YI'S SOLUTIONS

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？

学习更多历史版本

https://github.com/ilikeyi/Solutions/tree/main/_Learn/Packaging.tutorial

摘要

目录

目录

目录

目录

目录

目录

章节 1　封装

A.　先决条件

I.　正在运行的系统

1.　检查正在运行的系统是否健康

检查正在运行的系统是否健康尤为重要，如果扫描后提示需要修复，向脱机映像添加累积更新和执行其它操作时会报错。如何检查，运行：sfc /scannow

验证过程完成后，如果问题已发现并纠正，您将在命令提示符窗口中看到类似以下内容：

1.1.　Windows 资源保护未发现任何完整性违规情况。

待处理：无需修复。

1.2.　Windows 资源保护发现了损坏的文件，但无法修复其中一些文件。

待处理：需要修复，可使用以下命令尝试修复：

Dism /Online /Cleanup-Image /CheckHealth

Dism /Online /Cleanup-Image /ScanHealth

dism /Online /Cleanup-Image /RestoreHealth

注意：若遇到无法修复，请重新安装系统。

2.　使用 DISM 命令时

2.1.　学习 DISM 概述

https://learn.microsoft.com/zh-cn/windows-hardware/manufacture/desktop/what-is-dism

DISM 限制，使用 DISM 维护 Windows 映像时，管理员必须注意其固有的限制。其中之一是它不支持弹性文件系统 (ReFS)。ReFS 是适用于 Windows Server 2022、Windows Server 2019、Windows Server 2016、Windows Server 2012 R2 和 Windows Server 2012 操作系统的全新文件系统。它可以高效地扩展到大型数据集，并提供比旧文件系统更高的数据可用性和性能。

警告您，磁盘分区，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。

正在运行的操作系统是 Windows 10 或者低于 Windows 11 25H2 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) 升级正在运行的操作系统或重新安装到更高的版本（建议）；

2) 升级或安装新版 ADK 或 PowerShell 版本（不建议）

- 可尝试升级到最新 PowerShell 7 以上的版本；
- 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。

3. 加速 Windows 操作系统的 N 种方法

在处理封装任务时，安装累积更新、安装驱动、安装 InBox Apps 里的应用时，会产生大量的临时文件，可用以下几种方法加速系统：

3.1. 关闭 Windows 防病毒

- 开启 Windows 防病毒会扫描文件、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”。

3.1.1. 打开“Windows 安全”或运行：windowsdefender:

3.1.2. 选择“病毒和威胁防护”或运行：windowsdefender://threat

3.1.3. 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：

3.1.3.1. 实时保护

3.1.3.2. 云提供的保护

3.1.3.3. 自动提交样本

3.1.3.4. 篡改防护

3.1.4. 未处于封装时，建议您开启 Windows 防病毒。

3.2. 为你的电脑关闭 Device/Credential Guard

即便是关闭 Windows 安全中心后，基于虚拟化的安全性处于运行中，系统运行速度也会大打折扣，自微软更新 Windows 11

24H2 之后，每次全新安装或者更新都会启用 Device/Credential Guard（或者应该说是 VBS），这就会导致开了嵌套虚拟化的 Vmware Workstation 和其它虚拟机软件无法正常使用，以及降速等。

但是，微软提供了一个名为 Device Guard and Credential Guard hardware readiness tool 工具来关闭 Device/Credential Guard，如何关闭：

3.2.1. 下载 Device Guard and Credential Guard hardware readiness tool

打开连接：https://www.microsoft.com/en-us/download/details.aspx?id=53337 后点击下载按钮，下载后获得 dgreadiness_v3.6.zip 文件，解压到 C 盘（解压到其它位置请执行时替换路径）。

3.2.2. 执行关闭命令

打开 Terminal（终端），依次运行：

```
Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope LocalMachine -Force
```

```
C:\dgreadiness_v3.6\DG_Readiness_Tool_v3.6.ps1 -Disable
```

