Microsoft Windows Server 2022

不同的系统版本，有着不同的封装方式，封装过程中包含："语言包：添加、关联、删除"、"驱动：添加、删除"、"累积更新：添加、删除"等。

在这背后藏着很多的隐藏故事，想解开这些，你准备好开始尝试封装了吗?

摘要

目录

章节 1　　封装

A.　　先决条件

II　　正在运行的系统

1.　　使用 DISM 命令制作高版本镜像时

正在运行的操作系统是 Windows 10 或者低于 Windows 11 24H2 时，在某些情况下使用 DISM 命令在制作高版本镜像时，会引发一些未知的问题，例如：在 Windows 10 操作系统里运行 DISM 命令来处理 Windows Server 2025 脱机映像时，在封装过程中也许会收到报错信息："此应用无法在你的电脑运行"，解决方法：

1.1.　　升级正在运行的操作系统或重新安装到更高的版本（建议）；

1.2.　　升级或安装新版 ADK 或 PowerShell 版本（不建议）

1.2.1.　　可尝试升级到最新 PowerShell 7 以上的版本；

1.2.2.　　安装最新版 ADK 后并替换 DISM 命令后能解决 DISM 版本低的问题，但是：封装脚本主要使用的命令行是 PowerShell 命令行，所以不建议你使用以上方法，最佳的方法是：升级正在运行的操作系统或重新安装到更高的版本。

2.　　磁盘分区

2.1.　　挂载脱机映像到 REFS 磁盘分区后，执行部分 DISM 命令会出现异常，使用 NTFS 格式的磁盘分区不受此影响。

2.2.　　ISO 解压后，所在的位置不受 REFS 分区影响。

3.　　Windows 安全中心

- 在处理封装任务时，将产生大量的临时文件，安装 InBox Apps 里的应用时会释放大量的安装文件；

- 开启 Windows 安全中心会扫描文件、会占用大量的 CPU。

- 测试中：未关闭前耗时 1 小时 22 分，关闭后耗时 20 分钟。

如何关闭：

绿色为命令行，按住 Windows 键并按 R 键启动"运行"。

3.1.　　打开"Windows 安全中心"或运行：windowsdefender:

3.2.　　选择"病毒和威胁防护"或运行：windowsdefender://threat

3.3.　　找到"病毒和威胁防护设置"，点击"管理设置"或运行：windowsdefender://threatsettings，建议您关闭部分功能：

3.3.1.　　实时保护

3.3.2.　　云提供的保护

3.3.3.　　自动提交样本

3.3.4. 篡改防护

3.4. 未处于封装时，建议您开启 Windows 安全中心。

4. 命令行

4.1. 可选"Terminal"或"PowerShell ISE"，未安装"Terminal"，请前往 https://github.com/microsoft/terminal/releases 后下载；

4.2. 以管理员身份打开"Terminal"或"PowerShell ISE"，设置 PowerShell 执行策略：绕过，PS 命令行：

Set-ExecutionPolicy -ExecutionPolicy Bypass -Force

4.3. 在本文中，绿色部分属于 PS 命令行，请复制后，粘贴到"Terminal"对话框，按回车键（Enter）后开始运行；

4.4. 有 .ps1 时，点击文件右键，选择以 PowerShell 运行，或复制路径，粘贴到"Terminal "或"PowerShell ISE"里运行，带冒号的路径，在命令行添加 & 字符，示例：& "D:\YiSolutions_Encapsulation_SIP.ps1"

II 要求

1. 系统安装包

1.1. 准备下载初始版本或开发者版本

1.1.1. x64

1.1.1.1. 文件名：en-us_windows_server_2022_x64_dvd_620d7eac.iso

文件列表：https://files.rg-adguard.net/file/9a0f4eb7-c3a9-e46b-3fc8-cdb71289dbfb

1.2. 示例下载 en-us_windows_server_2022_x64_dvd_620d7eac.iso 后，解压到：D:\en-us_windows_server_2022_x64_dvd_620d7eac

注意：解压到 D 盘前，你应该检查是否 ReFS 分区格式，如果是 ReFS 分区格式时：执行部分 DISM 命令将出现异常。解决方法：请使用 NTFS 格式的磁盘分区。

