Chapter 26 Lab

Create a proxy function for the Export-CSV cmdlet. Name the proxy function Export-TDF. Remove the –Delimiter parameter, and instead hardcode it to always use –Delimiter "`t" (that’s a backtick, followed by the letter “T,” in double quotation marks).

Work with the proxy function in a script file. At the bottom of the file, after the closing } of the function, put the following to test the function:

Get-Service | Export-TDF c:\services.tdf

Run the script to test the function, and verify that it creates a tab-delimited file named c:\services.tdf.

Here is one possible solution with comments that explain what we did:

<#

First, we need to run these lines to create the metadata:

$metadata = New-Object System.Management.Automation.CommandMetaData (Get-Command Export-CSV)

[System.Management.Automation.ProxyCommand]::Create($metadata) | Out-File ProxyExportCSV.ps1

#>

Function Export-TDF {

#we deleted the help link in cmdletbinding and added our own

<#

.Synopsis

Export to tab delimited file

.Description

This is a proxy command to Export-CSV which is hard coded to export

data to a tab-delimited file.

#>

[CmdletBinding(DefaultParameterSetName='Delimiter',

SupportsShouldProcess=$true,

ConfirmImpact='Medium'

)

]

param(

[Parameter(Mandatory=$true, ValueFromPipeline=$true, ValueFromPipelineByPropertyName=$true)]

[psobject]

${InputObject},

[Parameter(Position=0)]

[ValidateNotNullOrEmpty()]

[string]

${Path},

[Alias('PSPath')]

[ValidateNotNullOrEmpty()]

[string]

${LiteralPath},

[switch]

${Force},

[Alias('NoOverwrite')]

[switch]

${NoClobber},

[ValidateSet('Unicode','UTF7','UTF8','ASCII','UTF32','BigEndianUnicode','Default','OEM')]

[string]

${Encoding},

[switch]

${Append},

#we deleted the Delimiter parameter that used to be here

[Parameter(ParameterSetName='UseCulture')]

[switch]

${UseCulture},

[Alias('NTI')]

[switch]

${NoTypeInformation})

begin

{

try {

$outBuffer = $null

if ($PSBoundParameters.TryGetValue('OutBuffer', [ref]$outBuffer))

{

$PSBoundParameters['OutBuffer'] = 1

}

$wrappedCmd = $ExecutionContext.InvokeCommand.GetCommand('Export-Csv', [System.Management.Automation.CommandTypes]::Cmdlet)

<#

we added a hard coded reference to include the original -delimiter parameter

with the tab character.

#>

$scriptCmd = {& $wrappedCmd @PSBoundParameters -delimiter "`t"}

$steppablePipeline = $scriptCmd.GetSteppablePipeline($myInvocation.CommandOrigin)

$steppablePipeline.Begin($PSCmdlet)

} catch {

throw

}

}

process

{

try {

$steppablePipeline.Process($\_)

} catch {

throw

}

}

end

{

try {

$steppablePipeline.End()

} catch {

throw

}

}

#We deleted the links for forwarded help

} #end function

#test it out

Get-Service | Export-TDF c:\services.tdf