Yi' Solutions

Microsoft Windows Server 2022

Different system versions have different packaging methods. The packaging process includes: "Language pack: add, associate, delete", "Drive: add, delete", "Cumulative update: add, delete" etc.

There are many hidden stories hidden behind this. If you want to unlock these, are you ready to start trying to encapsulate them?

Summary

Chapter 1   Encapsulation

Microsoft Windows Server 2022

Table of contents

Chapter 1   ENCAPSULATION

A.   Prerequisites

II   Running operating system

When the operating system you are running is Windows 10 or lower than Windows 11 24H2, in some cases, using the DISM command to create a higher version image will cause some unknown problems. For example, when running the DISM command in the Windows 10 operating system to process the Windows Server 2025 offline image, you may receive an error message during the packaging process: "This application cannot run on your computer." Solution:

1.   Upgrade the running operating system or reinstall to a higher version (recommended);

2.   Upgrade or install a new version of ADK or PowerShell (not recommended)

    2.1.   You can try to upgrade to the latest PowerShell 7 or higher version;

    2.2.   After installing the latest version of ADK and replacing the DISM command, the problem of low DISM version can be solved. However, the command line mainly used by the packaging script is the PowerShell command line, so it is not recommended that you use the above method. The best method is to upgrade the running operating system or reinstall to a higher version.

II   ISO tools

Use a software that can edit ISO files, such as: PowerISO, DAEMON Tools, ISO Workshop;

III   Requirements

1.   System installation package

    1.1.   Prepare: en-us_windows_server_2022_x64_dvd_620d7eac.iso

    1.2.   Unzip to: D:\en-us_windows_server_2022_x64_dvd_620d7eac

    1.3.   After decompression is complete, change the directory en-us_windows_server_2022_x64_dvd_620d7eac to D:\OS_2022

        Note: Before decompressing to disk D, you should check whether it is a ReFS partition. If it is a ReFS partition, some commands will fail. Solution: Please use a disk partition in NTFS format.

    1.4.   All scripts and all paths have been set to D:\OS_2022 by default as the image source.

2.   Language Pack

    2.1.   Learn

        2.1.1.   Add languages to a Windows 11 image

        2.1.2.   Language and region Features on Demand (FOD)

    2.2.   Language pack: Download

        20348.1.210507-1500.fe_release_amd64fre_SERVER_LOF_PACKAGES_OEM.iso

IV   Windows Security

• When processing the encapsulation task, a large number of temporary files will be generated, and a large number of installation files will be released when installing the application in InBox Apps;

• Turning on Windows Security scans files and takes up a lot of CPU.

- In test: 1 hour and 22 minutes before shutdown, 20 minutes after shutdown.

How to close:

With the command line in green, hold down the Windows key and press R to launch Run.

1. Open Windows Security or run: windowsdefender:

2. Select "Virus & Threat Protection" or Run: windowsdefender://threat

3. Find "Virus & Threat Protection Settings", click "Manage Settings" or Run: windowsdefender://threatsettings, we recommend that you turn off some features:

   3.1. Real-time protection

   3.2. Cloud=delivered protection

   3.3. Automatic sample submission

   3.4. Tamper Protection

4. When you're not encapsulated, we recommend that you turn on Windows Security.

V   Command line

1. Optional "Terminal" or "PowerShell ISE", if "Terminal" is not installed, please go to: https://github.com/microsoft/terminal/releases After downloading;

2. Open "Terminal" or "PowerShell ISE" as administrator, it is recommended to set the PowerShell execution policy: bypass, PS command line:

   Set-ExecutionPolicy -ExecutionPolicy Bypass -Force

3. In this article, PS command line, green part, please copy it, paste it into the "Terminal" dialog box, press Enter and start running;

4. When there is .ps1, right-click the file and select Run with PowerShell, or copy the path and paste it into Terminal to run, the path with a colon, add the & character in the command line, example: & "D:\YiSolutions_Encapsulation_SIP.ps1"

B.   Language package: extract

II   Language pack: Ready

Mounted 20348.1.210507-1500.fe_release_amd64fre_SERVER_LOF_PACKAGES_OEM.iso or unzip it to any location;

III   Language pack: Extract scheme

1. Add

   1.1. Language name: Simplified Chinese - China, language tag: zh-CN, Scope of application: Install.Wim，Boot.Wim，WinRE.Wim

2. Delete

   2.1. Language name: English - United States, language tag: en-US, Scope of application: Install.Wim，Boot.Wim，WinRE.Wim

IV   Execute the extract command

- Auto = automatically search all local disks, default;

- Customize the path, for example, specify the E drive: $ISO = "E:\"