3.2.3. 运行后，重新启动计算机

重启后 Windows 会进入 Credential Guard Opt-out Tool，此时按下 Windows 徽标键或 F3 后再按下回车即可继续。

3.2.4. 查看状态

重新启动计算机后，运行 Msinfo32，在系统摘要查看“基于虚拟化的安全”状态，或运行后查看状态：

```
C:\dgreadiness_v3.6\DG_Readiness_Tool_v3.6.ps1 -Ready
```

3.2.5. 如何恢复

运行命令行后重新启动计算机生效。

```
C:\dgreadiness_v3.6\DG_Readiness_Tool_v3.6.ps1 -Enable
```

使用 Device Guard and Credential Guard hardware readiness tool 工具无法关闭，或者尝试使用雷电模拟器的工具：https://res.ldmnq.com/test/HyperV-off.exe，下载后以管理员身份运行，完成后重新启动计算机。

3.3. 开启 Windows 原生 NVMe 存储堆栈

阅读微软官方新闻：Windows Server 2025 正式推出原生 NVMe：开启存储性能新时代

开启 Windows 原生 NVMe 存储堆栈，获取至高约 80% 的 IOPS 性能暴涨，如果使用的 NVMe 硬盘，建议您开启此项，开启前需

要满足：Windows server 2025 和 Windows 11 26100.6899 / 26200.6899 以上，该特征仅对 NVMe SSD 生效，请查看当前硬盘是否满足。

启用步骤：

3.3.1. 确保当前使用的存储控制器为 Windows 自带的标准 NVM Express 控制器，如当前正在使用其他非 Windows 自带的驱动，建议卸载。

3.3.2. 确保当前使用的操作系统版本号为 26100.6899 / 26200.6899 或 更高的版本

3.3.3. 以下两种方式（来自微软官方，本质上操作相同，二选一 即可）

3.3.3.1. 运行 PowerShell（管理员），执行以下命令，重启电脑

```
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Policies\Microsoft\FeatureManagement\Overrides /v 1176759950 /t REG_DWORD /d 1 /f
```

3.3.3.2. 或下载解压并安装以下组策略 MSI 文件，将其设为已启用，重启电脑

https://download.microsoft.com/download/123547b0-bff7-419d-96ba-d1cfee92f442/Windows%2011%2024H2,%20Windows%2011%2025H2%20and%20Windows%20Server%202025%20KB5066835%20251014_21251%20Feature%20Preview.msi

安装后位于：组策略 > 计算机配置 > 管理模板 > KB5066835 251014_21251 Feature Preview > Windows 11, version 24H2, 25H2

3.3.4. Windows 11 开启方式相对 Windows Server 有所不同，还需在 PowerShell（管理员）添加并执行以下命令：

```
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Policies\Microsoft\FeatureManagement\Overrides /v 1853569164 /t REG_DWORD /d 1 /f
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Policies\Microsoft\FeatureManagement\Overrides /v 156965516 /t REG_DWORD /d 1 /f
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Policies\Microsoft\FeatureManagement\Overrides /v 735209102 /t REG_DWORD /d 1 /f
```

3.3.5. 若已使用上述命令并重启电脑后依旧无法开启，也可用 ViveTool 命令强制开启

需要 ViveTool，前往：https://github.com/thebookisclosed/ViVe/releases 下载，下载完成后运行：

```
vivetool.exe /enable /id:48433719,49453572,5536923
```

28020 以上还需要运行：vivetool.exe /enable /id:48613417

3.3.6. 解决无法进入安全模式

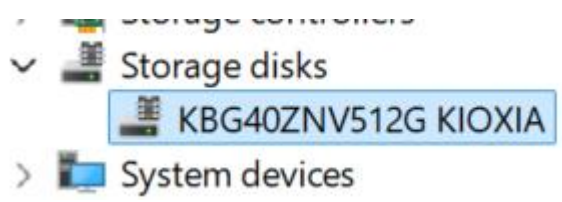
Windows 安全模式默认不加载此 NVMe 驱动，故还需执行以下命令使其加载，生效后安全模式即可正常使用

reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SafeBoot\Minimal\{75416E63-5912-4DFA-AE8F-3EFACCAFFB14}

