Title: Network Workstation Management

Name: Zhengzheng Li

Date: 2025/3/5

a. Purpose

The purpose of this PowerShell script is to simplify and automate common administrative tasks for managing network workstations in an organization. By using this script, system administrators can save time, reduce manual errors, and ensure consistent execution of routine tasks.

b. Function

The script performs the following tasks in order:

1. Retrieving System Information

   - ◎ The script uses the **Get-WmiObject** cmdlet to retrieve system information such as OS version, CPU usage, memory usage, and disk space.
   - ◎ This information is displayed in a readable format for quick analysis.

2. Restarting Remote Services

   - ◎ The script uses the **Get-Service** and **Restart-Service** cmdlets to restart a specified service on a remote workstation.
   - ◎ It checks if the service exists and is running before attempting to restart it.

3. Creating a New User Account

   - ◎ The script uses the **New-LocalUser** cmdlet to create a new local user account on a workstation.
   - ◎ It allows the administrator to specify the username, password, and description for the new account.

C. Usage

The script is designed to be run from a PowerShell terminal with administrative privileges. It accepts arguments to specify the task to perform and any required parameters.

Syntax

```
.\NetworkWorkstationManagement.ps1 -Task <TaskName> [-ComputerName <ComputerName>] [-ServiceName <ServiceName>] [-Username <Username>] [-Password <Password>] [-Description <Description>]
```

Options and Arguments

-Task: Specifies the task to perform. Valid values are:

- **SystemInfo**: Retrieves system information.
- **RestartService**: Restarts a specified service.
- **CreateUser**: Creates a new local user account.

-ComputerName: (Optional) Specifies the name of the remote computer. If not provided, the script runs on the local machine.
-ServiceName: (Required for **RestartService**) Specifies the name of the service to restart.
-Username: (Required for **CreateUser**) Specifies the username for the new local account.
-Password: (Required for **CreateUser**) Specifies the password for the new local account.
-Description: (Optional for **CreateUser**) Specifies a description for the new local account.

Examples

1. Retrieve system information for the local machine:

   ```
   .\NetworkWorkstationManagement.ps1 -Task SystemInfo
   ```

2. Restart the "Spooler" service on a remote computer:

   ```
   .\NetworkWorkstationManagement.ps1 -Task RestartService -ComputerName "RemotePC01" -ServiceName "Spooler"
   ```

3. Create a new local user account on the local machine:

   ```
   .\NetworkWorkstationManagement.ps1 -Task CreateUser -Username "JohnDoe" -Password "P@ssw0rd" -Description "New IT Staff"
   ```

d. Script

Below is the complete PowerShell script:

```
param (
    [Parameter(Mandatory = $true)]
    [ValidateSet("SystemInfo", "RestartService", "CreateUser")]
    [string]$Task,

    [Parameter(Mandatory = $false)]
    [string]$ComputerName = $env:COMPUTERNAME,

    [Parameter(Mandatory = $false)]
    [string]$ServiceName,

    [Parameter(Mandatory = $false)]
    [string]$Username,

    [Parameter(Mandatory = $false)]
    [string]$Password,

    [Parameter(Mandatory = $false)]
    [string]$Description
)

function Get-SystemInfo {
    param (
        [string]$ComputerName
    )
    Write-Host "Retrieving system information for $ComputerName..." -ForegroundColor Green
    $os = Get-WmiObject -Class Win32_OperatingSystem -ComputerName $ComputerName
    $cpu = Get-WmiObject -Class Win32_Processor -ComputerName $ComputerName
    $memory = Get-WmiObject -Class Win32_PhysicalMemory -ComputerName
$ComputerName
    $disk = Get-WmiObject -Class Win32_LogicalDisk -Filter "DeviceID='C:'" -ComputerName
$ComputerName

    # Calculate total memory from all RAM modules
    $totalMemory = ($memory | Measure-Object -Property Capacity -Sum).Sum

    Write-Host "OS Version: $($os.Caption) $($os.Version)"
    Write-Host "CPU: $($cpu.Name)"
    Write-Host "Total Memory: $([math]::round($totalMemory / 1GB, 2)) GB"
    Write-Host "Free Disk Space: $([math]::round($disk.FreeSpace / 1GB, 2)) GB"
}

function Restart-RemoteService {
    param (
        [string]$ComputerName,
        [string]$ServiceName
    )
    Write-Host "Restarting service '$ServiceName' on $ComputerName..." -ForegroundColor
Green
    $service = Get-Service -Name $ServiceName -ComputerName $ComputerName -ErrorAction
SilentlyContinue
    if ($service) {
        if ($service.Status -eq "Running") {
            Restart-Service -InputObject $service -Force
            Write-Host "Service '$ServiceName' restarted successfully." -ForegroundColor
Green
        } else {
            Write-Host "Service '$ServiceName' is not running." -ForegroundColor Yellow
        }
    } else {
        Write-Host "Service '$ServiceName' not found." -ForegroundColor Red
    }
}

function Create-LocalUser {
    param (
        [string]$Username,
        [string]$Password,
        [string]$Description
    )
    Write-Host "Creating new local user '$Username'..." -ForegroundColor Green
    $securePassword = ConvertTo-SecureString -String $Password -AsPlainText -Force
    New-LocalUser -Name $Username -Password $securePassword -Description $Description
-AccountNeverExpires -PasswordNeverExpires
    Write-Host "User '$Username' created successfully." -ForegroundColor Green
}

# Main script logic
switch ($Task) {
    "SystemInfo" {
        Get-SystemInfo -ComputerName $ComputerName
    }
    "RestartService" {
            if (-not $ServiceName) {
                  Write-Host "Error: ServiceName parameter is required for RestartService task."
-ForegroundColor Red
                  exit
            }
            Restart-RemoteService -ComputerName $ComputerName -ServiceName $ServiceName
      }
      "CreateUser" {
            if (-not $Username -or -not $Password) {
                  Write-Host "Error: Username and Password parameters are required for
CreateUser task." -ForegroundColor Red
                  exit
            }
            Create-LocalUser -Username $Username -Password $Password -Description
$Description
      }
      default {
            Write-Host "Invalid task specified." -ForegroundColor Red
      }
}
```

Conclusion:

This PowerShell script provides a versatile and efficient way to perform common administrative tasks on network workstations. By automating these tasks, system administrators can focus on more critical issues and ensure the smooth operation of their organization's IT infrastructure. The script is easy to use and can be extended to include additional functionality as needed.