- Extract.ps1

  - \Expand\Extract.ps1

  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Extract.ps1

- Copy the code

```
$ISO = "Auto"

$SaveTo = "D:\OS_2022_Custom"

$Extract_language_Pack = @(

  @{ Tag = "zh-CN"; Act = "Add"; Scope = @( "Install\Install"; "Install\WinRE"; "Boot\Boot" ) }

  @{ Tag = "en-US"; Act = "Del"; Scope = @( "Install\Install"; "Install\WinRE"; "Boot\Boot" ) }

)

Function Extract_Language

{

  param( $Act, $NewLang, $Expand )

  Function Match_Required_Fonts

  {

    param( $Lang )

    $Fonts = @(

      @{ Match = @("as", "ar-SA", "ar", "ar-AE", "ar-BH", "ar-DJ", "ar-DZ", "ar-EG", "ar-ER", "ar-IL", "ar-IQ", "ar-JO", "ar-KM", "ar-KW", "ar-LB", "ar-LY",
"ar-MA", "ar-MR", "ar-OM", "ar-PS", "ar-QA", "ar-SD", "ar-SO", "ar-SS", "ar-SY", "ar-TD", "ar-TN", "ar-YE", "arz-Arab", "ckb-Arab", "fa", "fa-AF", "fa-IR",
"glk-Arab", "ha-Arab", "ks-Arab", "ks-Arab-IN", "ku-Arab", "ku-Arab-IQ", "mzn-Arab", "pa-Arab", "pa-Arab-PK", "pnb-Arab", "prs", "prs-AF", "prs-Arab",
"ps", "ps-AF", "sd-Arab", "sd-Arab-PK", "tk-Arab", "ug", "ug-Arab", "ug-CN", "ur", "ur-IN", "ur-PK", "uz-Arab", "uz-Arab-AF"); Name = "Arab"; }

      @{ Match = @("bn-IN", "as-IN", "bn", "bn-BD", "bpy-Beng"); Name = "Beng"; }

      @{ Match = @("da-dk", "iu-Cans", "iu-Cans-CA"); Name = "Cans"; }

      @{ Match = @("chr-Cher-US", "chr-Cher"); Name = "Cher"; }

      @{ Match = @("hi-IN", "bh-Deva", "brx", "brx-Deva", "brx-IN", "hi", "ks-Deva", "mai", "mr", "mr-IN", "ne", "ne-IN", "ne-NP", "new-Deva", "pi-Deva",
"sa", "sa-Deva", "sa-IN"); Name = "Deva"; }

      @{ Match = @("am", "am-ET", "byn", "byn-ER", "byn-Ethi", "ti", "ti-ER", "ti-ET", "tig", "tig-ER", "tig-Ethi", "ve-Ethi", "wal", "wal-ET", "wal-Ethi");
Name = "Ethi"; }

      @{ Match = @("gu", "gu-IN"); Name = "Gujr"; }

      @{ Match = @("pa", "pa-IN", "pa-Guru"); Name = "Guru"; }

      @{ Match = @("zh-CN", "cmn-Hans", "gan-Hans", "hak-Hans", "wuu-Hans", "yue-Hans", "zh-gan-Hans", "zh-hak-Hans", "zh-Hans", "zh-SG", "zh-
wuu-Hans", "zh-yue-Hans"); Name = "Hans"; }

      @{ Match = @("zh-TW", "cmn-Hant", "hak-Hant", "lzh-Hant", "zh-hak-Hant", "zh-Hant", "zh-HK", "zh-lzh-Hant", "zh-MO", "zh-yue-Hant"); Name =
"Hant"; }
```

```powershell
      @{ Match = @("he", "he-IL", "yi"); Name = "Hebr"; }

      @{ Match = @("ja", "ja-JP"); Name = "Jpan"; }

      @{ Match = @("km", "km-KH"); Name = "Khmr"; }

      @{ Match = @("kn", "kn-IN"); Name = "Knda"; }

      @{ Match = @("ko", "ko-KR"); Name = "Kore"; }

      @{ Match = @("de-de", "lo", "lo-LA"); Name = "Laoo"; }

      @{ Match = @("ml", "ml-IN"); Name = "Mlym"; }

      @{ Match = @("or", "or-IN"); Name = "Orya"; }

      @{ Match = @("si", "si-LK"); Name = "Sinh"; }

      @{ Match = @("tr-tr", "arc-Syrc", "syr", "syr-SY", "syr-Syrc"); Name = "Syrc"; }

      @{ Match = @("ta", "ta-IN", "ta-LK", "ta-MY", "ta-SG"); Name = "Taml"; }

      @{ Match = @("te", "te-IN"); Name = "Telu"; }

      @{ Match = @("th", "th-TH"); Name = "Thai"; }

   )

   ForEach ($item in $Fonts) {

      if (($item.Match) -Contains $Lang) {

         return $item.Name

      }

   }

   return "Not_matched"

}

Function Match_Other_Region_Specific_Requirements

{

   param( $Lang )

   $RegionSpecific = @(

      @{ Match = @("zh-TW"); Name = "Taiwan"; }

   )

   ForEach ($item in $RegionSpecific) {

      if (($item.Match) -Contains $Lang) {

         return $item.Name

      }

   }

   return "Skip_specific_packages"

}
```

```powershell
Function Extract_Process

{

  param( $Package, $Name, $NewSaveTo )

  $NewSaveTo = "$($SaveTo)\$($NewSaveTo)\Language\$($Act)\$($NewLang)"

  New-Item -Path $NewSaveTo -ItemType Directory -ErrorAction SilentlyContinue | Out-Null

  if ($ISO -eq "Auto") {

    Get-PSDrive -PSProvider FileSystem -ErrorAction SilentlyContinue | ForEach-Object {

      ForEach ($item in $Package) {

        $TempFilePath = Join-Path -Path $_.Root -ChildPath $item -ErrorAction SilentlyContinue

        if (Test-Path $TempFilePath -PathType Leaf) {

          Write-host "`n  Find: " -NoNewLine; Write-host $TempFilePath -ForegroundColor Green

          Write-host "  Copy to: " -NoNewLine; Write-host $NewSaveTo

          Copy-Item -Path $TempFilePath -Destination $NewSaveTo -Force

        }

      }

    }

  } else {

    ForEach ($item in $Package) {

      $TempFilePath = Join-Path -Path $ISO -ChildPath $item -ErrorAction SilentlyContinue

      Write-host "`n  Find: " -NoNewline; Write-host $TempFilePath -ForegroundColor Green

      if (Test-Path $TempFilePath -PathType Leaf) {

        Write-host "  Copy to: " -NoNewLine; Write-host $NewSaveTo

        Copy-Item -Path $TempFilePath -Destination $NewSaveTo -Force

      } else {

        Write-host "  Not found"

      }

    }

  }

  Write-host "`n  Verify the language pack file"

  ForEach ($item in $Package) {

    $Path = "$($NewSaveTo)\$([IO.Path]::GetFileName($item))"

    if (Test-Path $Path -PathType Leaf) {

      Write-host "  Discover: " -NoNewLine; Write-host $Path -ForegroundColor Green

    } else {
```

```powershell
                    Write-host "   Not found: " -NoNewLine; Write-host $Path -ForegroundColor Red

                }

            }

        }

        $AdvLanguage = @(

            @{

                Path = "Install\Install"

                Rule = @(

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-LanguageFeatures-Fonts-{DiyLang}-Package~31bf3856ad364e35~amd64~~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-Server-Language-Pack_x64_{Lang}.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-LanguageFeatures-Basic-{Lang}-Package~31bf3856ad364e35~amd64~~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-LanguageFeatures-Handwriting-{Lang}-Package~31bf3856ad364e35~amd64~~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-LanguageFeatures-OCR-{Lang}-Package~31bf3856ad364e35~amd64~~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-LanguageFeatures-Speech-{Lang}-Package~31bf3856ad364e35~amd64~~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-LanguageFeatures-TextToSpeech-{Lang}-Package~31bf3856ad364e35~amd64~~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-MSPaint-FoD-Package~31bf3856ad364e35~amd64~{Lang}~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-MSPaint-FoD-Package~31bf3856ad364e35~wow64~{Lang}~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-Notepad-FoD-Package~31bf3856ad364e35~amd64~{Lang}~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-Notepad-FoD-Package~31bf3856ad364e35~wow64~{Lang}~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~{Lang}~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~{Lang}~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~{Lang}~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~{Lang}~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~{Lang}~.cab"

                    "LanguagesAndOptionalFeatures\Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~{Lang}~.cab"

                )

            }

            @{

                Path = "Install\WinRE"

                Rule = @(

                    "Windows Preinstallation Environment\x64\WinPE_OCs\WinPE-FontSupport-{Lang}.cab"

                    "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\lp.cab"

                    "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-securestartup_{Lang}.cab"

                    "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-atbroker_{Lang}.cab"
```

```
        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-audiocore_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-audiodrivers_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-enhancedstorage_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-narrator_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-scripting_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-speech-tts_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-srh_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-srt_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-wds-tools_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-wmi_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-appxpackaging_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-storagewmi_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-wifi_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-rejuv_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-opcservices_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-hta_{Lang}.cab"

    )
}
@{

    Path = "Boot\Boot"

    Rule = @(

        "Windows Preinstallation Environment\x64\WinPE_OCs\WinPE-FontSupport-{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\lp.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\WinPE-Setup_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\WINPE-SETUP-Server_{Lang}.CAB"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-securestartup_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-atbroker_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-audiocore_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-audiodrivers_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-enhancedstorage_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-narrator_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-scripting_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-speech-tts_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-srh_{Lang}.cab"
```

```
                    "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-srt_{Lang}.cab"

                    "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-wds-tools_{Lang}.cab"

                    "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-wmi_{Lang}.cab"

                )

            }

        )

        $NewFonts = Match_Required_Fonts -Lang $NewLang

        $SpecificPackage = Match_Other_Region_Specific_Requirements -Lang $NewLang

        Foreach ($item in $Expand) {

            $Language = @()

            Foreach ($itemList in $AdvLanguage) {

                if ($itemList.Path -eq $item) {

                    Foreach ($PrintLang in $itemList.Rule) {

                        $Language += "$($PrintLang)".Replace("{Lang}", $NewLang).Replace("{DiyLang}", $NewFonts).Replace("{Specific}", $SpecificPackage)

                    }

                    Extract_Process -NewSaveTo $itemList.Path -Package $Language -Name $item

                }

            }

        }

    }

    ForEach ($item in $Extract_language_Pack) { Extract_Language -Act $item.Act -NewLang $item.Tag -Expand $item.Scope }
```