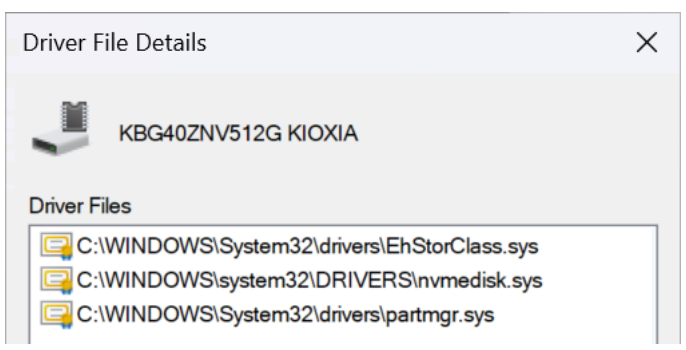
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SafeBoot\Network\{75416E63-5912-4DFA-AE8F-3EFACCAFFB14}

3.3.7. 检查该特性是否成功开启的方式与关闭方法

3.3.7.1. 成功开启 Windows 原生 NVMe 模式后，你的 NVMe SSD 在设备管理器中将被归类到全新的 存储磁盘 路径下，如图：



3.3.7.2. 成功开启 Windows 原生 NVMe 模式后，在设备管理器中，右键你的 NVMe SSD 选择属性，驱动程序 >驱动程序详细信息，会看到驱动程序文件会包含全新的 nvmedisk.sys ，如图：



3.3.7.3. 关闭方法

运行：regedit（注册表编辑器），进入

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Policies\Microsoft\FeatureManagement\Overrides 路径，将 1176759950、156965516、1853569164、735209102 四个键值改为 0 或 直接删除，重启电脑即可。

用 Vivetool 开启的 vivetool.exe /disable /id:48433719,49453572,5536923

28020 以上还需要运行：vivetool.exe /disable /id:48613417

4. 禁用常见文件类型安全警告

- 为什么关闭？选择任意一个 ISO 文件操作时，会弹出“安全警告”，很烦。
- “添加”和“恢复默认策略”后记得“重新启动 Explorer.exe 进程”并立即生效。

4.1. 如何添加

4.1.1. 第一步，选择规则

提供了常见的文件类型规则和仅 *.ISO 规则：

4.1.1.1. 常见文件类型规则

```
$SafetyWarningsExclude = ".exe;.reg;.msi;.bat;.cmd;.com;.vbs;.hta;.scr;.pif;.js;.iso;"
```

4.1.1.2. 仅 .iso 规则

```
$SafetyWarningsExclude = ".iso;"
```

4.1.2. 第二步，添加到系统策略

```
if ((Test-Path -LiteralPath "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Associations") -ne $true) { New-Item "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Associations" -force -ErrorAction SilentlyContinue | Out-Null }; New-ItemProperty -LiteralPath 'HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Associations' -Name 'LowRiskFileTypes' -Value $SafetyWarningsExclude -PropertyType String -force -ErrorAction SilentlyContinue | Out-Null
```

4.2. 如何恢复默认策略

```
Remove-Item -Path "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Associations" -Force -Recurse -ErrorAction SilentlyContinue | Out-Null
```

4.3. 重新启动 Explorer.exe 进程并立即生效

```
Stop-Process -ProcessName explorer -force -ErrorAction SilentlyContinue; Start-Sleep 5; $Running = Get-Process explorer -ErrorAction SilentlyContinue; if (-not ($Running)) { Start-Process "explorer.exe" }
```

5. 虚拟闪存盘

内存盘是什么？内存盘也被称为虚拟内存盘，它是一种可以提高电脑内存和文件快速访问的技术。但是内存盘会导致电脑在关闭之后会出现数据丢失，内存盘是比较不安全是一种设置。

