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

YI

YI’S SOLUTIONS

目录

章节 1　部署映像

A.　先决条件

II　要求

1.　系统安装包

1.1.　准备：en-us_windows_server_2022_x64_dvd_620d7eac.iso

1.2.　解压到：D:\en-us_windows_server_2022_x64_dvd_620d7eac

1.3.　解压完成后，将目录 en-us_windows_11_business_editions_version_22h2_x64_dvd_17a08c 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)

2.2.　语言包：下载

20348.1.210507-1500.fe_release_amd64fre_SERVER_LOF_PACKAGES_OEM.iso

III　命令行

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

2.　以管理员身份打开“Terminal”或“PowerShell ISE”，建议设置 PowerShell 执行策略：绕过，PS 命令行：

Set-ExecutionPolicy -ExecutionPolicy Bypass -Force

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

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 里运行。

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
  - \Expand\Extract.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/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"; }

        @{ 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
    } 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. 自定义部署映像

II 自定义部署映像：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 -ea SilentlyContinue
```

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. 自定义部署映像：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 -ea SilentlyContinue
```

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，查看报告“语言安装包适用于 WinRE.wim”。
- 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
  - \Expand\Install\WinRE\WinRE.Instl.lang.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/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 $Initl_install_Language_Component) {
        if ($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. 组件：映像中已安装的所有包

3.1.4.2.1. 查看

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

3.1.4.2.2. 导出到 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. 保存映像

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

3.1.6. 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\WinRE\Mount" -Discard
```

3.1.7. 重建 WinRE.wim 后，可缩小文件大小

- WinRE.Rebuild.ps1
  - \Expand\Install\WinRE\WinRE.Rebuild.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/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   Rebuild".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. 备份 WinRE.wim

- WinRE.Backup.ps1
  - \Expand\Install\WinRE\WinRE.Backup.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.2022/Expand/Install/WinRE/WinRE.Backup.ps1

- 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 -ea SilentlyContinue

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

3.1.9. 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，查看报告“语言安装包适用于 Install.wim”。
- 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
  - \Expand\Install\Install.Instl.lang.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/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. 语言包：删除，可选

- 添加语言后，如果要部署到非英语区域，可通过删除英语语言组件来节省空间。 删除语言时，请按与添加语言组件相反的顺序卸载语言组件。
- 添加中文后，反向删除“英语 - 美国”，区域：en-US，需提前提取语言包
- Install.Del.Specified.lang.Tag.ps1
  - \Expand\Install\WinRE\WinRE.Del.Specified.lang.Tag.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Expand/Install/Install/Install.Del.Specified.lang.Tag.ps1

- 复制代码

```
$Lang = "en-US"
$Mount = "D:\OS_2022_Custom\Install\Install\Mount"
$Sources = "D:\OS_2022_Custom\Install\Install\Language\Del\en-US"
```

```
$Initl_install_Language_Component = @()

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

   $Initl_install_Language_Component += $_.PackageName

}

$Language = @(

   @{ Match = "*WordPad*wow64*"; File = "Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

   @{ Match = "*WordPad*amd64*"; File = "Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~$($Lang)~.cab"; }

   @{ Match = "*WMIC*FoD*Package*wow64*"; File = "Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

   @{ Match = "*WMIC*FoD*Package*amd64*"; File = "Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

   @{ Match = "*StepsRecorder*wow64*"; File = "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

   @{ Match = "*StepsRecorder*amd64*"; File = "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~$($Lang)~.cab"; }

   @{ Match = "*Printing*PMCPPC*amd64*"; File = "Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

   @{ Match = "*PowerShell*wow64*"; File = "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

   @{ Match = "*PowerShell*amd64*"; File = "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~$($Lang)~.cab"; }

   @{ Match = "*MediaPlayer*wow64*"; File = "Microsoft-Windows-MediaPlayer-Package-wow64-$($Lang).cab"; }

   @{ Match = "*MediaPlayer*amd64*"; File = "Microsoft-Windows-MediaPlayer-Package-AMD64-$($Lang).cab"; }

   @{ Match = "*Notepad*wow64*"; File = "Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

   @{ Match = "*Notepad*amd64*"; File = "Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

   @{ Match = "*InternetExplorer*"; File = "Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

   @{ Match = "*TextToSpeech*"; File = "Microsoft-Windows-LanguageFeatures-TextToSpeech-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

   @{ Match = "*LanguageFeatures-Speech*"; File = "Microsoft-Windows-LanguageFeatures-Speech-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

