Found:  MSVCRT110
Software Found:  Office 16 Click-to-Run Extensibility Component
Software Found:  Office 16 Click-to-Run Localization Component
Software Found:  Photo Common
Software Found:  Photo Gallery
Software Found:  Photo Gallery
Software Found:  Steam
Software Found:  Windows Live Communications Platform
Software Found:  Windows Live Essentials
Software Found:  Windows Live Essentials
Software Found:  Windows Live Installer
Software Found:  Windows Live Photo Common
Software Found:  Windows Live PIMT Platform
Software Found:  Windows Live SOXE
Software Found:  Windows Live SOXE Definitions
Software Found:  Windows Live UX Platform
Software Found:  Windows Live UX Platform Language Pack
```

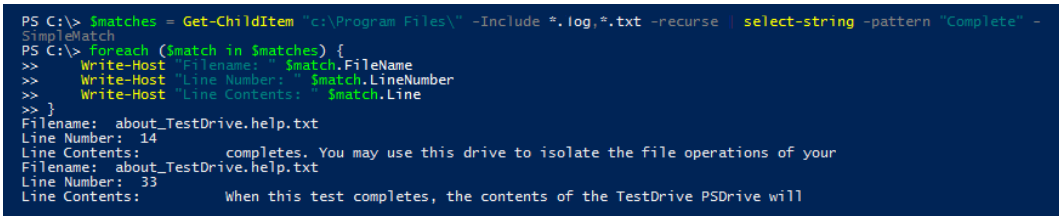


```
C:\Program Files\Common Files\microsoft shared\MSInfo\msinfo32.exe  | Name: Microsoft® Windows® Operating System | Versi
on: 6.2.9200.16384
C:\Program Files\Internet Explorer\iediagcmd.exe  | Name: Windows® Internet Explorer | Version: 10.00.9200.16384
C:\Program Files\Internet Explorer\ieinstal.exe  | Name: Windows® Internet Explorer | Version: 10.00.9200.16384
C:\Program Files\Internet Explorer\ielowutil.exe  | Name: Windows® Internet Explorer | Version: 10.00.9200.16384
C:\Program Files\Internet Explorer\iexplore.exe  | Name: Windows® Internet Explorer | Version: 10.00.9200.16384
C:\Program Files\Oracle\VirtualBox Guest Additions\uninst.exe  | Name: Oracle VM VirtualBox Guest Additions | Version: 5
.1.14.0
C:\Program Files\Oracle\VirtualBox Guest Additions\VBoxControl.exe  | Name: Oracle VM VirtualBox Guest Additions | Versi
on: 5.1.14.112924
C:\Program Files\Oracle\VirtualBox Guest Additions\VBoxDrvInst.exe  | Name: Oracle VM VirtualBox Guest Additions | Versi
on: 5.1.14.112924
C:\Program Files\Oracle\VirtualBox Guest Additions\VBoxTray.exe  | Name: Oracle VM VirtualBox Guest Additions | Version:
 5.1.14.112924
C:\Program Files\Oracle\VirtualBox Guest Additions\VBoxWHQLFake.exe  | Name: Product Name n/a | Version: Product Version
 n/a
C:\Program Files\Windows Mail\wab.exe  | Name: Microsoft® Windows® Operating System | Version: 6.2.9200.16384
C:\Program Files\Windows Mail\wabmig.exe  | Name: Microsoft® Windows® Operating System | Version: 6.2.9200.16384
C:\Program Files\Windows NT\Accessories\wordpad.exe  | Name: Microsoft® Windows® Operating System | Version: 6.2.9200.16
384
C:\Program Files (x86)\Common Files\Microsoft Shared\ink\pipanel.exe  | Name: Microsoft® Windows® Operating System | Ver
sion: 6.2.9200.16384
C:\Program Files (x86)\Common Files\Microsoft Shared\MSInfo\msinfo32.exe  | Name: Microsoft® Windows® Operating System |
 Version: 6.2.9200.16384
C:\Program Files (x86)\Internet Explorer\ExtExport.exe  | Name: Windows® Internet Explorer | Version: 10.00.9200.16384
C:\Program Files (x86)\Internet Explorer\ieinstal.exe  | Name: Windows® Internet Explorer | Version: 10.00.9200.16384
C:\Program Files (x86)\Internet Explorer\ielowutil.exe  | Name: Windows® Internet Explorer | Version: 10.00.9200.16384
C:\Program Files (x86)\Internet Explorer\iexplore.exe  | Name: Windows® Internet Explorer | Version: 10.00.9200.16384
C:\Program Files (x86)\Windows Mail\wab.exe  | Name: Microsoft® Windows® Operating System | Version: 6.2.9200.16384
C:\Program Files (x86)\Windows Mail\wabmig.exe  | Name: Microsoft® Windows® Operating System | Version: 6.2.9200.16384
C:\Program Files (x86)\Windows NT\Accessories\wordpad.exe  | Name: Microsoft® Windows® Operating System | Version: 6.2.9
200.16384
```

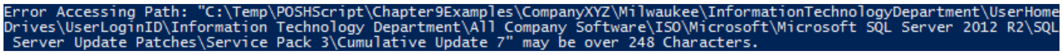
Chapter 9: File Scanning



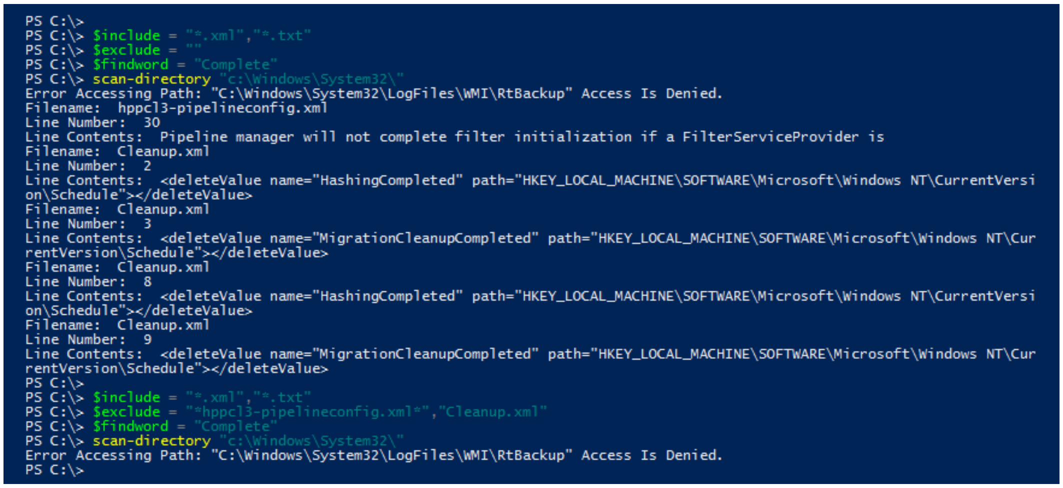
```
PS C:\> Get-ChildItem "c:\Program Files\" -Include *.log,*.txt -recurse

    Directory: C:\Program Files\ESET


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         3/5/2017   8:43 PM              0 eset_install.log

    Directory: C:\Program Files\Windows Defender


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/16/2016   6:43 AM           1091 ThirdPartyNotices.txt

    Directory: C:\Program Files\Windows NT\TableTextService


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/16/2016   6:42 AM          14186 TableTextServiceAmharic.txt
-a----        7/16/2016   6:42 AM        1272944 TableTextServiceArray.txt
-a----        7/16/2016   6:42 AM         980224 TableTextServiceDaYi.txt
-a----        7/16/2016   6:43 AM          14198 TableTextServiceTigrinya.txt
-a----        7/16/2016   6:42 AM          45170 TableTextServiceYi.txt