1.3. 解压完成后，将目录 D:\en-us_windows_server_2022_x64_dvd_620d7eac 更改为 D:\OS_2022

1.4. 所有脚本、所有路径，已默认设置为 D:\OS_2022 为映像来源。

2. 语言包

2.1. 学习

阅读时，请了解"蓝色"重要突出部分。

2.1.1. 将语言添加到 Windows 11 映像

2.1.2. 语言和区域按需功能 (FOD)

2.1.2.1. 字体

- 添加语言包时，触发了对应的区域时，需添加所需的字体功能，下载"所有可用语言 FOD 的列表"了解更多。

- 在"语言包：提取"时，已加入自动识别功能，可了解函数：Function Match_Required_Fonts

2.1.2.2. 区域关联

区域关联是什么？

- 映像语言仅英文版时，添加 zh-HK 语言包后，映像语言不会新增，应先安装 zh-TW 后，再安装 zh-HK 即可获得对应的关联。

- 可参阅微软官方原版：Windows 10、Windows 11 繁体版。

已知区域关联：

2.1.2.2.1. 区域：zh-TW，可选关联区域：zh-HK

2.2. 语言包：下载

2.2.1. 文件包：https://software-download.microsoft.com/download/sg/20348.1.210507-1500.fe_release_amd64fre_SERVER_LOF_PACKAGES_OEM.iso

文件列表：https://files.rg-adguard.net/file/f4a036a7-5c8e-6bd6-764a-83655c1a9ce5

B. 语言包：提取

II 语言包：准备

挂载 20348.1.210507-1500.fe_release_amd64fre_SERVER_LOF_PACKAGES_OEM.iso 或解压到任意位置；

III 语言包：提取方案

1. 添加

1.1. 语言名称：简体中文 - 中国，区域：zh-CN，适用范围：Install.Wim，Boot.Wim，WinRE.Wim

2. 删除

2.1. 语言名称：英语 - 美国，区域：en-US，适用范围：Install.Wim，Boot.Wim，WinRE.Wim

IV   执行提取命令

- Auto = 自动搜索本地所有磁盘，默认；

- 自定义路径，例如指定为 E 盘：$ISO = "E:\"

- Extract.ps1

  o   \Expand\Extract.ps1

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

- 复制代码

```
$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"; }
```

```powershell
    @{ 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"; }

    @{ 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"
```

```
}

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
```

```powershell
        } else {

            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.　自定义封装

II　自定义封装：Install.wim

1.　查看 Install.wim 详细信息

映像名称、映像描述、映像大小、架构、版本、索引号等；

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

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

循环操作区域，开始，

2.　指定挂载 Install.wim 路径

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

3. 开始挂载 Install.wim

默认索引号：1

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

处理 Install.wim 映像内的文件，可选项，开始，

3.1. 自定义封装：WinRE.wim

注意：

- WinRE.wim 属于 Install.wim 映像内的文件；

- Install.wim 有多个索引号时，仅处理任意一个 WinRE.wim 即可；

- 同步至所有索引号即可减少 Install.wim 体积；学习"如何批量替换 Install.wim 里的所有索引号里的 WinRE.wim"。

3.1.1. 查看 WinRE.wim 详细信息

映像名称、映像描述、映像大小、架构、版本、索引号等；

$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. 指定挂载 WinRE.wim 路径

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

3.1.3. 开始挂载 WinRE.wim

默认索引号：1

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

3.1.4. 语言包

- 自动安装语言包：获取"组件：映像中已安装的所有包"后进行匹配，匹配到对应的名称后，再安装本地对应的语言包文件。

- 添加语言时，须对应不同的架构版本，未对应时，添加过程中报错等提示。

3.1.4.1. 语言包：添加

- 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

- 复制代码

```
$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) {
```

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

        Write-host "  Component name: " -NoNewline

        Write-host $Component -ForegroundColor Green

        Write-host "  Language 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. 脱机映像语言：更改

3.1.4.2.1. 更改默认语言、区域设置和其他国际设置

区域：zh-CN

```
Dism /Image:"D:\OS_2022_Custom\Install\WinRE\Mount" /Set-AllIntl:zh-CN
```

3.1.4.2.2. 查看可用的语言设置

```
Dism /Image:"D:\OS_2022_Custom\Install\WinRE\Mount" /Get-Intl
```

3.1.4.3. 组件：映像中已安装的所有包

3.1.4.3.1. 查看

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

3.1.4.3.2. 导出到 Csv