C.   Custom encapsulation

II   Custom encapsulation: Install.wim

1.   View Install.wim details

Image name, image description, image size, architecture, version, index number, etc.;

```
$ViewFile = "D:\OS_2022\Sources\Install.wim"

Get-WindowsImage -ImagePath $ViewFile | Foreach-Object { Get-WindowsImage -ImagePath $ViewFile -index $_.ImageIndex }
```

CYCLIC OPERATION AREA, START,

2.   Specify the path to mount install.wim

```
New-Item -Path "D:\OS_2022_Custom\Install\Install\Mount" -ItemType directory -ea SilentlyContinue
```

3.   Start mounting Install.wim

Default index number: 1

Mount-WindowsImage -ImagePath "D:\OS_2022\sources\install.wim" -Index "1" -Path "D:\OS_2022_Custom\Install\Install\Mount"

3.1.   Custom encapsulation: WinRE.wim

WARNING:

- WinRE.wim is a file within the Install.wim image;

- When Install.wim has multiple index numbers, only process any WinRE.wim;

- Synchronizing to all index numbers reduces the Install.wim volume, Learn "How to bulk replace WinRE.wim in all index numbers in Install.wim".

3.1.1.   View WinRE.wim details

Image name, image description, image size, architecture, version, index number, etc.;

$ViewFile = "D:\OS_2022_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

Get-WindowsImage -ImagePath $ViewFile | Foreach-Object { Get-WindowsImage -ImagePath $ViewFile -index $_.ImageIndex }

3.1.2.   Specify the path to mount WinRE.wim

New-Item -Path "D:\OS_2022_Custom\Install\WinRE\Mount" -ItemType directory -ea SilentlyContinue

3.1.3.   Start mounting WinRE.wim

Default index number: 1

$FileName = "D:\OS_2022_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

Mount-WindowsImage -ImagePath $FileName -Index "1" -Path "D:\OS_2022_Custom\Install\WinRE\Mount"

3.1.4.   Language pack

- Automatically install language packs: Get "Component: All installed packages in the image" and match them. After matching the corresponding names, install the local corresponding language pack files, View the report "Language installation package for WinRE.wim".

- When adding languages, different schema versions must be corresponded, and if not, errors are reported during the addition process.

3.1.4.1.   Language pack: add

- WinRE.Instl.lang.ps1

    o   \Expand\Install\WinRE\WinRE.Instl.lang.ps1

    o   https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Install/WinRE/WinRE.Instl.lang.ps1

- Copy the code

```powershell
$Mount = "D:\OS_2022_Custom\Install\WinRE\Mount"

$Sources = "D:\OS_2022_Custom\Install\WinRE\Language\Add\zh-CN"

$Initl_install_Language_Component = @()

Get-WindowsPackage -Path $Mount | ForEach-Object {

  $Initl_install_Language_Component += $_.PackageName

}

Add-WindowsPackage -Path $Mount -PackagePath "$($Sources)\WinPE-FontSupport-zh-CN.cab"

$Language_List = @(

  @{ Match = "*WinPE-LanguagePack-Package*"; File = "lp.cab"; }

  @{ Match = "*SecureStartup*"; File = "winpe-securestartup_zh-CN.cab"; }

  @{ Match = "*ATBroker*"; File = "winpe-atbroker_zh-CN.cab"; }

  @{ Match = "*AudioCore*"; File = "winpe-audiocore_zh-CN.cab"; }

  @{ Match = "*AudioDrivers*"; File = "winpe-audiodrivers_zh-CN.cab"; }

  @{ Match = "*EnhancedStorage*"; File = "winpe-enhancedstorage_zh-CN.cab"; }

  @{ Match = "*Narrator*"; File = "winpe-narrator_zh-CN.cab"; }

  @{ Match = "*scripting*"; File = "winpe-scripting_zh-CN.cab"; }

  @{ Match = "*Speech-TTS*"; File = "winpe-speech-tts_zh-CN.cab"; }

  @{ Match = "*srh*"; File = "winpe-srh_zh-CN.cab"; }

  @{ Match = "*srt*"; File = "winpe-srt_zh-CN.cab"; }

  @{ Match = "*wds-tools*"; File = "winpe-wds-tools_zh-CN.cab"; }

  @{ Match = "*-WMI-Package*"; File = "winpe-wmi_zh-CN.cab"; }

  @{ Match = "*WinPE-AppxPackaging*"; File = "winpe-appxpackaging_zh-CN.cab"; }

  @{ Match = "*StorageWMI*"; File = "winpe-storagewmi_zh-CN.cab"; }

  @{ Match = "*WiFi*"; File = "winpe-wifi_zh-CN.cab"; }

  @{ Match = "*rejuv*"; File = "winpe-rejuv_zh-CN.cab"; }

  @{ Match = "*opcservices*"; File = "winpe-opcservices_zh-CN.cab"; }

  @{ Match = "*hta*"; File = "winpe-hta_zh-CN.cab"; }

)

ForEach ($Rule in $Language_List) {

  Write-host "`n  Rule name: $($Rule.Match)" -ForegroundColor Yellow; Write-host "  $('-' * 80)"

  ForEach ($Component in $Initl_install_Language_Component) {

    if ($Component -like "*$($Rule.Match)*") {

      Write-host "  Component name: " -NoNewline

      Write-host $Component -ForegroundColor Green
```

```
                    Write-host "   Language pack file: " -NoNewline

                    Write-host "$($Sources)\$($Rule.File)" -ForegroundColor Green

                    Write-Host "   Installing ".PadRight(22) -NoNewline

                    try {

                      Add-WindowsPackage -Path $Mount -PackagePath "$($Sources)\$($Rule.File)" | Out-Null

                      Write-host "Finish" -ForegroundColor Green

                    } catch {

                      Write-host "Failed" -ForegroundColor Red

                    }

                    break

                  }

                }

              }
```

3.1.4.2.   Components: All packages installed in the image

3.1.4.2.1.   View

```
Get-WindowsPackage -Path "D:\OS_2022_Custom\Install\WinRE\Mount" | Out-GridView
```

3.1.4.2.2.   Export to Csv

```
$SaveTo = "D:\OS_2022_Custom\Install\WinRE\Report.$(Get-Date -Format
"yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS_2022_Custom\Install\WinRE\Mount" | Export-CSV -
NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green
```

3.1.5.   Save image: WinRE.wim

```
Save-WindowsImage -Path "D:\OS_2022_Custom\Install\WinRE\Mount"
```

3.1.6.   Unmount image: WinRE.wim

Close any applications that may be accessing files in the image, including File Explorer.

```
Dismount-WindowsImage -Path "D:\OS_2022_Custom\Install\WinRE\Mount" -Discard
```

3.1.7.   After rebuilding WinRE.wim, the file size can be reduced

- WinRE.Rebuild.ps1

   o   \Expand\Install\WinRE\WinRE.Rebuild.ps1

   o   https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Install/WinRE/WinRE.Rebuild.ps1