    Directory: C:\Program Files\WindowsPowerShell\Modules\Pester\3.4.0\en-US


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        7/16/2016   6:43 AM           3110 about_BeforeEach_AfterEach.help.txt
-a----        7/16/2016   6:43 AM           6396 about_Mocking.help.txt
-a----        7/16/2016   6:43 AM           5056 about_Pester.help.txt
-a----        7/16/2016   6:43 AM           5945 about_should.help.txt
-a----        7/16/2016   6:43 AM           1156 about_TestDrive.help.txt
```



```
PS C:\> $matches = Get-ChildItem "c:\Program Files\" -Include *.log,*.txt -recurse | select-string -pattern "Complete" -
SimpleMatch
PS C:\> foreach ($match in $matches) {
>>     Write-Host "Filename: " $match.FileName
>>     Write-Host "Line Number: " $match.LineNumber
>>     Write-Host "Line Contents: " $match.Line
>> }
Filename:  about_TestDrive.help.txt
Line Number:  14
Line Contents:          completes. You may use this drive to isolate the file operations of your
Filename:  about_TestDrive.help.txt
Line Number:  33
Line Contents:          When this test completes, the contents of the TestDrive PSDrive will
```

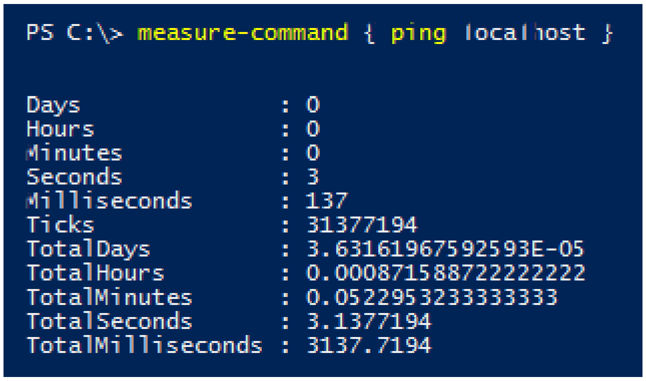


```
Error Accessing Path: "C:\Temp\POSHScript\Chapter9Examples\CompanyXYZ\Milwaukee\InformationTechnologyDepartment\UserHome
Drives\UserLoginID\Information Technology Department\All Company Software\ISO\Microsoft\Microsoft SQL Server 2012 R2\SQL
 Server Update Patches\Service Pack 3\Cumulative Update 7" may be over 248 Characters.
```

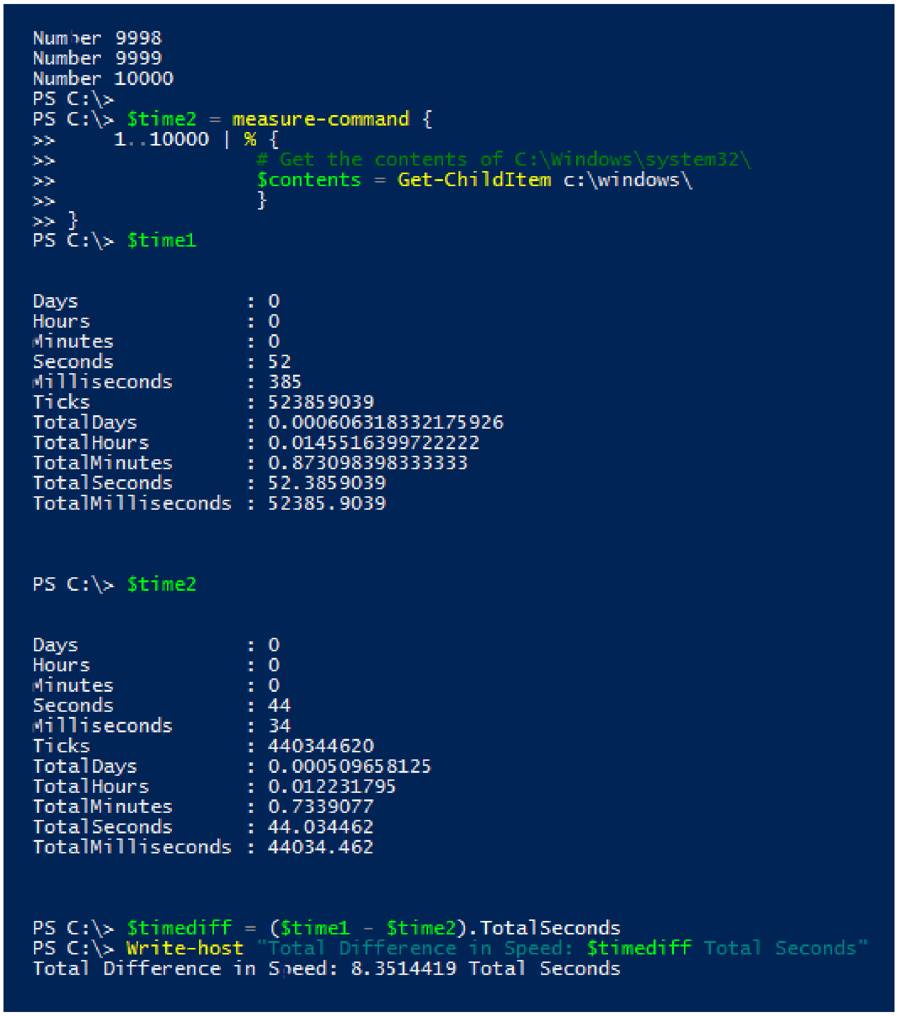