```
$SaveTo = "D:\OS_2022_Custom\Install\WinRE\Report.Components.$(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.   累积更新

准备可用的累积更新文件，请更改示例文件名：KB_WinRE.cab

3.1.5.1.   添加

```
$KBPath = "D:\OS_2022_Custom\Install\WinRE\Update\KB_WinRE.cab"

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

3.1.5.2.   删除

```
$KBPath = "D:\OS_2022_Custom\Install\WinRE\Update\KB_WinRE.cab"

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

3.1.5.3.   固化更新

固化后不可卸载，固化时将清理恢复映像并重置任何被取代的组件的基础。

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

3.1.5.3.1.   固化更新后清理组件

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

  }

}
```

3.1.6.   驱动

3.1.7.   保存映像：WinRE.wim

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

3.1.8.   卸载映像：WinRE.wim

关闭所有可能正在访问映像中文件的应用程序，包括文件资源管理器。

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

Yi's Solutions

3.1.9. 重建 WinRE.wim 后，可缩小文件大小

- 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

- 复制代码

```
$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  Under reconstruction ".PadRight(28) -NoNewline

  try {

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

    Write-Host "Finish" -ForegroundColor Green

  } catch {

    Write-Host $_ -ForegroundColor Yellow

    Write-host $Failed -ForegroundColor Red

  }

}

Write-Host "`n  Rename: " -NoNewline -ForegroundColor Yellow

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

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

}
```

3.1.10. 备份 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.

- 复制代码

$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

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

3.1.11. 替换 Install.wim 映像内的 WinRE.wim

- 每次挂载 Install.wim 后"替换 WinRE.wim";

- 学习"获取 Install.wim 所有索引号后并替换旧的 WinRE.wim"。

处理 Install.wim 映像内的文件，结束。

4. 语言包

- 自动安装语言包：获取"组件：映像中已安装的所有包"后进行匹配，匹配到对应的名称后，再安装本地对应的语言包文件。

- 添加语言时，须对应不同的架构版本，未对应时，添加过程中报错等提示。

4.1. 语言包：添加

- 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

- 复制代码

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

$Sources = "D:\OS_2022_Custom\Install\Install\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)\Microsoft-Windows-LanguageFeatures-Fonts-Hans-Package~31bf3856ad364e35~amd64~~.cab"

$Language_List = @(

  @{ Match = "*Server-LanguagePack-Package*"; File = "Microsoft-Windows-Server-Language-Pack_x64_zh-CN.cab"; }

```
  @{ Match = "*LanguageFeatures-Basic*"; File = "Microsoft-Windows-LanguageFeatures-Basic-zh-CN-
Package~31bf3856ad364e35~amd64~~.cab"; }

  @{ Match = "*Handwriting*"; File = "Microsoft-Windows-LanguageFeatures-Handwriting-zh-CN-
Package~31bf3856ad364e35~amd64~~.cab"; }

  @{ Match = "*OCR*"; File = "Microsoft-Windows-LanguageFeatures-OCR-zh-CN-Package~31bf3856ad364e35~amd64~~.cab"; }

  @{ Match = "*LanguageFeatures-Speech*"; File = "Microsoft-Windows-LanguageFeatures-Speech-zh-CN-
Package~31bf3856ad364e35~amd64~~.cab"; }

  @{ Match = "*TextToSpeech*"; File = "Microsoft-Windows-LanguageFeatures-TextToSpeech-zh-CN-
Package~31bf3856ad364e35~amd64~~.cab"; }

  @{ Match = "*MSPaint*amd64*"; File = "Microsoft-Windows-MSPaint-FoD-Package~31bf3856ad364e35~amd64~zh-CN~.cab"; }

  @{ Match = "*MSPaint*wow64*"; File = "Microsoft-Windows-MSPaint-FoD-Package~31bf3856ad364e35~wow64~zh-CN~.cab"; }

  @{ Match = "*Notepad*amd64*"; File = "Microsoft-Windows-Notepad-FoD-Package~31bf3856ad364e35~amd64~zh-CN~.cab"; }

  @{ Match = "*Notepad*wow64*"; File = "Microsoft-Windows-Notepad-FoD-Package~31bf3856ad364e35~wow64~zh-CN~.cab"; }

  @{ Match = "*PowerShell*amd64*"; File = "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~zh-
CN~.cab"; }