- Copy the code

```
$FileName = "D:\OS_2022_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

Get-WindowsImage -ImagePath $Filename -ErrorAction SilentlyContinue | ForEach-Object {

  Write-Host "   Image name: " -NoNewline

  Write-Host $_.ImageName -ForegroundColor Yellow

  Write-Host "   The index number: " -NoNewline

  Write-Host $_.ImageIndex -ForegroundColor Yellow

  Write-Host "`n   Rebuilding ".PadRight(28) -NoNewline

  Export-WindowsImage -SourceImagePath $Filename -SourceIndex $_.ImageIndex -DestinationImagePath "$($FileName).New" -CompressionType max

  Write-Host "Finish`n" -ForegroundColor Green

}

if (Test-Path "$($FileName).New" -PathType Leaf) {

  Remove-Item -Path $Filename

  Move-Item -Path "$($FileName).New" -Destination $Filename

  Write-Host "Finish" -ForegroundColor Green

} else {

  Write-host "Failed" -ForegroundColor Red

}
```

3.1.8.   Backup WinRE.wim

- WinRE.Backup.ps1

  o \Expand\Install\WinRE\WinRE.Backup.ps1

  o https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Install/WinRE/WinRE.Backup.ps1

- Copy the code

```
$WimLibPath = "D:\OS_2022_Custom\Install\Install\Update\Winlib"

$FileName = "D:\OS_2022_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

New-Item -Path $WimLibPath -ItemType Directory -ea SilentlyContinue

Copy-Item -Path $FileName -Destination $WimLibPath -Force
```

3.1.9.   Replace WinRE.wim within the Install.wim image

- After each installation of Install.wim, use item "Replace the WinRE.wim";

- Learning "After obtaining all the index numbers of Install.wim and replace the old WinRE.wim".

4. Language pack

- Automatically install language packs: Get "Component: All installed packages in the image" and match them. After matching the corresponding names, install the local corresponding language pack files, View the report "Language installation package for Install.wim".

- When adding languages, different schema versions must be corresponded, and if not, errors are reported during the addition process.

4.1. Language pack: add

- Install.Instl.lang.ps1

  o \Expand\Install\Install.Instl.lang.ps1

  o https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Install/Install.Instl.lang.ps1

- Copy the code

```
Function Language_Install
{
  param($Mount, $Sources, $Lang)

  $Initl_install_Language_Component = @()

  if (Test-Path $Mount -PathType Container) {

    Get-WindowsPackage -Path $Mount | ForEach-Object { $Initl_install_Language_Component += $_.PackageName }

  } else {

    Write-Host "Not mounted: $($Mount)"

    return

  }

  $Script:Init_Folder_All_File = @()

  if (Test-Path "$($Sources)\$($Lang)" -PathType Container) {

    Get-ChildItem -Path $Sources -Recurse -Include "*.cab" -ErrorAction SilentlyContinue | ForEach-Object {

      $Script:Init_Folder_All_File += $_.FullName

    }

    Write-host "`n  Available language pack installation files"

    if ($Script:Init_Folder_All_File.Count -gt 0 ) {

      ForEach ($item in $Script:Init_Folder_All_File) {

        Write-host "  $($item)"

      }

    } else {

      Write-host "There are no language pack files locally"
```

```powershell
            return

        }

    } else {

        Write-Host "Path does not exist: $($Sources)\$($Lang)"

        return

    }

    $Script:Init_Folder_All_File_Match_Done = @()

    $Script:Init_Folder_All_File_Exclude = @()

    $Global:Search_File_Order = @(

        @{

            Name = "Fonts"

            Description = "Fonts"

            Rule = @(

                @{ Match_Name = "*Fonts*"; IsMatch = "No"; Capability = ""; }

            )

        }

        @{

            Name = "Basic"

            Description = "Basic"

            Rule = @(

                @{ Match_Name = "*LanguageFeatures-Basic*"; IsMatch = "Yes"; Capability = "Language.Basic~~~lb-LU~0.0.1.0"; }

                @{ Match_Name = "*Server-LanguagePack-Package*"; IsMatch = "Yes"; Capability = "Language.Basic~~~lb-LU~0.0.1.0"; }

            )

        }

        @{

            Name = "OCR"

            Description = "Optical character recognition"

            Rule = @(

                @{ Match_Name = "*LanguageFeatures-OCR*"; IsMatch = "Yes"; Capability = "Language.OCR~~~fr-FR~0.0.1.0"; }

            )

        }

        @{

            Name = "Handwriting"

            Description = "Handwriting recognition"
```

```
        Rule = @(

            @{ Match_Name = "*LanguageFeatures-Handwriting*"; IsMatch = "Yes"; Capability = "Language.Handwriting~~~fr-
FR~0.0.1.0"; }

        )

    }

    @{

        Name = "TextToSpeech"

        Description = "Text-to-speech"

        Rule = @(

            @{ Match_Name = "*LanguageFeatures-TextToSpeech*"; IsMatch = "Yes"; Capability = "Language.TextToSpeech~~~fr-
FR~0.0.1.0"; }

        )

    }

    @{

        Name = "Speech"

        Description = "Speech recognition"

        Rule = @(

            @{ Match_Name = "*LanguageFeatures-Speech*"; IsMatch = "Yes"; Capability = "Language.Speech~~~fr-FR~0.0.1.0"; }

        )

    }

    @{

        Name = "RegionSpecific"

        Description = "Other region-specific requirements"

        Rule = @(

            @{ Match_Name = "*InternationalFeatures*zh-TW*"; IsMatch = "Yes"; Capability = ""; }

        )

    }

    @{

        Name = "Retail"

        Description = "Retail demo experience"

        Rule = @(

            @{ Match_Name = "*RetailDemo*"; IsMatch = "Yes"; Capability = ""; }

        )

    }

    @{
```

```
        Name = "Features_On_Demand"

        Description = "Features on demand"

        Rule = @(

          @{ Match_Name = "*MSPaint*amd64*"; IsMatch = "Yes"; Capability = "Microsoft.Windows.MSPaint~~~~0.0.1.0"; }

          @{ Match_Name = "*MSPaint*wow64*"; IsMatch = "Yes"; Capability = "Microsoft.Windows.MSPaint~~~~0.0.1.0"; }

          @{ Match_Name = "*Notepad*amd64*"; IsMatch = "Yes"; Capability = "Microsoft.Windows.Notepad~~~~0.0.1.0"; }

          @{ Match_Name = "*Notepad*wow64*"; IsMatch = "Yes"; Capability = "Microsoft.Windows.Notepad~~~~0.0.1.0"; }

          @{ Match_Name = "*PowerShell-ISE-FOD-Package*amd64*"; IsMatch = "Yes"; Capability =
"Microsoft.Windows.PowerShell.ISE~~~~0.0.1.0"; }

          @{ Match_Name = "*PowerShell-ISE-FOD-Package*wow64*"; IsMatch = "Yes"; Capability =
"Microsoft.Windows.PowerShell.ISE~~~~0.0.1.0"; }

          @{ Match_Name = "*StepsRecorder*amd64*"; IsMatch = "Yes"; Capability = "App.StepsRecorder~~~~0.0.1.0"; }

          @{ Match_Name = "*StepsRecorder*wow64*"; IsMatch = "Yes"; Capability = "App.StepsRecorder~~~~0.0.1.0"; }

          @{ Match_Name = "*WordPad*amd64*"; IsMatch = "Yes"; Capability = "Microsoft.Windows.WordPad~~~~0.0.1.0"; }

          @{ Match_Name = "*WordPad*wow64*"; IsMatch = "Yes"; Capability = "Microsoft.Windows.WordPad~~~~0.0.1.0"; }

        )

      }

    )

    ForEach ($item in $Global:Search_File_Order) {

      New-Variable -Scope global -Name "Init_File_Type_$($item.Name)" -Value @() -Force

    }

    ForEach ($WildCard in $Script:Init_Folder_All_File) {

      ForEach ($item in $Global:Search_File_Order) {

        ForEach ($TTT in $item.Rule) {

          if ($WildCard -like "*$($TTT.Match_Name)*") {

            Write-host "`n  Fuzzy matching: " -NoNewline; Write-host $TTT.Match_Name -ForegroundColor Green

            Write-host "  Language pack file: " -NoNewline; Write-host $WildCard -ForegroundColor Green

            $OSDefaultUser = (Get-Variable -Scope global -Name "Init_File_Type_$($item.Name)" -ErrorAction
SilentlyContinue).Value

            $TempSave = @{ Match_Name = $TTT.Match_Name; Capability = $TTT.Capability; FileName = $WildCard }

            $new = $OSDefaultUser + $TempSave

            if ($TTT.IsMatch -eq "Yes") {

              ForEach ($Component in $Initl_install_Language_Component) {

                if ($Component -like "*$($TTT.Match_Name)*") {

                  Write-host "  Component name: " -NoNewline; Write-host $Component -ForegroundColor Green

                  New-Variable -Scope global -Name "Init_File_Type_$($item.Name)" -Value $new -Force
```

```powershell
                    $Script:Init_Folder_All_File_Match_Done += $WildCard

                    break

                  }

                }

          } else {

             Write-host "   Do not match, install directly" -ForegroundColor Yellow

             New-Variable -Scope global -Name "Init_File_Type_$($item.Name)" -Value $new -Force

             $Script:Init_Folder_All_File_Match_Done += $WildCard

          }

        }

      }

   }

}

Write-host "`n  Grouping is complete, pending installation" -ForegroundColor Yellow

Write-host "  $('-' * 80)"

ForEach ($WildCard in $Global:Search_File_Order) {

   $OSDefaultUser = (Get-Variable -Scope global -Name "Init_File_Type_$($WildCard.Name)" -ErrorAction SilentlyContinue).Value

   Write-host "`n  $($WildCard.Description) ( $($OSDefaultUser.Count) item )"

   if ($OSDefaultUser.Count -gt 0) {

      ForEach ($item in $OSDefaultUser) {

         Write-host "   $($item.FileName)" -ForegroundColor Green

      }

   } else {

      Write-host "   Not available" -ForegroundColor Red

   }

}

Write-host "`n  Not matched, no longer installed" -ForegroundColor Yellow

Write-host "  $('-' * 80)"

ForEach ($item in $Script:Init_Folder_All_File) {

   if ($Script:Init_Folder_All_File_Match_Done -notcontains $item) {

      $Script:Init_Folder_All_File_Exclude += $item

      Write-host "  $($item)" -ForegroundColor Red

   }

}
```

```powershell
        Write-host "`n  Install" -ForegroundColor Yellow

      Write-host "   $('-' * 80)"

    ForEach ($WildCard in $Global:Search_File_Order) {

      $OSDefaultUser = (Get-Variable -Scope global -Name "Init_File_Type_$($WildCard.Name)" -ErrorAction SilentlyContinue).Value

      Write-host "`n   $($WildCard.Description) ( $($OSDefaultUser.Count) item )"; Write-host "   $('-' * 80)"

      if ($OSDefaultUser.Count -gt 0) {

        ForEach ($item in $OSDefaultUser) {

          Write-host "   Language pack file: " -NoNewline; Write-host $item.FileName -ForegroundColor Green

          Write-Host "   Installing ".PadRight(22) -NoNewline

          if (Test-Path $item.FileName -PathType Leaf) {

            try {

              Add-WindowsPackage -Path $Mount -PackagePath $item.FileName | Out-Null

              Write-host "Finish`n" -ForegroundColor Green

            } catch {

              Write-host "Failed" -ForegroundColor Red

              Write-host "   $($_)" -ForegroundColor Red

            }

          } else {

            Write-host "Does not exist`n"

          }

        }

      } else {

        Write-host "   Not available`n" -ForegroundColor Red

      }

    }

}

Language_Install -Mount "D:\OS_2022_Custom\Install\Install\Mount" -Sources "D:\OS_2022_Custom\Install\Install\Language\Add"
-Lang "zh-CN"
```