```
PS C:\>
PS C:\> $include = "*.xml","*.txt"
PS C:\> $exclude = ""
PS C:\> $findword = "Complete"
PS C:\> scan-directory "c:\Windows\System32\"
Error Accessing Path: "C:\Windows\System32\LogFiles\WMI\RtBackup" Access Is Denied.
Filename:  hppcl3-pipelineconfig.xml
Line Number:  30
Line Contents:  Pipeline manager will not complete filter initialization if a FilterServiceProvider is
Filename:  Cleanup.xml
Line Number:  2
Line Contents:  <deleteValue name="HashingCompleted" path="HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersi
on\Schedule"></deleteValue>
Filename:  Cleanup.xml
Line Number:  3
Line Contents:  <deleteValue name="MigrationCleanupCompleted" path="HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\Cur
rentVersion\Schedule"></deleteValue>
Filename:  Cleanup.xml
Line Number:  8
Line Contents:  <deleteValue name="HashingCompleted" path="HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersi
on\Schedule"></deleteValue>
Filename:  Cleanup.xml
Line Number:  9
Line Contents:  <deleteValue name="MigrationCleanupCompleted" path="HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\Cur
rentVersion\Schedule"></deleteValue>
PS C:\>
PS C:\> $include = "*.xml","*.txt"
PS C:\> $exclude = "*hppcl3-pipelineconfig.xml*","Cleanup.xml"
PS C:\> $findword = "Complete"
PS C:\> scan-directory "c:\Windows\System32\"
Error Accessing Path: "C:\Windows\System32\LogFiles\WMI\RtBackup" Access Is Denied.
PS C:\>
```

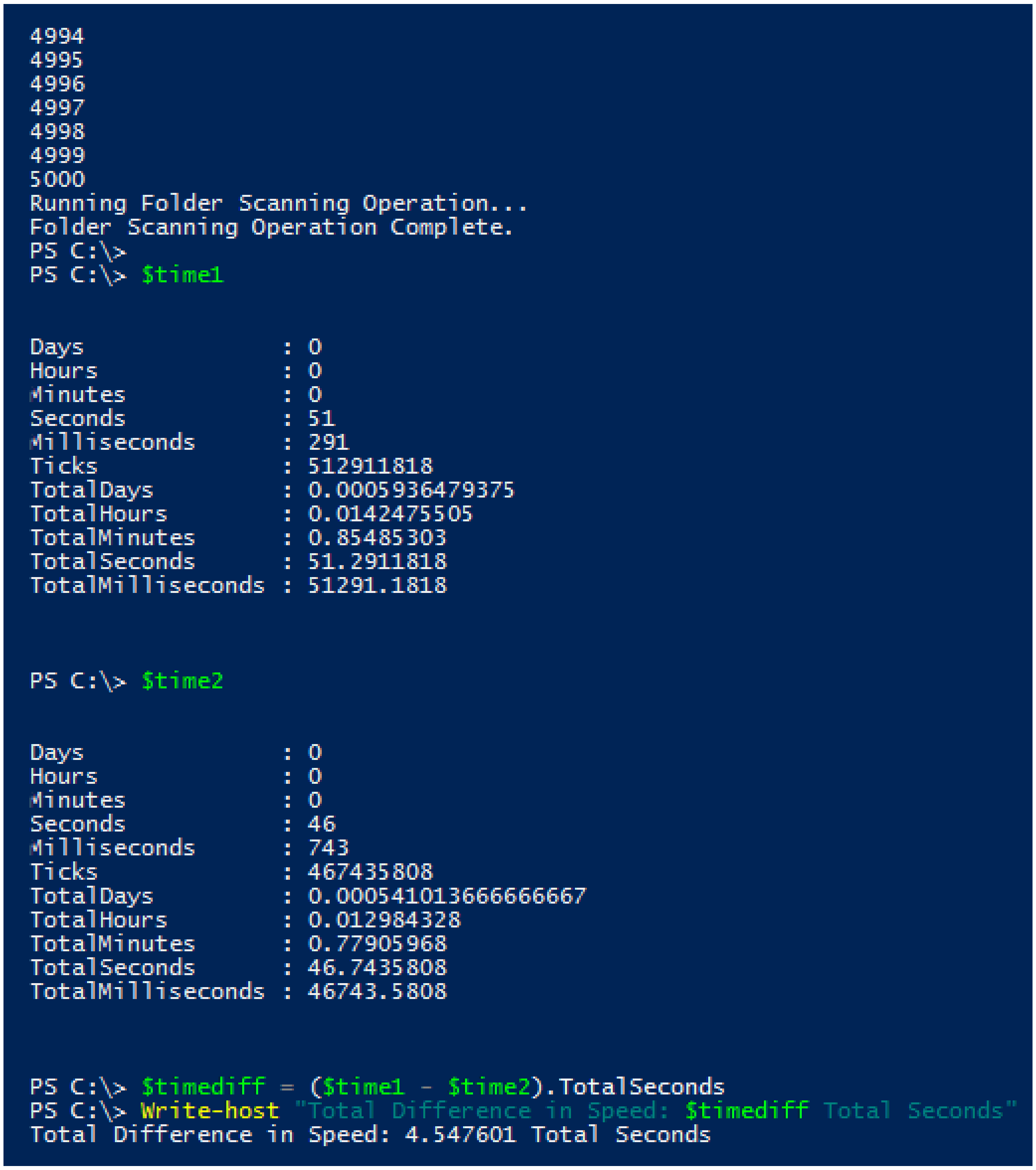
Chapter 10: Optimizing Script Execution Speed



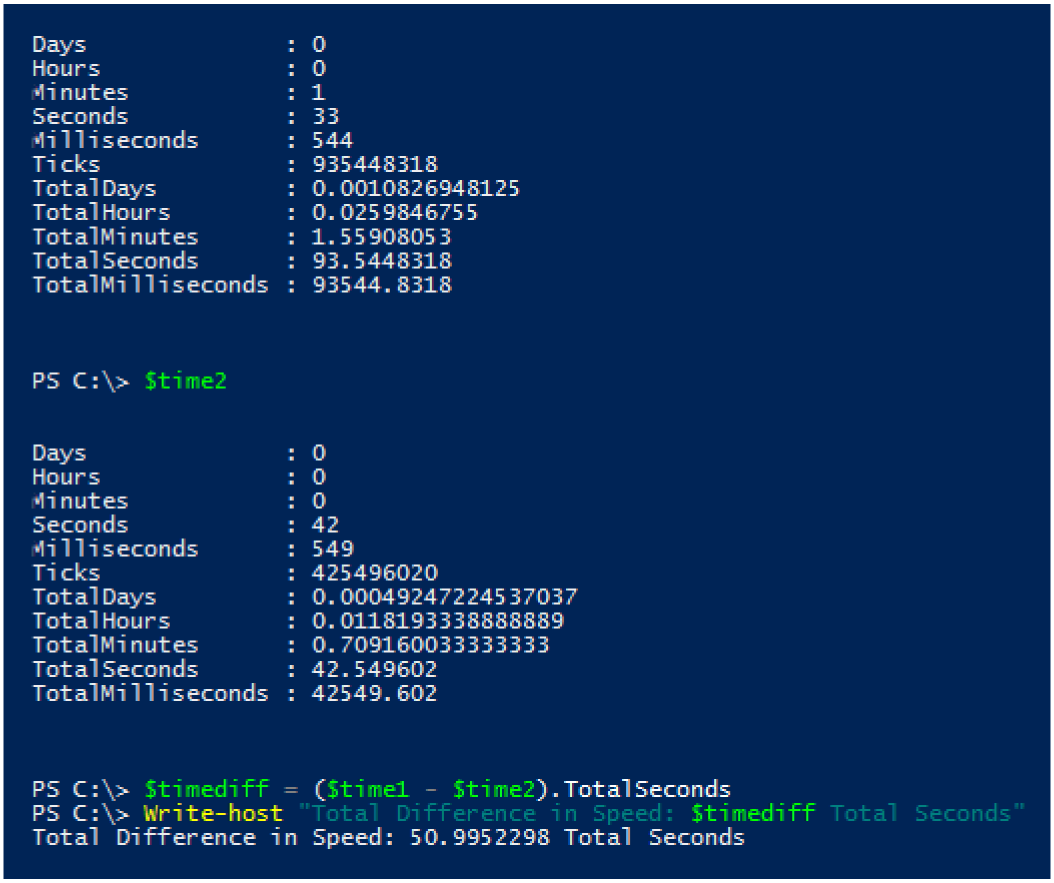
```
PS C:\> measure-command { ping localhost }