  @{ Match = "*PowerShell*wow64*"; File = "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~zh-
CN~.cab"; }

  @{ Match = "*StepsRecorder*amd64*"; File = "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~zh-
CN~.cab"; }

  @{ Match = "*StepsRecorder*wow64*"; File = "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~zh-
CN~.cab"; }

  @{ Match = "*WordPad*amd64*"; File = "Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~zh-CN~.cab"; }

  @{ Match = "*WordPad*wow64*"; File = "Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~zh-CN~.cab"; }

  @{ Match = "*Client*LanguagePack*zh-TW*"; File = "Microsoft-Windows-InternationalFeatures-Taiwan-
Package~31bf3856ad364e35~amd64~~.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
```

```
            break

        } catch {

            Write-host "Failed" -ForegroundColor Red

        }

      }

    }

  }
```

4.2. 脱机映像语言：更改

- 从 Windows 11 开始，DISM 设置的默认系统 UI 语言在所有版本中保持不变（家庭版除外）。 对于所有商业版，在开箱即用体验 (OOBE) 期间选择的语言会设置为系统首选 UI 语言，Windows 将以此语言显示；对于家庭版，在 OOBE 期间选择的语言将继续用作默认系统 UI 语言。

- 从 Windows 10 版本 2004 开始，如果将基于 .appx 的语言体验包 (LXP) 支持的语言作为参数传递，则该语言将设置为系统首选 UI 语言，其父语言将设置为默认系统 UI 语言。 在以前的版本中，仅支持基于 .cab 的语言包。

4.2.1. 更改默认语言、区域设置和其他国际设置

区域：zh-CN

```
Dism /Image:"D:\OS_2022_Custom\Install\Install\Mount" /Set-AllIntl:zh-CN
```

4.2.2. 查看可用的语言设置

```
Dism /Image:"D:\OS_2022_Custom\Install\Install\Mount" /Get-Intl
```

4.3. 组件：映像中已安装的所有包

4.3.1. 查看

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

4.3.2. 导出到 Csv

```
$SaveTo = "D:\OS_2022_Custom\Install\Install\Report.Components.$(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. 累积更新

5.1. 下载

查阅"Windows Server 2022 更新历史"，例如安装累积更新：KB5030216

前往下载页面：https://www.catalog.update.microsoft.com/Search.aspx?q=Kb5030216 或 "直连下载"（无法下载时请进入下载页面），保存到：

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

5.2. 添加

```
$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. 固化更新

固化后不可卸载，固化时将清理恢复映像并重置任何被取代的组件的基础。

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

5.3.1. 固化更新后清理组件

```
$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. 驱动

7. 部署引擎：添加

- 了解"部署引擎"，如果添加到 ISO 安装介质，可跳过添加到已挂载。

- 如果添加部署引擎到已挂载里，请继续在当前位置执行下一步。

8. 健康

保存前应检查是否损坏，健康状态异常时，中止保存

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

9. 替换 WinRE.wim

已批量替换 Install.wim 里的所有索引号里的 WinRE.wim 请跳过该步骤。

$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

10. 保存映像

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

11. 卸载映像

关闭所有可能正在访问映像中文件的应用程序，包括文件资源管理器。

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

循环操作区域，结束。

12. 如何批量替换 Install.wim 里的所有索引号里的 WinRE.wim

12.1. 获取 WimLib

前往 https://wimlib.net 官方网站后，选择不同的版本： arm64, x64, x86，下载完成后解压到： D:\Wimlib

12.2. 如何在 Install.wim 里提取和更新 WinRE.wim

12.2.1. 从 Install.wim 里提取 WinRE.wim 文件 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

- 复制代码

$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

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

12.2.2. 获取 Install.wim 所有索引号后并替换旧的 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

- 复制代码

```
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

}
```

13. 重建 Install.wim 后可缩小文件大小

13.1. 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

- 复制代码

```
$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  Rebuild".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   自定义封装：boot.wim

1.   查看 Boot.wim 文件信息

映像名称、映像描述、映像大小、架构、版本、索引号等；

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

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

2.   指定挂载 Boot.wim 路径

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

3.   开始挂载 Boot.wim

默认索引号：2

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

4.   语言包

- 自动安装语言包：获取"组件：映像中已安装的所有包"后进行匹配，匹配到对应的名称后，再安装本地对应的语言包文件。

- 添加语言时，须对应不同的架构版本，未对应时，添加过程中报错等提示。

4.1.   语言包：添加

- Boot.Instl.lang.ps1

    o   \Expand\Boot\Boot.Instl.lang.ps1

- 复制代码