4.2.   Components: All packages installed in the image

4.2.1.   View

```powershell
        Get-WindowsPackage -Path "D:\OS_2022_Custom\Install\Install\Mount" | Out-GridView
```

4.2.2.   Export to Csv

```powershell
        $SaveTo = "D:\OS_2022_Custom\Install\Install\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"
```

```
Get-WindowsPackage -Path "D:\OS_2022_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green
```

5. Cumulative updates

   5.1. Download

   Check the "Windows Server 2022 Update History ", for example, install the cumulative update: KB5030216

   Go to the download page: https://www.catalog.update.microsoft.com/Search.aspx?q=Kb5030216 Or "Direct download" (If you cannot download, please go to the download page), save to

   D:\OS_2022_Custom\Install\Install\Update\windows10.0-kb5030216-x64_cbe587155f9818548b75f65d5cd41d341ed2fc61.msu

   5.2. Add

   ```
   $KBPath = "D:\OS_2022_Custom\Install\Install\Update\windows10.0-kb5030216-
   x64_cbe587155f9818548b75f65d5cd41d341ed2fc61.msu"

   Add-WindowsPackage -Path "D:\OS_2022_Custom\Install\Install\Mount" -PackagePath $KBPath
   ```

   5.3. Solid update

   It cannot be uninstalled after curing, which cleans the recovery image and resets the basis of any superseded components.

   ```
   Dism /Image:"D:\OS_2022_Custom\Install\Install\Mount" /cleanup-image /StartComponentCleanup /ResetBase
   ```

      5.3.1. Clean up components after curing updates

         • Install.Update.Curing.ps1

            o \Expand\Install\Install.Update.Curing.ps1

            o https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Install/Install.Update.Curing.ps1

         • Copy the code

         ```
         $Mount = "D:\OS_2022_Custom\Install\Install\Mount"

         Get-WindowsPackage -Path $Mount -ErrorAction SilentlyContinue | ForEach-Object {

           if ($_.PackageState -eq "Superseded") {

             Write-Host "   $($_.PackageName)" -ForegroundColor Green

             Remove-WindowsPackage -Path $Mount -PackageName $_.PackageName | Out-Null

           }

         }
         ```

6. Deployment engine: Add

   • Learn "Deployment Engine", if added to ISO installation media, can skip adding to mounted.

   • After adding the deployment engine, continue at the current location.

7. Health

Check whether there is any damage before saving. When the health status is abnormal, abort saving

Repair-WindowsImage -Path "D:\OS_2022_Custom\Install\Install\Mount" -ScanHealth

8. Replace the WinRE.wim

WinRE.wim in all index numbers in Install.wim has been replaced in batches. Please skip this step.