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 3
Milliseconds      : 137
Ticks             : 31377194
TotalDays         : 3.63161967592593E-05
TotalHours        : 0.000871588722222222
TotalMinutes      : 0.0522953233333333
TotalSeconds      : 3.1377194
TotalMilliseconds : 3137.7194
```



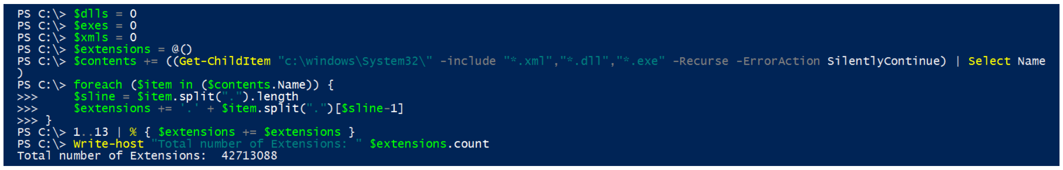
```
Number 9998
Number 9999
Number 10000
PS C:\>
PS C:\> $time2 = measure-command {
>>      1..10000 | % {
>>                  # Get the contents of C:\Windows\system32\
>>                  $contents = Get-ChildItem c:\windows\
>>                  }
>> }
PS C:\> $time1


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 52
Milliseconds      : 385
Ticks             : 523859039
TotalDays         : 0.000606318332175926
TotalHours        : 0.0145516399722222
TotalMinutes      : 0.873098398333333
TotalSeconds      : 52.3859039
TotalMilliseconds : 52385.9039



PS C:\> $time2


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 44
Milliseconds      : 34
Ticks             : 440344620
TotalDays         : 0.000509658125
TotalHours        : 0.012231795
TotalMinutes      : 0.7339077
TotalSeconds      : 44.034462
TotalMilliseconds : 44034.462



PS C:\> $timediff = ($time1 - $time2).TotalSeconds
PS C:\> Write-host "Total Difference in Speed: $timediff Total Seconds"
Total Difference in Speed: 8.3514419 Total Seconds
```



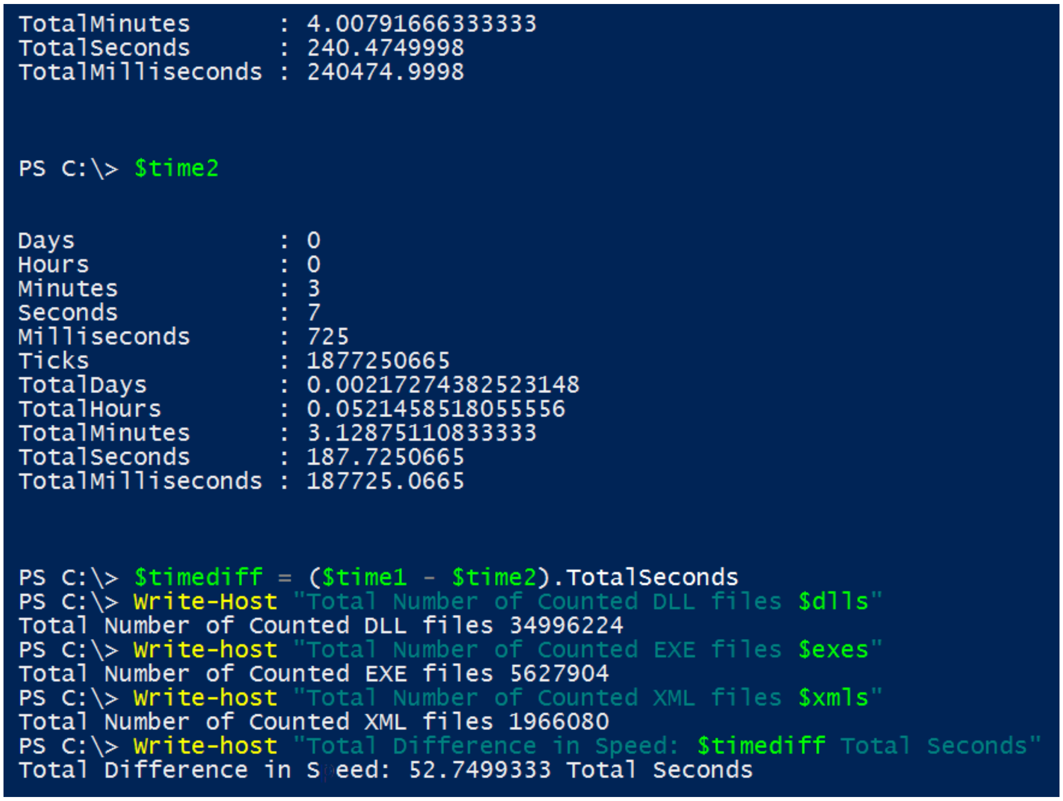
```
4994
4995
4996
4997
4998
4999
5000
Running Folder Scanning Operation...
Folder Scanning Operation Complete.
PS C:\>
PS C:\> $time1


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 51
Milliseconds      : 291
Ticks             : 512911818
TotalDays         : 0.0005936479375
TotalHours        : 0.0142475505
TotalMinutes      : 0.85485303
TotalSeconds      : 51.2911818
TotalMilliseconds : 51291.1818



PS C:\> $time2


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 46
Milliseconds      : 743
Ticks             : 467435808
TotalDays         : 0.000541013666666667
TotalHours        : 0.012984328
TotalMinutes      : 0.77905968
TotalSeconds      : 46.7435808
TotalMilliseconds : 46743.5808



PS C:\> $timediff = ($time1 - $time2).TotalSeconds
PS C:\> Write-host "Total Difference in Speed: $timediff Total Seconds"
Total Difference in Speed: 4.547601 Total Seconds
```



```
Days              : 0
Hours             : 0
Minutes           : 1
Seconds           : 33
Milliseconds      : 544
Ticks             : 935448318
TotalDays         : 0.0010826948125
TotalHours        : 0.0259846755
TotalMinutes      : 1.55908053
TotalSeconds      : 93.5448318
TotalMilliseconds : 93544.8318



PS C:\> $time2


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 42
Milliseconds      : 549
Ticks             : 425496020
TotalDays         : 0.000492472245337037
TotalHours        : 0.0118193338888889
TotalMinutes      : 0.709160033333333
TotalSeconds      : 42.549602
TotalMilliseconds : 42549.602



PS C:\> $timediff = ($time1 - $time2).TotalSeconds
PS C:\> Write-host "Total Difference in Speed: $timediff Total Seconds"
Total Difference in Speed: 50.9952298 Total Seconds
```

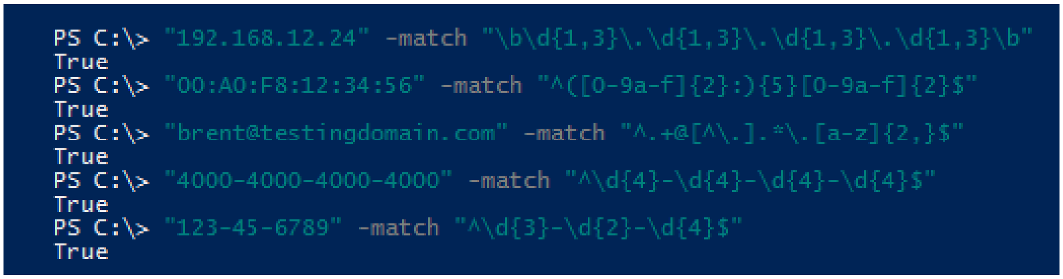