```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_zh*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 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. 脱机映像语言：更改

4.2.1. 更改默认语言、区域设置和其他国际设置

区域：zh-CN

```
Dism /Image:"D:\OS_2022_Custom\Boot\Boot\Mount" /Set-AllIntl:zh-CN
```

4.2.2. 查看可用的语言设置

```
Dism /Image:"D:\OS_2022_Custom\Boot\Boot\Mount" /Get-Intl
```

4.3. 组件：映像中已安装的所有包

4.3.1. 查看

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

4.3.2. 导出到 Csv

```
$SaveTo = "D:\OS_2022_Custom\Boot\Boot\Report.Components.$(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
```

4.4. 语言包：同步到 ISO 安装程序

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

4.5. 重新生成 Lang.ini

重新生成后，可调整"安装界面"，选择"语言"时的顺序，打开 lang.ini，默认首选值＝3，非默认值＝2。

4.5.1. 重新生成已挂载目录 lang.ini

重新生成的 Lang.ini 文件位置：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"

4.5.2. 重新生成 lang.ini 后，同步到安装程序

重新生成的 Lang.ini 文件位置：D:\OS_2022\Sources\lang.ini

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

5. 累积更新

准备可用的累积更新文件，请更改示例文件名：KB_Boot.cab

5.1. 添加

$KBPath = "D:\OS_2022_Custom\Boot\Boot\Update\KB_Boot.cab"

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

5.2. 删除

$KBPath = "D:\OS_2022_Custom\Boot\Boot\Update\KB_Boot.cab"

Remove-WindowsPackage -Path "D:\OS_2022_Custom\Boot\Boot\Mount" -PackagePath $KBPath

5.3. 固化更新

固化后不可卸载，固化时将清理恢复映像并重置任何被取代的组件的基础。

Dism /image:"D:\OS_2022_Custom\Boot\Boot\Mount" /cleanup-image /StartComponentCleanup /ResetBase

5.3.1. 固化更新后清理组件

```
$Mount = "D:\OS_2022_Custom\Boot\Boot\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. 驱动

7. 保存映像：Boot.wim

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

8. 卸载映像：Boot.wim

关闭所有可能正在访问映像中文件的应用程序，包括文件资源管理器。

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

IV   部署引擎

- 了解"自动添加 Windows 系统已安装的语言"，学习：https://github.com/ilikeyi/Multilingual，如何下载：

  ○ 进入网站后，点击 "代码"，"下载压缩包"，下载完成后得到 main.zip 压缩包文件。

  ○ 前往 https://github.com/ilikeyi/Multilingual/releases 下载页面，选择可用版本：1.1.1.1，选择下载源代码格式：zip，下载完成后得到
     Multilingual-1.1.1.1.zip 压缩包文件；

- 将已下载的 main.zip 或 Multilingual-1.1.1.1.zip，解压到：D:\Multilingual-1.1.1.1，重命名：D:\Multilingual

- 学习"无人值守 Windows 安装参考"，通过无人值守来干预安装过程。

1. 添加方式

   1.1.   添加到 ISO 安装介质

      1.1.1.   无人值守

         1.1.1.1.   添加到：[ISO]:\Autounattend.xml

                引导 ISO 安装时，Autounattend.xml 干预 WinPE 安装程序。

                复制 D:\Multilingual_Learn\Unattend\Mul.Unattend.xml 到 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.   添加到：[ISO]:\Sources\Unattend.xml

                挂载或解压 ISO 时，运行 [ISO]:\Setup.exe 安装程序后，[ISO]:\Sources\Unattend.xml 将干预安装过程。

                复制 D:\Multilingual_Learn\Unattend\Mul.Unattend.xml 到 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.   添加到：[ISO]:\sources\$OEM$\$$\Panther\unattend.xml

安装过程中复制到系统盘里，复制到：{系统盘}:\Windows\Panther\unattend.xml

1.1.1.3.1. 创建 OEM 路径

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

1.1.1.3.2. 复制

复制 D:\Multilingual_Learn\Unattend\Mul.Unattend.xml 到
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. 部署引擎：添加

添加"自动添加 Windows 系统已安装的语言"到 D:\OS_2022\sources\$OEM$\$1\Yi\Engine 目录里。

1.1.2.1. 部署引擎：复制

复制 D:\Multilingual\Engine 到 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. 部署引擎：自定义部署标记