虽然如此，但是我不这样认为，在封装过程中会频繁的释放安装包文件、生成日志等，挂载到虚拟盘时，这有很多好处，快速格式化。

添加语言包、添加累积更新、添加 InBox Apps 时，安装包存放于内存虚拟盘里，这样会占用大量的内存，建议您存放于非虚拟内存盘。

5.1. 软件推荐

5.1.1. Ultra RAMDisk | http://ultraramdisk.com

5.1.2. ImDisk | https://sourceforge.net/projects/imdisk-toolkit

5.1.3. AMD Radeon RAMDisk | http://www.radeonramdisk.com

5.1.4. Primo Ramdisk | https://www.romexsoftware.com/en-us/primo-ramdisk/index.html

5.1.5. SoftPerfect RAM Disk | https://www.softperfect.com/products/ramdisk

5.1.6. StarWind RAM Disk | https://www.starwindsoftware.com/high-performance-ram-disk-emulator

5.2. 如何创建

创建内存盘时应计算物理内存未使用率，打开“任务管理器”，“性能”，查看内存剩余率，建议：

5.3. 物理内存 16G，系统剩余 10G 时，建议划分：6G 内存 + 40G 交换文件，保留剩余内存 4G 以上；

5.4. 物理内存 32G：系统剩余 26G 时，建议划分：20 内存 + 40G 交换文件，保留剩余内存 6G 以上；

5.5. 物理内存 64G：系统剩余 54G 时，仅划分 40G 内存，无需创建交换文件，保留剩余内存 8G 以上；

5.6. 物理内存 128G：系统剩余 115G 时，划分 40 内存之间，无需创建交换文件。

注意：内存不足时会导致在封装过程中出现问题。

6. 命令行

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

6.2. 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 -Scope LocalMachine -Force
```

6.3. 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；

6.4. 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:\ISOTEMP

1.4. 所有脚本、所有路径，已默认设置为 D:\ISOTEMP 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

I. 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；

II. 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

III. 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/_Learn/Packaging.tutorial/OS.2022/Expand/Extract.ps1
- 复制代码

```
$ISO = "Auto"; $SaveTo = "D:\ISOTEMP_Custom";
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
      }
    }
  }
}
$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" ) }
)

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

C. 自定义封装

I. 自定义封装：Install.wim

1. 查看 Install.wim 详细信息

在开始挂载前，应先分析 Windows 映像的内容结构，通常包含多个不同的版本（如家庭版、企业版、教育版等），使用查看命令可显示：映像名称、映像描述、映像大小、架构、版本、索引号等，然后自定义所需的“索引号”进行挂载。

```
$ViewFile = "D:\ISOTEMP\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:\ISOTEMP_Custom\Install\Install\Mount" -ItemType directory
```

3. 开始挂载 Install.wim

默认索引号：1

```
Mount-WindowsImage -ImagePath "D:\ISOTEMP\sources\install.wim" -Index "1" -Path "D:\ISOTEMP_Custom\Install\Install\Mount"
```

- 验证挂载状态

挂载完成后，查看“已挂载的 Windows 映像信息，包括挂载目录、映像名称、状态”等，运行命令：

```
Get-WindowsImage -Mounted
```

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:\ISOTEMP_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:\ISOTEMP_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:\ISOTEMP_Custom\Install\WinRE\Mount"
```

- 验证挂载状态

  挂载完成后，查看“已挂载的 Windows 映像信息，包括挂载目录、映像名称、状态”等，运行命令：

  Get-WindowsImage -Mounted

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
  - \Expand\Install\WinRE\WinRE.Instl.lang.ps1
  - 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:\ISOTEMP_Custom\Install\WinRE\Mount"; $Sources = "D:\ISOTEMP_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. 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:\ISOTEMP_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:\ISOTEMP_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:\ISOTEMP_Custom\Install\WinRE\Mount" | Out-GridView
```

3.1.4.3.2. 导出到 Csv

```
$SaveTo = "D:\ISOTEMP_Custom\Install\WinRE\Report.Components.$(Get-Date -Format
"yyyyMMddHHmmss").csv"; Get-WindowsPackage -Path
"D:\ISOTEMP_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

```
Add-WindowsPackage -Path "D:\ISOTEMP_Custom\Install\WinRE\Mount" -PackagePath
"D:\ISOTEMP_Custom\Install\WinRE\Update\KB_WinRE.cab"
```

3.1.5.2. 删除

```
Remove-WindowsPackage -Path "D:\ISOTEMP_Custom\Install\WinRE\Mount" -PackagePath
"D:\ISOTEMP_Custom\Install\WinRE\Update\KB_WinRE.cab"
```

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:\ISOTEMP_Custom\Install\WinRE\Mount" /cleanup-image /StartComponentCleanup
/ResetBase
```