```
PS C:\> $dlls = 0
PS C:\> $exes = 0
PS C:\> $xmls = 0
PS C:\> $extensions = @()
PS C:\> $contents += ((Get-ChildItem "c:\windows\System32\" -include "*.xml","*.dll","*.exe" -Recurse -ErrorAction SilentlyContinue) | Select Name
)
PS C:\> foreach ($item in ($contents.Name)) {
>>>     $sline = $item.split(".").length
>>>     $extensions += '.' + $item.split(".")[$sline-1]
>>> }
PS C:\> 1..13 | % { $extensions += $extensions }
PS C:\> Write-host "Total number of Extensions: " $extensions.count
Total number of Extensions:  42713088
```

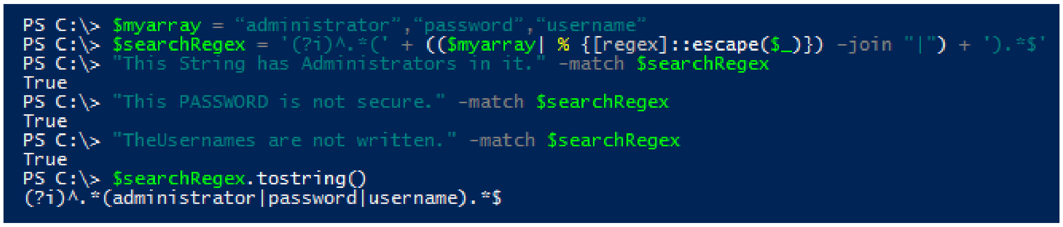


```
TotalMinutes      : 4.00791666333333
TotalSeconds      : 240.4749998
TotalMilliseconds : 240474.9998



PS C:\> $time2


Days              : 0
Hours             : 0
Minutes           : 3
Seconds           : 7
Milliseconds      : 725
Ticks             : 1877250665
TotalDays         : 0.00217274382523148
TotalHours        : 0.0521458518055556
TotalMinutes      : 3.12875110833333
TotalSeconds      : 187.7250665
TotalMilliseconds : 187725.0665



PS C:\> $timediff = ($time1 - $time2).TotalSeconds
PS C:\> Write-Host "Total Number of Counted DLL files $dlls"
Total Number of Counted DLL files 34996224
PS C:\> Write-host "Total Number of Counted EXE files $exes"
Total Number of Counted EXE files 5627904
PS C:\> Write-host "Total Number of Counted XML files $xmls"
Total Number of Counted XML files 1966080
PS C:\> Write-host "Total Difference in Speed: $timediff Total Seconds"
Total Difference in Speed: 52.7499333 Total Seconds
```

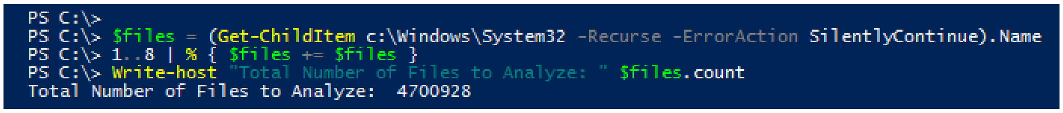
Chapter 11: Improving Performance by Using Regular Expressions



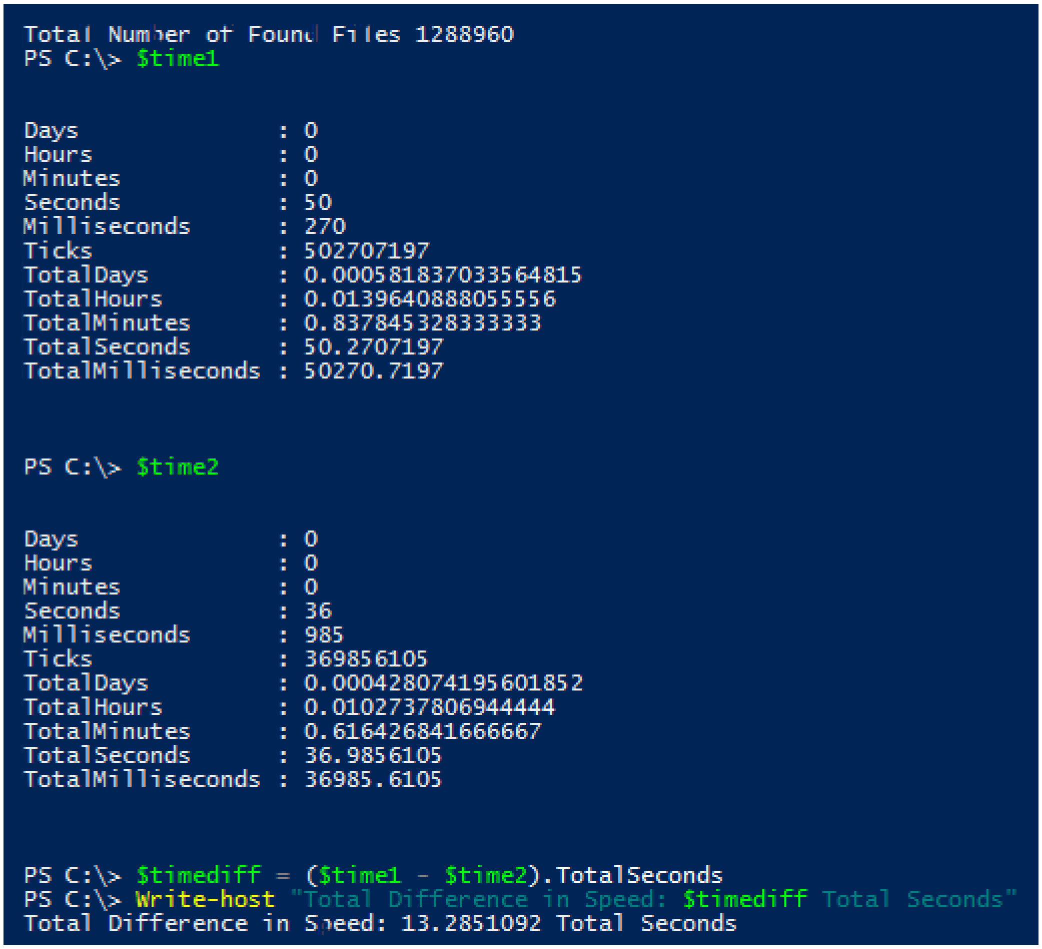
```
PS C:\> "192.168.12.24" -match "\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b"
True
PS C:\> "00:A0:F8:12:34:56" -match "^([0-9a-f]{2}:){5}[0-9a-f]{2}$"
True
PS C:\> "brent@testingdomain.com" -match "^.+@[^\.].*\.[a-z]{2,}$"
True
PS C:\> "4000-4000-4000-4000" -match "^\d{4}-\d{4}-\d{4}-\d{4}$"
True
PS C:\> "123-45-6789" -match "^\d{3}-\d{2}-\d{4}$"
True
```



```
PS C:\> $myarray = “administrator”,“password”,“username”
PS C:\> $searchRegex = '(?i)^.*(' + (($myarray| % {[regex]::escape($_)}) -join "|") + ').*$'
PS C:\> "This String has Administrators in it." -match $searchRegex
True
PS C:\> "This PASSWORD is not secure." -match $searchRegex
True
PS C:\> "TheUsernames are not written." -match $searchRegex
True
PS C:\> $searchRegex.tostring()
(?i)^.*(administrator|password|username).*$
```

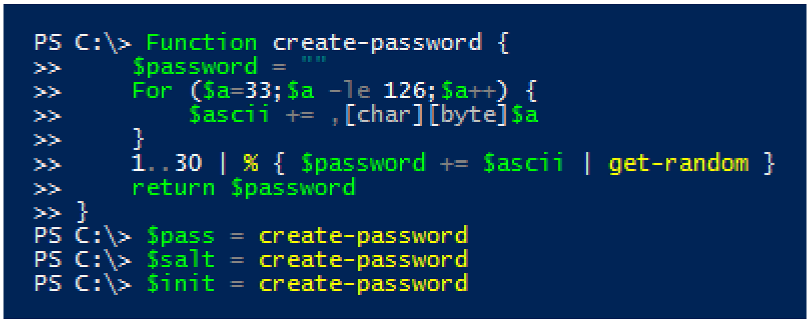