```
$Flag = @(

    "Is_Mark_Sync" # 允许全盘搜索并同步部署标记

    # 先决部署

#   "Auto_Update" # 允许自动更新

#   "Use_UTF8" # Beta 版：使用 Unicode UTF-8 提供全球语言支持

    "Disable_Network_Location_Wizard" # 网络位置向导

    "Disable_Cleanup_Appx_Tasks" # Appx 清理维护任务

    "Disable_Cleanup_On_Demand_Language" # 阻止清理未使用的按需功能语言包

    "Disable_Cleanup_Unsed_Language" # 阻止清理未使用的语言包

    "Prerequisites_Reboot" # 重新启动计算机

    # 完成首次部署

#   "Popup_Engine" # 允许首次弹出部署引擎主界面
```

```
#    "Allow_First_Pre_Experience" # 允许首次预体验，按计划

     "Reset_Execution_Policy" # 恢复 PowerShell 执行策略：受限

     "Clear_Solutions" # 删除整个解决方案

     "Clear_Engine" # 删除部署引擎，保留其它

#    "First_Experience_Reboot" # 重新启动计算机

)

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. 添加到已挂载

通过"自定义封装：Install.wim"，执行"开始挂载 Install.wim"，挂载到：D:\OS_2022_Custom\Install\Install\Mount

1.2.1. 无人值守

复制 D:\Multilingual_Learn\Unattend\Mul.Unattend.xml 到 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. 部署引擎

添加"自动添加 Windows 系统已安装的语言"到 D:\OS_2022_Custom\Install\Install\Mount\Yi\Engine 目录里。

1.2.2.1. 部署引擎：复制

复制 D:\Multilingual\Engine 到 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. 部署引擎：自定义部署标记

```
$Flag = @(

    "Is_Mark_Sync" # 允许全盘搜索并同步部署标记
```

```
                              # 先决部署

                         #   "Auto_Update" # 允许自动更新

                         #   "Use_UTF8" # Beta 版：使用 Unicode UTF-8 提供全球语言支持

                           "Disable_Network_Location_Wizard" # 网络位置向导

                           "Disable_Cleanup_Appx_Tasks" # Appx 清理维护任务

                           "Disable_Cleanup_On_Demand_Language" # 阻止清理未使用的按需功能语言包

                           "Disable_Cleanup_Unsed_Language" # 阻止清理未使用的语言包

                           "Prerequisites_Reboot" # 重新启动计算机

                              # 完成首次部署

                         #   "Popup_Engine" # 允许首次弹出部署引擎主界面

                         #   "Allow_First_Pre_Experience" # 允许首次预体验，按计划

                           "Reset_Execution_Policy" # 恢复 PowerShell 执行策略：受限

                           "Clear_Solutions" # 删除整个解决方案

                           "Clear_Engine" # 删除部署引擎，保留其它

                         #   "First_Experience_Reboot" # 重新启动计算机

                         )

                         ForEach ($item in $Flag) {

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

                           New-Item -Path "D:\OS_2022_Custom\Install\Install\Mount\Yi\Engine\Deploy\Allow" -ItemType Directory -
                         ErrorAction SilentlyContinue | Out-Null

                           Out-File -FilePath "D:\OS_2022_Custom\Install\Install\Mount\Yi\Engine\Deploy\Allow\$($item)" -Encoding utf8 -
                         ErrorAction SilentlyContinue

                         }
```