3.1.5.3.1. 固化更新后清理组件

```
$Mount = "D:\ISOTEMP_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; }}
```

3.1.6. 驱动

3.1.7. 保存映像：WinRE.wim

```
Save-WindowsImage -Path "D:\ISOTEMP_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:\ISOTEMP_Custom\Install\WinRE\Mount" -Discard
```

3.1.9. 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/_Learn/Packaging.tutorial/OS.2022/Expand/Install/WinRE/WinRE.Rebuild.ps1

- 复制代码

```
$FileName = "D:\ISOTEMP_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; }
```

3.1.10. 备份 WinRE.wim

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

- 复制代码

```
$WimLibPath = "D:\ISOTEMP_Custom\Install\Install\Update\Winlib"; $FileName = "D:\ISOTEMP_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
  - \Expand\Install\Install.Instl.lang.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Install/Install.Instl.lang.ps1

- 复制代码

```
Function Language_Install {
    param($Mount, $Sources, $Lang)
    $Initl_install_Language_Component = @(); $Script:Init_Folder_All_File_Match_Done = @(); $Script:Init_Folder_All_File_Exclude = @();
    if (Test-Path $Mount -PathType Container) { Get-WindowsPackage -Path $Mount | ForEach-Object { $Initl_install_Language_Component += $_.PackageName }; } else { Write-Host "Not mounted: $($Mount)"; return; }
```

```
$Script:Init_Folder_All_File = @()

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

    Get-ChildItem -Path $Sources -Recurse -Include "*.cab" -ErrorAction SilentlyContinue | ForEach-Object { $Script:Init_Folder_All_File += $_.FullName; }

    Write-Host "`n  Available language pack installation files";

    if ($Script:Init_Folder_All_File.Count -gt 0) { ForEach ($item in $Script:Init_Folder_All_File) { Write-Host "  $($item)"; } } else { Write-Host "There are no language pack files locally"; return; }

} else { Write-Host "Path does not exist: $($Sources)\$($Lang)"; return; }

$Script:Search_File_Order = @(

    @{ Name = "Fonts"; Description = "Fonts"; Rule = @( "*Fonts*"; ) }

    @{ Name = "Basic"; Description = "Basic"; Rule = @( "*LanguageFeatures-Basic*"; "*Client*Language*Pack*"; ) }

    @{ Name = "OCR"; Description = "Optical character recognition"; Rule = @( "*LanguageFeatures-OCR*"; ) }

    @{ Name = "Handwriting"; Description = "Handwriting recognition"; Rule = @( "*LanguageFeatures-Handwriting*"; ) }

    @{ Name = "TextToSpeech"; Description = "Text-to-speech"; Rule = @( "*LanguageFeatures-TextToSpeech*"; ); }

    @{ Name = "Speech";; Description = "Speech recognition";; Rule = @( "*LanguageFeatures-Speech*"; ) }

    @{ Name = "RegionSpecific"; Description = "Other region-specific requirements"; Rule = @( "*InternationalFeatures*"; ) }

    @{ Name = "Retail"; Description = "Retail demo experience"; Rule = @( "*RetailDemo*"; ) }