```
PS C:\>
PS C:\> $files = (Get-ChildItem c:\Windows\System32 -Recurse -ErrorAction SilentlyContinue).Name
PS C:\> 1..8 | % { $files += $files }
PS C:\> Write-host "Total Number of Files to Analyze: " $files.count
Total Number of Files to Analyze:  4700928
```

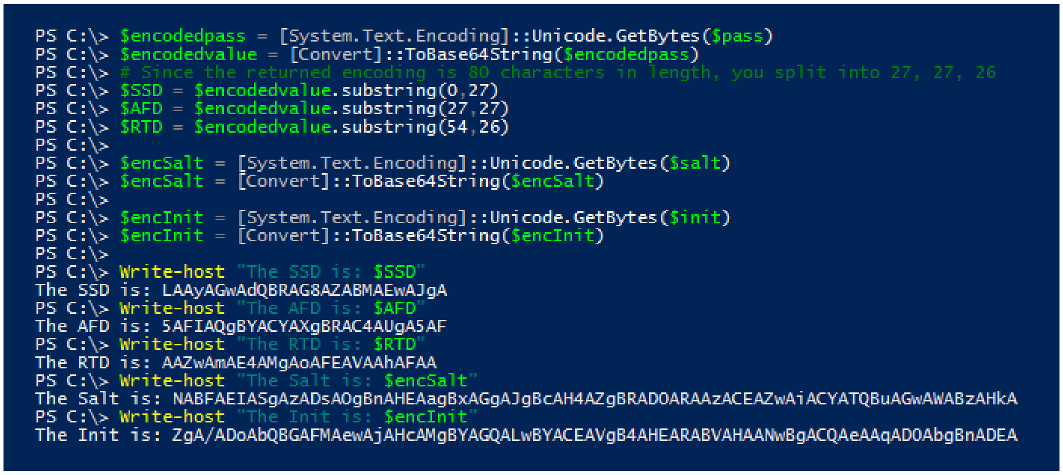


```
Total Number of Found Files 1288960
PS C:\> $time1


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 50
Milliseconds      : 270
Ticks             : 502707197
TotalDays         : 0.000581837033564815
TotalHours        : 0.0139640888055556
TotalMinutes      : 0.837845328333333
TotalSeconds      : 50.2707197
TotalMilliseconds : 50270.7197



PS C:\> $time2


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 36
Milliseconds      : 985
Ticks             : 369856105
TotalDays         : 0.000428074195601852
TotalHours        : 0.0102737806944444
TotalMinutes      : 0.616426841666667
TotalSeconds      : 36.9856105
TotalMilliseconds : 36985.6105



PS C:\> $timediff = ($time1 - $time2).TotalSeconds
PS C:\> Write-host "Total Difference in Speed: $timediff Total Seconds"
Total Difference in Speed: 13.2851092 Total Seconds
```

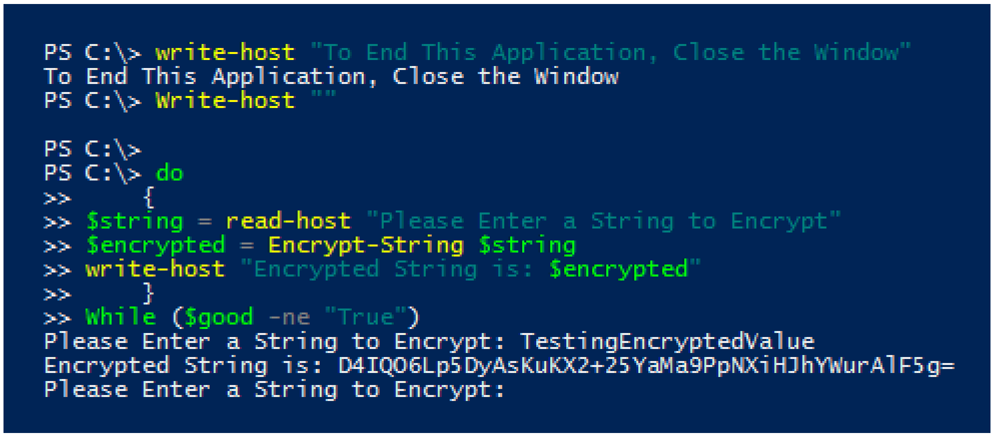
Chapter 12: Overall Script Workflow, Termination Files, and Merging Data Results



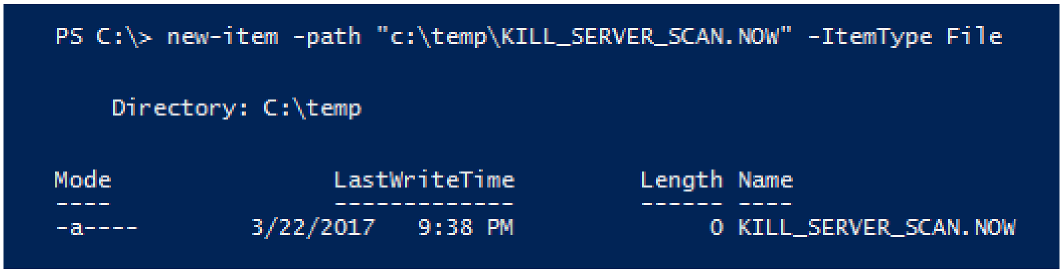
```
PS C:\> Function create-password {
>>     $password = ""
>>     For ($a=33;$a -le 126;$a++) {
>>         $ascii += ,[char][byte]$a
>>     }
>>     1..30 | % { $password += $ascii | get-random }
>>     return $password
>> }
PS C:\> $pass = create-password
PS C:\> $salt = create-password
PS C:\> $init = create-password
```



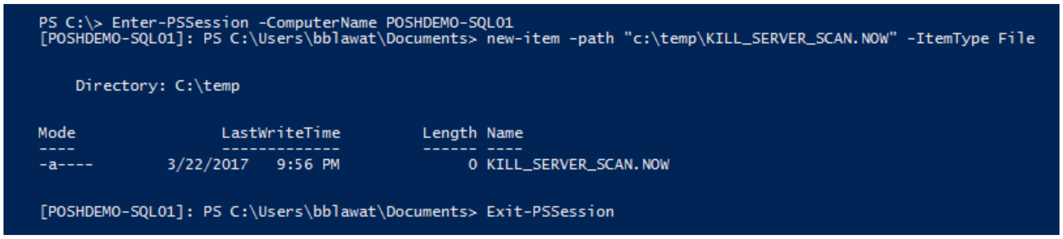
```
PS C:\> $encodedpass = [System.Text.Encoding]::Unicode.GetBytes($pass)
PS C:\> $encodedvalue = [Convert]::ToBase64String($encodedpass)
PS C:\> # Since the returned encoding is 80 characters in length, you split into 27, 27, 26
PS C:\> $SSD = $encodedvalue.substring(0,27)
PS C:\> $AFD = $encodedvalue.substring(27,27)
PS C:\> $RTD = $encodedvalue.substring(54,26)
PS C:\>
PS C:\> $encSalt = [System.Text.Encoding]::Unicode.GetBytes($salt)
PS C:\> $encSalt = [Convert]::ToBase64String($encSalt)
PS C:\>
PS C:\> $encInit = [System.Text.Encoding]::Unicode.GetBytes($init)
PS C:\> $encInit = [Convert]::ToBase64String($encInit)
PS C:\>
PS C:\> Write-host "The SSD is: $SSD"
The SSD is: LAAyAGwAdQBRAG8AZABMAEwAJgA
PS C:\> Write-host "The AFD is: $AFD"
The AFD is: 5AFIAQgBYACYAXgBRAC4AUgA5AF
PS C:\> Write-host "The RTD is: $RTD"
The RTD is: AAZwAmAE4AMgAoAFEAVAAhAFAA
PS C:\> Write-host "The Salt is: $encSalt"
The Salt is: NABFAEIASgAzADsAOgBnAHEAagBxAGgAJgBcAH4AZgBRADOARAAzACEAZwAiACYATQBuAGwAWABzAHkA
PS C:\> Write-host "The Init is: $encInit"
The Init is: ZgA/ADoAbQBGAFMAewAjAHcAMgBYAGQALwBYACEAVgB4AHEARABVAHAANwBgACQAeAAqADOAbgBnADEA
```



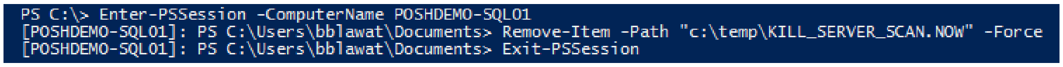
```
PS C:\> write-host "To End This Application, Close the Window"
To End This Application, Close the Window
PS C:\> Write-host ""