2. 部署引擎：进阶

2.1. 部署引擎：添加过程中

在复制部署引擎后，可添加部署标记来干预安装过程。

2.2. 无人值守方案

自定义无人值守时，以下文件存在时请同步修改：

- 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. 多语言或单语

多语言时，单语时，可互相切换，替换时，请替换文件里所有相同的。

2.2.1.1. 多语言

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

<InputLocale>%OSDInputLocale%</InputLocale>

<SystemLocale>%OSDSystemLocale%</SystemLocale>

<UILanguage>%OSDUILanguage%</UILanguage>

<UILanguageFallback>%OSDUILanguageFallback%</UILanguageFallback>

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

2.2.1.2. 单语

单语需指定区域，例如指定区域：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. 用户方案

默认使用自建用户 Administrator 并自动登录，可通过修改以下配置切换：自建、自定义用户。

2.2.2.1. 自建用户 Administrator

默认使用自建用户：Administrator 并自动登录，插入到 <OOBE> 和 </OOBE> 之间。

```
<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.   自定义用户

设置自定义用户后，安装系统完成后，在 OOBE 里，可选择本地、在线用户等设置。

2.2.2.2.1.   删除

用户名：从开始处删除 `<UserAccounts>` 到 `</UserAccounts>`

自动登录：从开始处删除 `<AutoLogon>` 到 `</AutoLogon>`

2.2.2.2.2.   替换

从开始处 `<OOBE>` 到 `</OOBE>`

```
            <OOBE>

              <ProtectYourPC>3</ProtectYourPC>

              <HideEULAPage>true</HideEULAPage>

              <HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>

            </OOBE>
```

D.　　　生成 ISO

　　II　　　下载 OScdimg

　　　　根据架构选择 Oscdimg 版本，下载后保存到：D:\ ，保存在其它路径请输入 OScdimg.exe 绝对路径；

　　　　　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

　　III　　　使用 oscdimg 命令行生成一个 ISO 文件，保存到：D:\WS2022.iso

- ISO.ps1

  - \Expand\ISO.ps1

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

- 复制代码

```
$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, $FileName)

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

章节 2　　　常见问题

　　II　　　清理所有挂载到

　　　　关闭所有可能正在访问映像中文件的应用程序，包括文件资源管理器。

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

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

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

III   修复挂载出现异常的问题

1.   查看已挂载

Get-WindowsImage -Mounted

2.   删除保存在注册表里的 DISM 挂载记录

Remove-Item -Path "HKLM:\SOFTWARE\Microsoft\WIMMount\Mounted Images*" -Force -Recurse -ErrorAction SilentlyContinue | Out-Null

3.   删除与已损坏的已装载映像关联的所有资源。

Clear-WindowsCorruptMountPoint

Dism /cleanup-wim

IV   清理

封装过程中会产生大量的临时文件，安装 InBox Apps 应用、安装累积更新、安装语言包时会临时释放安装文件，所以不定期清理过时的会长期占用大量的磁盘空间，建议您尝试以下方法来实现清理计划，以达到释放更多的空间：

1.   常见日志

1.1.   使用命令行清理

```
$TempPaths = @( $env:Temp; "$($env:SystemRoot)\Logs\DISM"; )

foreach ($TempPath in $TempPaths) {

  if (Test-Path -Path $TempPath) {

    write-host "  $($TempPath)" -ForegroundColor Green

    Get-ChildItem -Path $TempPath -Recurse -Force | ForEach-Object {

      try {

        Remove-Item $_.FullName -Force -Recurse -ErrorAction SilentlyContinue | Out-Null

      } catch {

        write-host $_ -ForegroundColor Red

      }

    }

  }

}
```

1.2.   手动删除

1.2.1.   DISM 日志

使用"磁盘清理"功能，无法清理 DISM 产生的日志，需手动删除，路径：{系统盘}:\Windows\Logs\DISM

1.2.2.　临时目录

使用"磁盘清理"功能，无法清理临时目录的文件，需要手动操作，运行：%Temp% 可快速定位并打开临时目录，路径：

{系统盘}:\Users\{用户名}\AppData\Local\Temp

1.2.3.　清理"终端"运行的命令行记录

Remove-Item -Path (Get-PSReadlineOption).HistorySavePath -ErrorAction SilentlyContinue

执行清理命令行记录后，需重新启动"终端"生效。

2.　磁盘清理

运行 cleanmgr，选择要清理的磁盘和类型。

章节 3　已知问题

1.　添加 Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~zh-CN~.cab 到 Windows Server 2022 Standard Core, Windows Server 2022 Datacenter Core 后会新增 Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.20348.1，执行删除会报错，暂时不建议操作。

Yi' Solutions

此副本封装教程隶属于 Yi's Solutions 内容，学习更多：

- Yi 的官方网站 | https://fengyi.tel/solutions

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

作者：Yi

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

文档版本：1.5

文档里包含的所有脚本，最后测试时间：10 / 2024

文档最近更新日期：10 / 2024

建议或反馈：https://github.com/ilikeyi/solutions/issues