    @{ Name = "Features_On_Demand"; Description = "Features on demand"; Rule = @( "*InternetExplorer*~x86~*"; "*InternetExplorer*~x64~*";
"*InternetExplorer*~amd64~*"; "*InternetExplorer*~wow64~*"; "*InternetExplorer*~arm64.x86~*"; "*InternetExplorer*~arm64~*"; "*MSPaint*~x86~*"; "*MSPaint*~x64~*";
"*MSPaint*~amd64~*"; "*MSPaint*~wow64~*"; "*MSPaint*~arm64.x86~*"; "*MSPaint*~arm64~*"; "*Notepad-FoD-Package*~x86~*"; "*Notepad-FoD-Package*~x64~*";
"*Notepad-FoD-Package*~amd64~*"; "*Notepad-FoD-Package*~wow64~*"; "*Notepad-FoD-Package*~arm64.x86~*"; "*Notepad-FoD-Package*~arm64~*"; "*Notepad-
System-FoD-Package*~x86~*"; "*Notepad-System-FoD-Package*~x64~*"; "*Notepad-System-FoD-Package*~amd64~*"; "*Notepad-System-FoD-Package*~wow64~*";
"*Notepad-System-FoD-Package*~arm64.x86~*"; "*Notepad-System-FoD-Package*~arm64~*"; "*MediaPlayer*~x86~*"; "*MediaPlayer*~x64~*"; "*MediaPlayer*~amd64~*";
"*MediaPlayer*~wow64~*"; "*MediaPlayer*~arm64.x86~*"; "*MediaPlayer*~arm64~*"; "*PowerShell*ISE*~x86~*"; "*PowerShell*ISE*~x64~*";
"*PowerShell*ISE*~amd64~*"; "*PowerShell*ISE*~wow64~*"; "*PowerShell*ISE*~arm64.x86~*"; "*PowerShell*ISE*~arm64~*"; "*StepsRecorder*~x86~*";
"*StepsRecorder*~x64~*"; "*StepsRecorder*~amd64~*"; "*StepsRecorder*~wow64~*"; "*StepsRecorder*~arm64.x86~*"; "*StepsRecorder*~arm64~*";
"*SnippingTool*~x86~*"; "*SnippingTool*~x64~*"; "*SnippingTool*~amd64~*"; "*SnippingTool*~wow64~*"; "*SnippingTool*~arm64.x86~*"; "*SnippingTool*~arm64~*";
"*WMIC*~x86~*"; "*WMIC*~x64~*"; "*WMIC*~amd64~*"; "*WMIC*~wow64~*"; "*WMIC*~arm64.x86~*"; "*WMIC*~arm64~*"; "*WordPad*~x86~*"; "*WordPad*~x64~*";
"*WordPad*~amd64~*"; "*WordPad*~wow64~*"; "*WordPad*~arm64.x86~*"; "*WordPad*~arm64~*"; "*Printing-WFS*~x86~*"; "*Printing-WFS*~x64~*"; "*Printing-
WFS*~amd64~*"; "*Printing-WFS*~wow64~*"; "*Printing-WFS*~arm64.x86~*"; "*Printing-WFS*~arm64~*"; "*Printing-PMCPPC*~x86~*"; "*Printing-PMCPPC*~x64~*";
"*Printing-PMCPPC*~amd64~*"; "*Printing-PMCPPC*~wow64~*"; "*Printing-PMCPPC*~arm64.x86~*"; "*Printing-PMCPPC*~arm64~*"; "*Telnet-Client*~x86~*"; "*Telnet-
Client*~x64~*"; "*Telnet-Client*~amd64~*"; "*Telnet-Client*~wow64~*"; "*Telnet-Client*~arm64.x86~*"; "*Telnet-Client*~arm64~*"; "*TFTP-Client*~x86~*"; "*TFTP-
Client*~x64~*"; "*TFTP-Client*~amd64~*"; "*TFTP-Client*~wow64~*"; "*TFTP-Client*~arm64.x86~*"; "*TFTP-Client*~arm64~*"; "*VBSCRIPT*~x86~*";
"*VBSCRIPT*~x64~*"; "*VBSCRIPT*~amd64~*"; "*VBSCRIPT*~wow64~*"; "*VBSCRIPT*~arm64.x86~*"; "*VBSCRIPT*~arm64~*"; "*WinOcr-FOD-Package*~x86~*";
"*WinOcr-FOD-Package*~x64~*"; "*WinOcr-FOD-Package*~amd64~*"; "*WinOcr-FOD-Package*~wow64~*"; "*WinOcr-FOD-Package*~arm64.x86~*"; "*WinOcr-FOD-
Package*~arm64~*"; "*ProjFS-OptionalFeature-FOD-Package*~x86~*"; "*ProjFS-OptionalFeature-FOD-Package*~x64~*"; "*ProjFS-OptionalFeature-FOD-
Package*~amd64~*"; "*ProjFS-OptionalFeature-FOD-Package*~wow64~*"; "*ProjFS-OptionalFeature-FOD-Package*~arm64.x86~*"; "*ProjFS-OptionalFeature-FOD-
Package*~arm64~*"; "*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package*~x86~*"; "*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package*~x64~*";
"*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package*~amd64~*"; "*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package*~wow64~*";
"*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package*~arm64.x86~*"; "*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package*~arm64~*";
"*ServerCoreFonts-NonCritical-Fonts-Support-FOD-Package*~x86~*"; "*ServerCoreFonts-NonCritical-Fonts-Support-FOD-Package*~x64~*"; "*ServerCoreFonts-NonCritical-
Fonts-Support-FOD-Package*~amd64~*"; "*ServerCoreFonts-NonCritical-Fonts-Support-FOD-Package*~wow64~*"; "*ServerCoreFonts-NonCritical-Fonts-Support-FOD-
Package*~arm64.x86~*"; "*ServerCoreFonts-NonCritical-Fonts-Support-FOD-Package*~arm64~*"; "*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package*~x86~*";
"*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package*~x64~*"; "*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package*~amd64~*"; "*ServerCoreFonts-
```