PS C:\>
PS C:\> do
>>      {
>> $string = read-host "Please Enter a String to Encrypt"
>> $encrypted = Encrypt-String $string
>> write-host "Encrypted String is: $encrypted"
>>      }
>> While ($good -ne "True")
Please Enter a String to Encrypt: TestingEncryptedValue
Encrypted String is: D4IQO6Lp5DyAsKuKX2+25YaMa9PpNXiHJhYWurAlF5g=
Please Enter a String to Encrypt:
```



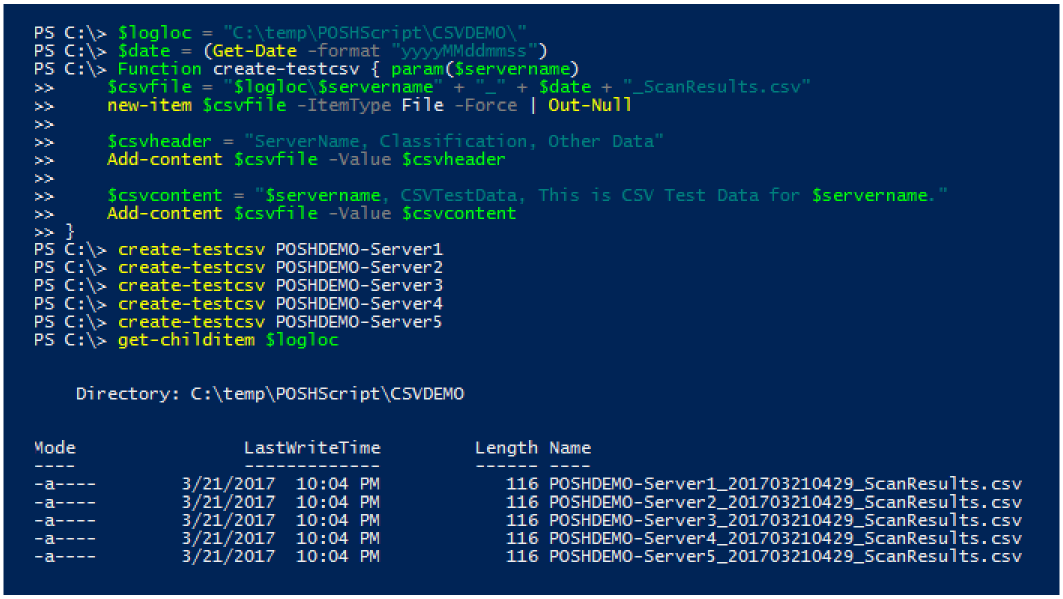
```
PS C:\> new-item -path "c:\temp\KILL_SERVER_SCAN.NOW" -ItemType File


    Directory: C:\temp


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        3/22/2017   9:38 PM              0 KILL_SERVER_SCAN.NOW
```



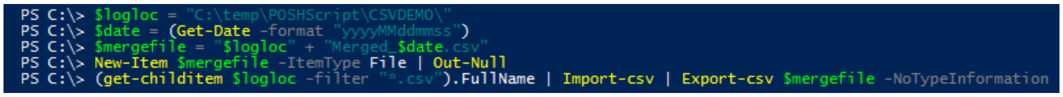
```
PS C:\> Enter-PSSession -ComputerName POSHDEMO-SQL01
[POSHDEMO-SQL01]: PS C:\Users\bblawat\Documents> new-item -path "c:\temp\KILL_SERVER_SCAN.NOW" -ItemType File


    Directory: C:\temp


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        3/22/2017   9:56 PM              0 KILL_SERVER_SCAN.NOW


[POSHDEMO-SQL01]: PS C:\Users\bblawat\Documents> Exit-PSSession
```



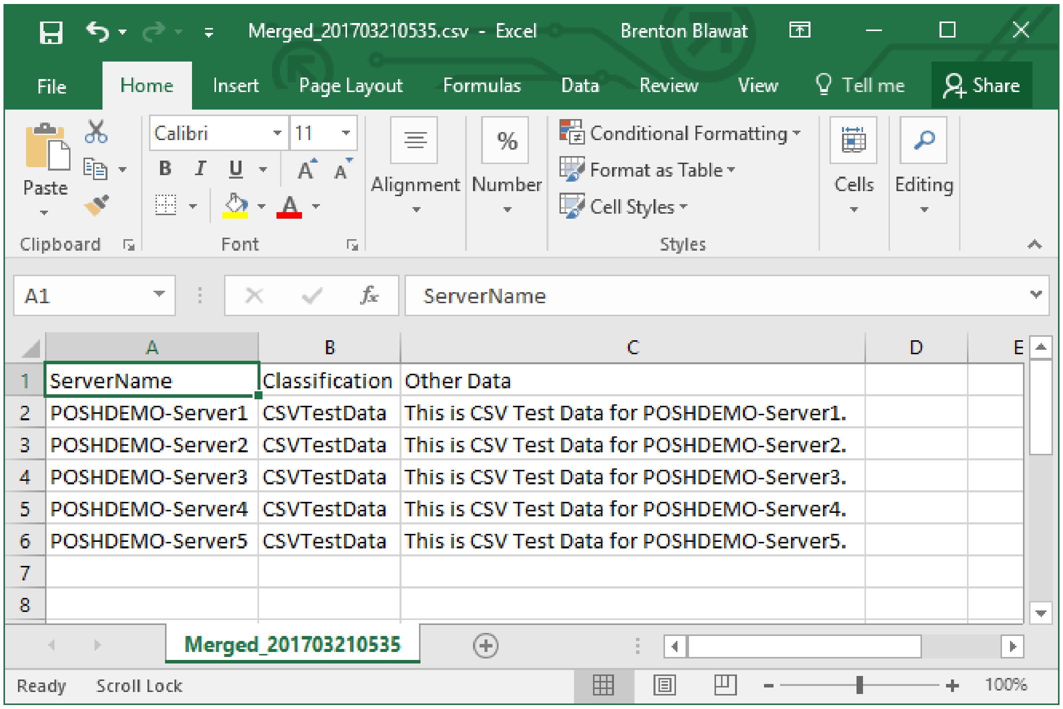
```
PS C:\> Enter-PSSession -ComputerName POSHDEMO-SQL01
[POSHDEMO-SQL01]: PS C:\Users\bblawat\Documents> Remove-Item -Path "c:\temp\KILL_SERVER_SCAN.NOW" -Force
[POSHDEMO-SQL01]: PS C:\Users\bblawat\Documents> Exit-PSSession
```