   @{ Match = "*OCR*"; File = "Microsoft-Windows-LanguageFeatures-OCR-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

   @{ Match = "*Handwriting*"; File = "Microsoft-Windows-LanguageFeatures-Handwriting-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

   @{ Match = "*LanguageFeatures-Basic*"; File = "Microsoft-Windows-LanguageFeatures-Basic-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }
```

```
    @{ Match = "*Client-LanguagePack-Package*"; File = "Microsoft-Windows-Client-Language-Pack_x64_$($Lang).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)*$($Lang)*") {
            Write-host "   Component name: " -NoNewline
            Write-host $Component -ForegroundColor Green
            Write-host "   语言包 file: " -NoNewline
            Write-host "$($Sources)\$($Rule.File)" -ForegroundColor Green
            Write-Host "   Deleting ".PadRight(22) -NoNewline
            try {
                Remove-WindowsPackage -Path $Mount -PackagePath "$($Sources)\$($Rule.File)" -ErrorAction SilentlyContinue |
Out-Null
                Write-host "Finish" -ForegroundColor Green
            } catch {
                Write-host "Failed" -ForegroundColor Red
                Write-host "   $($_)" -ForegroundColor Red
            }
            break
        }
    }
}
$InitlClearLanguagePackage = @()
Get-WindowsPackage -Path $Mount | ForEach-Object {
    if ($_.PackageName -like "*$($Lang)*") {
        $InitlClearLanguagePackage += $_.PackageName
    }
}
if ($InitlClearLanguagePackage.count -gt 0) {
    ForEach ($item in $InitlClearLanguagePackage) {
        Write-Host "`n   $($item)" -ForegroundColor Green
        Write-Host "   Deleting ".PadRight(22) -NoNewline
        try {
            Remove-AppxProvisionedPackage -Path $Mount -PackageName $item -ErrorAction SilentlyContinue | Out-Null
```

```
        Write-host "Finish" -ForegroundColor Green
    } catch {
        Write-host "Failed" -ForegroundColor Red
        Write-host "   $($_)" -ForegroundColor Red
    }
  }
}
```

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

```
$SaveTo = "D:\OS_2022_Custom\Install\Install\Report.$(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，可选项

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，可选项

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，可选项

```
$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. 添加部署引擎，可选

- 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。

7. 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
```

8. 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
```

9. 保存映像

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

10. 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。

11. 重建 Install.wim 后可缩小文件大小

11.1. Install.Rebuild.wim.ps1

- \Expand\Install\Install.Rebuild.wim.ps1
- https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/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
}
```

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
  - \Expand\Install\Install.WinRE.Extract.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/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 -ea SilentlyContinue
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
  - \Expand\Install\Install.WinRE.Replace.wim.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/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
}
```

III 自定义部署映像：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 -ea SilentlyContinue
```

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，查看报告“语言安装包适用于 Boot.wim”。
- 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
  - \Expand\Boot\Boot.Instl.lang.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.2022/Expand/Boot/Boot.Instl.lang.ps1

- 复制代码

```
$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 $Component -ForegroundColor Green
      Write-host "   语言包 file: " -NoNewline
      Write-host "$($Sources)\$($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. 组件：映像中已安装的所有包

4.2.1. 查看

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

4.2.2. 导出到 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. 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
```

4.4. 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。

4.4.1. 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"
```

4.4.2. 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. 保存映像

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

6. 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\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：1.1.0.4，选择下载源代码格式：zip，下载完成后得到 Multilingual-1.1.0.4.zip 压缩包文件；

- 将已下载的 main.zip 或 Multilingual-1.1.0.4.zip，解压到：D:\Multilingual-1.1.0.4，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\Unattend\Mul.Unattend.xml 到 D:\OS_2022\Autounattend.xml

```
Copy-Item "D:\Multilingual\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\Unattend\Mul.Unattend.xml 到 D:\OS_2022\Sources\Unattend.xml

```
Copy-Item "D:\Multilingual\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\Unattend\Mul.Unattend.xml 到 D:\OS_2022\Sources\OEM\Panther\Unattend.xml

```
Copy-Item "D:\Multilingual\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“自定义部署映像：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\Unattend\Mul.Unattend.xml 到

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

```
Copy-Item "D:\Multilingual\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/_Documents/Attachment/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
```

作者：Yi

网站：https://fengyi.tel

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

文档版本：1.0

文档模型：完整版

更新日期：2024 - 1

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

Yi

Yi's Solutions