$WinRE = "D:\OS_2022_Custom\Install\Install\Update\Winlib\WinRE.wim"

$CopyTo = "D:\OS_2022_Custom\Install\Install\Mount\Windows\System32\Recovery"

Copy-Item -Path $WinRE -Destination $CopyTo -Force

9. Save image: Install.wim

Save-WindowsImage -Path "D:\OS_2022_Custom\Install\Install\Mount"

10. Unmount image: Install.wim

Close any applications that may be accessing files in the image, including File Explorer.

Dismount-WindowsImage -Path "D:\OS_2022_Custom\Install\Install\Mount" -Discard

CYCLIC OPERATION AREA, END.

11. How to batch replace WinRE.wim in all index numbers in Install.wim

11.1. Get WimLib

After going to the official website of https://wimlib.net, select a different version: arm64, x64, x86, and extract it to: D:Wimlib after downloading.

11.2. How to extract and update WinRE.wim in Install.wim

11.2.1. Extract the WinRE.wim file from Install.wim

- Install.WinRE.Extract.ps1

    o \Expand\Install\Install.WinRE.Extract.ps1

    o https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Install/Install.WinRE.Extract.ps1

- Copy the code

$Arguments = @(

  "extract",

  "D:\OS_2022\sources\install.wim", "1",

  "\Windows\System32\Recovery\Winre.wim",

  "--dest-dir=""D:\OS_2022_Custom\Install\Install\Update\Winlib"""

)

New-Item -Path "D:\OS_2022_Custom\Install\Install\Update\Winlib" -ItemType Directory -ea SilentlyContinue

Start-Process -FilePath "d:\wimlib\wimlib-imagex.exe" -ArgumentList $Arguments -wait -nonewwindow

11.2.2.   Get all index numbers of Install.wim and replace the old WinRE.wim

- Install.WinRE.Replace.wim.ps1

  o   \Expand\Install\Install.WinRE.Replace.wim.ps1

  o   https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Install/Install.WinRE.Replace.wim.ps1

- Copy the code

```
Get-WindowsImage -ImagePath "D:\OS_2022\sources\install.wim" -ErrorAction SilentlyContinue | ForEach-Object {

    Write-Host "  Image name: " -NoNewline

    Write-Host $_.ImageName -ForegroundColor Yellow

    Write-Host "  The index number: " -NoNewline

    Write-Host $_.ImageIndex -ForegroundColor Yellow

    Write-Host "`n  Replacement "

    $Arguments = @(

      "update",

      "D:\OS_2022\sources\install.wim",

      $_.ImageIndex,

      "--command=""add 'D:\OS_2022_Custom\Install\Install\Update\Winlib\WinRE.wim'
'\Windows\System32\Recovery\WinRe.wim'"""

    )

    Start-Process -FilePath "d:\wimlib\wimlib-imagex.exe" -ArgumentList $Arguments -wait -nonewwindow

    Write-Host "  Finish`n" -ForegroundColor Green

}
```

12.   Rebuilding Install.wim reduces file size

- Install.Rebuild.wim.ps1

  o   \Expand\Install\Install.Rebuild.wim.ps1

  o   https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Install/Install.Rebuild.wim.ps1

- Copy the code

```
$InstallWim = "D:\OS_2022\sources\install.wim"

Get-WindowsImage -ImagePath $InstallWim -ErrorAction SilentlyContinue | ForEach-Object {
```

```
      Write-Host "  Image name: " -NoNewline

      Write-Host $_.ImageName -ForegroundColor Yellow

      Write-Host "  The index number: " -NoNewline

      Write-Host $_.ImageIndex -ForegroundColor Yellow

      Write-Host "`n  Rebuilding".PadRight(28) -NoNewline

      Export-WindowsImage -SourceImagePath $InstallWim -SourceIndex $_.ImageIndex -DestinationImagePath "$($InstallWim).New" -
CompressionType max | Out-Null

      Write-Host "Finish`n" -ForegroundColor Green

    }

    if (Test-Path "$($InstallWim).New" -PathType Leaf) {

      Remove-Item -Path $InstallWim

      Move-Item -Path "$($InstallWim).New" -Destination $InstallWim

      Write-Host "Finish" -ForegroundColor Green

    } else {

      Write-host "Failed" -ForegroundColor Red

    }
```

III   Custom encapsulation: boot.wim

   1.   View Boot.wim details

           Image name, image description, image size, architecture, version, index number, etc.;

           ```
           $ViewFile = "D:\OS_2022\Sources\Boot.wim"

           Get-WindowsImage -ImagePath $ViewFile | Foreach-Object { Get-WindowsImage -ImagePath $ViewFile -index $_.ImageIndex }
           ```

   2.   Specify the path to mount Boot.wim

           ```
           New-Item -Path "D:\OS_2022_Custom\Boot\Boot\Mount" -ItemType directory -ea SilentlyContinue
           ```

   3.   Start mounting Boot.wim

           Default index number: 2

           ```
           Mount-WindowsImage -ImagePath "D:\OS_2022\sources\boot.wim" -Index "2" -Path "D:\OS_2022_Custom\Boot\Boot\Mount"
           ```

   4.   Language pack

           • Automatically install language packs: Get "Component: All installed packages in the image" and match them. After matching the
             corresponding names, install the local corresponding language pack files, View the report "Language installation package for Boot.wim".

           • When adding languages, different schema versions must be corresponded, and if not, errors are reported during the addition process.

           4.1.   Language pack: add

- Boot.Instl.lang.ps1

  -

  -

- Copy the code

```powershell
$Mount = "D:\OS_2022_Custom\Boot\Boot\Mount"

$Sources = "D:\OS_2022_Custom\Boot\Boot\Language\Add\zh-CN"

$Initl_install_Language_Component = @()

Get-WindowsPackage -Path $Mount | ForEach-Object {

  $Initl_install_Language_Component += $_.PackageName

}

Add-WindowsPackage -Path $Mount -PackagePath "$($Sources)\WinPE-FontSupport-zh-CN.cab"

$Language = @(

  @{ Match = "*WinPE*Setup*Server*Package*"; File = "WINPE-SETUP-Server_zh-CN.CAB"; }

  @{ Match = "*WinPE*Setup*Package*"; File = "WinPE-Setup_zh-CN.cab"; }

  @{ Match = "*WinPE-LanguagePack-Package*"; File = "lp.cab"; }

  @{ Match = "*SecureStartup*"; File = "winpe-securestartup_zh-CN.cab"; }

  @{ Match = "*ATBroker*"; File = "winpe-atbroker_zh-CN.cab"; }

  @{ Match = "*AudioCore*"; File = "winpe-audiocore_zh-CN.cab"; }

  @{ Match = "*AudioDrivers*"; File = "winpe-audiodrivers_zh-CN.cab"; }

  @{ Match = "*EnhancedStorage*"; File = "winpe-enhancedstorage_zh-CN.cab"; }

  @{ Match = "*Narrator*"; File = "winpe-narrator_zh-CN.cab"; }

  @{ Match = "*scripting*"; File = "winpe-scripting_zh-CN.cab"; }

  @{ Match = "*Speech-TTS*"; File = "winpe-speech-tts_zh-CN.cab"; }

  @{ Match = "*srh*"; File = "winpe-srh_zh-CN.cab"; }

  @{ Match = "*srt*"; File = "winpe-srt_zh-CN.cab"; }

  @{ Match = "*wds-tools*"; File = "winpe-wds-tools_zh-CN.cab"; }

  @{ Match = "*-WMI-Package*"; File = "winpe-wmi_zh-CN.cab"; }

)

ForEach ($Rule in $Language) {

  Write-host "`n  Rule name: $($Rule.Match)" -ForegroundColor Yellow; Write-host "  $('-' * 80)"

  ForEach ($Component in $Initl_install_Language_Component) {

    if ($Component -like "*$($Rule.Match)*") {

      Write-host "  Component name: " -NoNewline

      Write-host $Component -ForegroundColor Green
```

```
            Write-host "   Language pack file: " -NoNewline

            Write-host "$($Sources)\$($Rule.File)" -ForegroundColor Green

            Write-Host "   Installing ".PadRight(22) -NoNewline

            try {

               Add-WindowsPackage -Path $Mount -PackagePath "$($Sources)\$($Rule.File)" | Out-Null

               Write-host "Finish" -ForegroundColor Green

            } catch {

               Write-host "Failed" -ForegroundColor Red

            }

            break

         }

      }

   }
```