```
PS C:\> $logloc = "C:\temp\POSHScript\CSVDEMO\"
PS C:\> $date = (Get-Date -format "yyyyMMddmmss")
PS C:\> Function create-testcsv { param($servername)
>>     $csvfile = "$logloc\$servername" + "_" + $date + "_ScanResults.csv"
>>     new-item $csvfile -ItemType File -Force | Out-Null
>>
>>     $csvheader = "ServerName, Classification, Other Data"
>>     Add-content $csvfile -Value $csvheader
>>
>>     $csvcontent = "$servername, CSVTestData, This is CSV Test Data for $servername."
>>     Add-content $csvfile -Value $csvcontent
>> }
PS C:\> create-testcsv POSHDEMO-Server1
PS C:\> create-testcsv POSHDEMO-Server2
PS C:\> create-testcsv POSHDEMO-Server3
PS C:\> create-testcsv POSHDEMO-Server4
PS C:\> create-testcsv POSHDEMO-Server5
PS C:\> get-childitem $logloc


    Directory: C:\temp\POSHScript\CSVDEMO


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        3/21/2017  10:04 PM            116 POSHDEMO-Server1_201703210429_ScanResults.csv
-a----        3/21/2017  10:04 PM            116 POSHDEMO-Server2_201703210429_ScanResults.csv
-a----        3/21/2017  10:04 PM            116 POSHDEMO-Server3_201703210429_ScanResults.csv
-a----        3/21/2017  10:04 PM            116 POSHDEMO-Server4_201703210429_ScanResults.csv
-a----        3/21/2017  10:04 PM            116 POSHDEMO-Server5_201703210429_ScanResults.csv
```



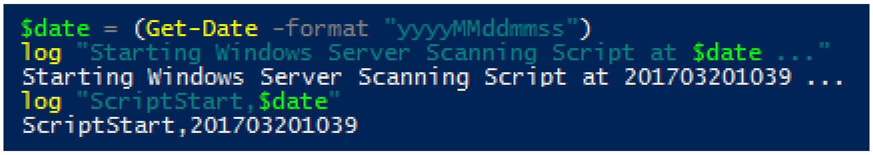
```
PS C:\> $logloc = "C:\temp\POSHScript\CSVDEMO\"
PS C:\> $date = (Get-Date -format "yyyyMMddmmss")
PS C:\> $mergefile = "$logloc" + "Merged_$date.csv"
PS C:\> New-Item $mergefile -ItemType File | Out-Null
PS C:\> (get-childitem $logloc -filter "*.csv").FullName | Import-csv | Export-csv $mergefile -NoTypeInformation
```



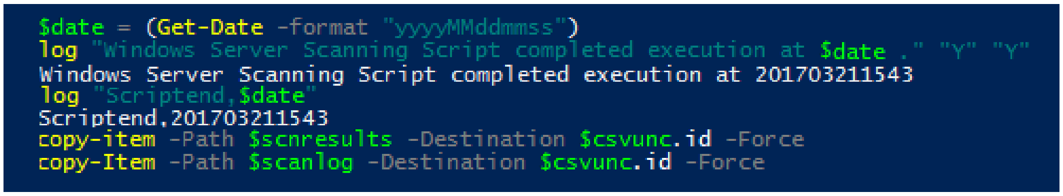
	A	B	C
1	ServerName	Classification	Other Data
2	POSHDEMO-Server1	CSVTestData	This is CSV Test Data for POSHDEMO-Server1.
3	POSHDEMO-Server2	CSVTestData	This is CSV Test Data for POSHDEMO-Server2.
4	POSHDEMO-Server3	CSVTestData	This is CSV Test Data for POSHDEMO-Server3.
5	POSHDEMO-Server4	CSVTestData	This is CSV Test Data for POSHDEMO-Server4.
6	POSHDEMO-Server5	CSVTestData	This is CSV Test Data for POSHDEMO-Server5.

Chapter 13: Creating the Windows Server Scanning Script and Post Execution Cleanup

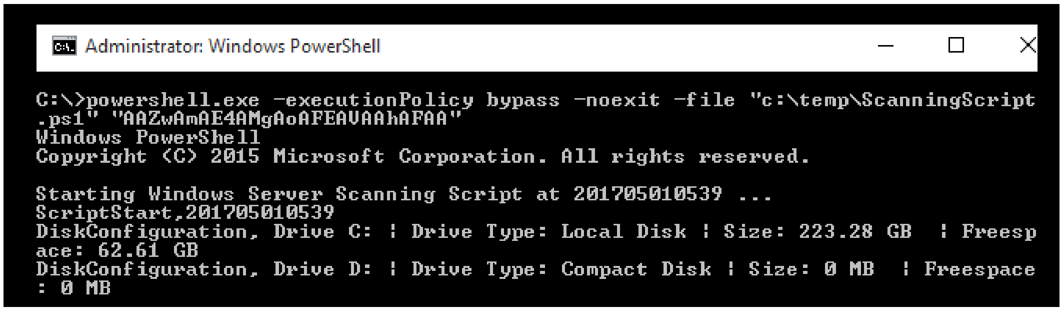
```
<#
.SYNOPSIS
This is a server discovery script which will scan different server components to determine
the current configuration.

.DESCRIPTION
This script will scan processes, Windows services, scheduled tasks, server features, disk information,
registry, and files for pertinent server information.

Author: Brenton J.W. Blawat / Packt Publishing / Author / email@email.com

.PARAMETER RTD
This script requires a run time decryptor as a parameter to the script.

.EXAMPLE
powershellscript.ps1 /RTD "Run Time Decryptor"

.NOTES
You must have administrative rights to the server you are scanning. Certain functions will not work properly
without running the script as system or administrator.#>
```



```
$date = (Get-Date -format "yyyyMMddmmss")
log "Starting Windows Server Scanning Script at $date ..."
Starting Windows Server Scanning Script at 201703201039 ...
log "ScriptStart,$date"
ScriptStart,201703201039
```



```
$date = (Get-Date -format "yyyyMMddmmss")
log "Windows Server Scanning Script completed execution at $date ." "Y" "Y"
Windows Server Scanning Script completed execution at 201703211543
log "Scriptend,$date"
Scriptend,201703211543
copy-item -Path $scnresults -Destination $csvunc.id -Force
copy-Item -Path $scanlog -Destination $csvunc.id -Force
```



```
C:\>powershell.exe -executionPolicy bypass -noexit -file "c:\temp\ScanningScript
.ps1" "AAZwAmAE4AMgAoAFEAVAAhAFAA"
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

Starting Windows Server Scanning Script at 201705010539 ...
ScriptStart,201705010539
DiskConfiguration, Drive C: ¦ Drive Type: Local Disk ¦ Size: 223.28 GB  ¦ Freesp
ace: 62.61 GB
DiskConfiguration, Drive D: ¦ Drive Type: Compact Disk ¦ Size: 0 MB  ¦ Freespace
: 0 MB
```