4.2. Components: All packages installed in the image

4.2.1. View

```
Get-WindowsPackage -Path "D:\OS_2022_Custom\Boot\Boot\Mount" | Out-GridView
```

4.2.2. Export to Csv

```
$SaveTo = "D:\OS_2022_Custom\Boot\Boot\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS_2022_Custom\Boot\Boot\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green
```

4.3. Language packs: sync to ISO installer

```
Copy-Item -Path "D:\OS_2022_Custom\Boot\Boot\Mount\sources\zh-CN" -Destination "D:\OS_2022\sources\zh-CN" -Recurse -
Force
```

4.4. Regenerate Lang.ini

After regeneration, you can adjust the "Installation Interface", the order when selecting "Language", open lang.ini, the default preferred value = 3, non-default value = 2.

4.4.1. Regenerate the mounted directory lang.ini

Re-generated Lang.ini file location: D:\OS_2022_Custom\Boot\Boot\Mount\Sources\lang.ini

```
Dism /image:"D:\OS_2022_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS_2022_Custom\Boot\Boot\Mount"
```

4.4.2. After regenerating lang.ini, synchronize to the installer

Re-generated Lang.ini file location: D:\OS_2022\Sources\lang.ini

Dism /image:"D:\OS_2022_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS_2022"

5.   Save image: Boot.wim

Save-WindowsImage -Path "D:\OS_2022_Custom\Boot\Boot\Mount"

6.   Unmount image: Boot.wim

Close any applications that may be accessing files in the image, including File Explorer.

Dismount-WindowsImage -Path "D:\OS_2022_Custom\Boot\Boot\Mount" -Discard

IV   Deployment engine

- Learn about "Automatically Adding Languages Installed in Windows Systems", learn: https://github.com/ilikeyi/Multilingual, how to download:

  o   After entering the website, click "Code", "Download Compressed Package", and after the download is completed, you will get the main.zip compressed package file.

  o   Go to the https://github.com/ilikeyi/Multilingual/releases download page, select the available version: 1.1.1.1, select the download source code format: zip, and get the Multilingual-1.1.1.1.zip compressed package file after the download is completed;

- Unzip the downloaded main.zip or Multilingual-1.1.1.1.zip to: D:\Multilingual-1.1.1.1, and rename: D:\Multilingual

- Learn "Unattended Windows Setup Reference", Intervene in the installation process by leaving it unattended.

1.   Add method

    1.1.   Add to ISO installation media

        1.1.1.   Unattended

            1.1.1.1.   Add to: [ISO]:\Autounattend.xml

                Autounattend.xml interferes with the WinPE installer when booting an ISO installation.

                Copy D:\Multilingual_Learn\Unattend\Mul.Unattend.xml to D:\OS_2022\Autounattend.xml

                Copy-Item "D:\Multilingual_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS_2022\Autounattend.xml" -Force

            1.1.1.2.   Add to: [ISO]:\Sources\Unattend.xml

                When mounting or unpacking an ISO, after running the [ISO]:\Setup.exe installer, [ISO]:\Sources\Unattend.xml will intervene in the installation process.

                Copy D:\Multilingual_Learn\Unattend\Mul.Unattend.xml to D:\OS_2022\Sources\Unattend.xml

                Copy-Item "D:\Multilingual_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS_2022\Sources\Unattend.xml" -Force

            1.1.1.3.   Add to: [ISO]:\sources\$OEM$\$$\Panther\unattend.xml

                Copy it to the system disk during the installation process, copy to: {system disk}:\Windows\Panther\unattend.xml

1.1.1.3.1. Create OEM path

New-Item -Path "D:\OS_2022\sources\`$OEM$\`$$\Panther" -ItemType Directory

1.1.1.3.2. Copy

Copy D:\Multilingual_Learn\Unattend\Mul.Unattend.xml to
D:\OS_2022\Sources\OEM\Panther\Unattend.xml

Copy-Item "D:\Multilingual_Learn\Unattend\Mul.Unattend.xml" -Destination
"D:\OS_2022\sources\`$OEM$\`$$\Panther\Unattend.xml" -Force

1.1.2. Deployment engine: add

Add "Automatically add installed languages for Windows systems" to D:\OS_2022\sources\$OEM$\$1\Yi\Engine in the
directory.

1.1.2.1. Deployment Engine: Copy

Copy D:\Multilingual\Engine to D:\OS_2022\Sources\$OEM$\$1\Yi\Engine

Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS_2022\sources\`$OEM$\`$1\Yi\Engine" -Recurse -
Force

1.1.2.2. Deployment engine: custom deployment tags

```
$Flag = @(

    "Is_Mark_Sync" # Allow global search and synchronization of deployment tags

    # Prerequisite deployment

#    "Auto_Update" # Allow automatic updates

#    "Use_UTF8" # Beta: Global language support using Unicode UTF-8

    "Disable_Network_Location_Wizard" # Network Location Wizard

    "Disable_Cleanup_Appx_Tasks" # Appx Cleanup and maintenance tasks

    "Disable_Cleanup_On_Demand_Language" # Prevent cleanup of unused on-demand feature language
packs

    "Disable_Cleanup_Unsed_Language"    # Prevent cleaning of unused language packs

    "Prerequisites_Reboot" # Restart your computer

    # Complete first deployment

#    "Popup_Engine" # Allow the deployment engine main interface to pop up for the first time

#    "Allow_First_Pre_Experience" # Allow first preview, as planned

    "Reset_Execution_Policy" # Restore PowerShell execution policy: Restricted

    "Clear_Solutions" # Delete the entire solution

    "Clear_Engine" # Delete the deployment engine and keep the others

#    "First_Experience_Reboot" # Restart your computer
```

```
  )

  ForEach ($item in $Flag) {

    Write-host "  $($item)" -ForegroundColor Green

    New-Item -Path "D:\OS_2022\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -
ErrorAction SilentlyContinue | Out-Null

    Out-File -FilePath "D:\OS_2022\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow\$($item)" -Encoding utf8 -
ErrorAction SilentlyContinue

  }
```

1.2. Add to mounted

Through "Custom encapsulation: Install.wim", execute "Start mounting Install.wim" and mount to:
D:\OS_2022_Custom\Install\Install\Mount

1.2.1. Unattended

Copy D:\Multilingual_Learn\Unattend\Mul.Unattend.xml to
D:\OS_2022_Custom\Install\Install\Mount\Panther\Unattend.xml

```
Copy-Item "D:\Multilingual\_Learn\Unattend\Mul.Unattend.xml" -Destination
"D:\OS_2022_Custom\Install\Install\Mount\Panther" -Force
```

1.2.2. Deployment engine: add

Add "Automatically add languages installed on Windows systems" to the
D:\OS_2022_Custom\Install\Install\Mount\Yi\Engine directory.

1.2.2.1. Deployment Engine: Copy

Copy D:\Multilingual\Engine to D:\OS_2022_Custom\Install\Install\Mount\Yi\Engine

```
Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS_2022_Custom\Install\Install\Mount\Yi\Engine" -
Recurse -Force
```

1.2.2.2. Deployment engine: custom deployment tags

```
$Flag = @(

  "Is_Mark_Sync" # Allow global search and synchronization of deployment tags

  # Prerequisite deployment

#   "Auto_Update" # Allow automatic updates

#   "Use_UTF8" # Beta: Global language support using Unicode UTF-8

  "Disable_Network_Location_Wizard" # Network Location Wizard

  "Disable_Cleanup_Appx_Tasks" # Appx Cleanup and maintenance tasks

  "Disable_Cleanup_On_Demand_Language" # Prevent cleanup of unused on-demand feature language
packs

  "Disable_Cleanup_Unsed_Language"    # Prevent cleaning of unused language packs
```

```
              "Prerequisites_Reboot" # Restart your computer

              # Complete first deployment

          #   "Popup_Engine" # Allow the deployment engine main interface to pop up for the first time

          #   "Allow_First_Pre_Experience" # Allow first preview, as planned

              "Reset_Execution_Policy" # Restore PowerShell execution policy: Restricted

              "Clear_Solutions" # Delete the entire solution

              "Clear_Engine" # Delete the deployment engine and keep the others

          #   "First_Experience_Reboot" # Restart your computer

          )

          ForEach ($item in $Flag) {

            Write-host "  $($item)" -ForegroundColor Green

            New-Item -Path "D:\OS_2022\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -
          ErrorAction SilentlyContinue | Out-Null

            Out-File -FilePath "D:\OS_2022\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow\$($item)" -Encoding utf8 -
          ErrorAction SilentlyContinue

          }
```

2.   Deployment Engine: Advanced

2.1.   Deployment engine: adding process

After copying the deployment engine, you can add deployment tags to intervene in the installation process.

2.2.   Unattended solution

When the customization is unattended, please modify it simultaneously if the following files exist:

- D:\OS_2022\Autounattend.xml

- D:\OS_2022\Sources\Unattend.xml

- D:\OS_2022\sources\$OEM$\$$\Panther\unattend.xml

- D:\OS_2022_Custom\Install\Install\Mount\Panther\Unattend.xml

2.2.1.   Multilingual or monolingual

In multi-language and monolingual, you can switch between each other. When replacing, please replace all the same ones in the file.

2.2.1.1.   Multi-language

```
<UILanguage>%OSDUILanguage%</UILanguage>

<InputLocale>%OSDInputLocale%</InputLocale>

<SystemLocale>%OSDSystemLocale%</SystemLocale>
```

```
<UILanguage>%OSDUILanguage%</UILanguage>

<UILanguageFallback>%OSDUILanguageFallback%</UILanguageFallback>

<UserLocale>%OSDUserLocale%</UserLocale>
```

2.2.1.2. Monolingual

A single language needs to specify a language tag, for example, specify a language tag: zh-CN

```
<UILanguage>zh-CN</UILanguage>

<InputLocale>zh-CN</InputLocale>

<SystemLocale>zh-CN</SystemLocale>

<UILanguage>zh-CN</UILanguage>

<UILanguageFallback>zh-CN</UILanguageFallback>

<UserLocale>zh-CN</UserLocale>
```

2.2.2. User plan

By default, the self-created user Administrator is used and logged in automatically. It can be switched by modifying the following configuration: self-created or customized user.

2.2.2.1. Self-created user Administrator

By default, the self-created user: Administrator is used and logged in automatically, inserted between <OOBE> and </OOBE>.

```
<UserAccounts>

  <LocalAccounts>

    <LocalAccount wcm:action="add">

      <Password>

        <Value></Value>

        <PlainText>true</PlainText>

      </Password>

      <Description>Administrator</Description>

      <DisplayName>Administrator</DisplayName>

      <Group>Administrators</Group>

      <Name>Administrator</Name>

    </LocalAccount>

  </LocalAccounts>

</UserAccounts>

<AutoLogon>

  <Password>
```

```
    <Value></Value>

      <PlainText>true</PlainText>

    </Password>

    <Enabled>true</Enabled>

    <Username>Administrator</Username>

  </AutoLogon>
```

          2.2.2.2.   Custom user

After setting up a custom user and installing the system, in OOBE, you can choose settings such as local and online users.

          2.2.2.3.   Delete

Username: Removed from start `<UserAccounts>` to `</UserAccounts>`

Autologin: Remove from start `<AutoLogon>` to `</AutoLogon>`

          2.2.2.4.   Replace

From the beginning `<OOBE>` to `</OOBE>`

```
<OOBE>

  <ProtectYourPC>3</ProtectYourPC>

  <HideEULAPage>true</HideEULAPage>

  <HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>

</OOBE>
```

D.   Generate ISO

1.   Download OScdimg

Select the Oscdimg version according to the architecture, and save it to: D:\ after downloading. To save in other paths, please enter the absolute path of OScdimg.exe;

       1.1.   x64

https://github.com/ilikeyi/solutions/raw/main/_Software/Oscdimg/amd64/oscdimg.exe

       1.2.   x86

https://github.com/ilikeyi/solutions/raw/main/_Software/Oscdimg/x86/oscdimg.exe

       1.3.   arm64

https://github.com/ilikeyi/solutions/raw/main/_Software/Oscdimg/arm64/oscdimg.exe

2. Use the oscdimg command line to generate an ISO file and save it to: D:\WS2022.iso

- ISO.ps1

  - \Expand\ISO.ps1

  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/ISO.ps1

- Copy the code

```
$Oscdimg = "D:\Oscdimg.exe"

$ISO = "D:\Win2022"

$Volume = "Win2022"

$SaveTo = "D:\Win2022.iso"

$Arguments = @("-m", "-o", "-u2", "-udfver102", "-l""$($Volume)""", "-bootdata:2#p0,e,b""$($ISO)\boot\etfsboot.com""#pEF,e,b""$($ISO)\efi\microsoft\boot\efisys.bin""", $ISO, $SaveTo)

Start-Process -FilePath $Oscdimg -ArgumentList $Arguments -wait -nonewwindow
```

Yɪ' Sᴏʟᴜᴛɪᴏɴs

This copy packaging tutorial is part of Yi's Solutions content, learn more:

- ▪ Yi's official website | https://fengyi.tel/solutions

- ▪ Github | https://github.com/ilikeyi/solutions

Author: Yi

EMail: 775159955@qq.com, ilikeyi@outlook.com

Document version: 1.2

Documentation model: Lite version

Translation: Chinese to English version

Updated: 2024 - 8

Suggestions or feedback: https://github.com/ilikeyi/solutions/issues