```
NonCritical-Fonts-TrueType-FOD-Package*~wow64~*"; "*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package*~arm64.x86~*"; "*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package*~arm64~*"; "*SimpleTCP-FOD-Package*~x86~*"; "*SimpleTCP-FOD-Package*~x64~*"; "*SimpleTCP-FOD-Package*~amd64~*"; "*SimpleTCP-FOD-Package*~wow64~*"; "*SimpleTCP-FOD-Package*~arm64.x86~*"; "*SimpleTCP-FOD-Package*~arm64~*"; "*VirtualMachinePlatform-Client-Disabled-FOD-Package*~x86~*"; "*VirtualMachinePlatform-Client-Disabled-FOD-Package*~x64~*"; "*VirtualMachinePlatform-Client-Disabled-FOD-Package*~amd64~*"; "*VirtualMachinePlatform-Client-Disabled-FOD-Package*~wow64~*"; "*VirtualMachinePlatform-Client-Disabled-FOD-Package*~arm64.x86~*"; "*VirtualMachinePlatform-Client-Disabled-FOD-Package*~arm64~*"; "*DirectoryServices-ADAM-Client-FOD-Package*~x86~*"; "*DirectoryServices-ADAM-Client-FOD-Package*~x64~*"; "*DirectoryServices-ADAM-Client-FOD-Package*~amd64~*"; "*DirectoryServices-ADAM-Client-FOD-Package*~wow64~*"; "*DirectoryServices-ADAM-Client-FOD-Package*~arm64.x86~*"; "*DirectoryServices-ADAM-Client-FOD-Package*~arm64~*"; "*EnterpriseClientSync-Host-FOD-Package*~x86~*"; "*EnterpriseClientSync-Host-FOD-Package*~x64~*"; "*EnterpriseClientSync-Host-FOD-Package*~amd64~*"; "*EnterpriseClientSync-Host-FOD-Package*~wow64~*"; "*EnterpriseClientSync-Host-FOD-Package*~arm64.x86~*"; "*EnterpriseClientSync-Host-FOD-Package*~arm64~*"; "*SenseClient-FoD-Package*~x86~*"; "*SenseClient-FoD-Package*~x64~*"; "*SenseClient-FoD-Package*~amd64~*"; "*SenseClient-FoD-Package*~wow64~*"; "*SenseClient-FoD-Package*~arm64.x86~*"; "*SenseClient-FoD-Package*~arm64~*"; "*SmbDirect-FOD-Package*~x86~*"; "*SmbDirect-FOD-Package*~x64~*"; "*SmbDirect-FOD-Package*~amd64~*"; "*SmbDirect-FOD-Package*~wow64~*"; "*SmbDirect-FOD-Package*~arm64.x86~*"; "*SmbDirect-FOD-Package*~arm64~*"; "*TerminalServices-AppServer-Client-FOD-Package*~x86~*"; "*TerminalServices-AppServer-Client-FOD-Package*~x64~*"; "*TerminalServices-AppServer-Client-FOD-Package*~amd64~*"; "*TerminalServices-AppServer-Client-FOD-Package*~wow64~*"; "*TerminalServices-AppServer-Client-FOD-Package*~arm64.x86~*"; "*TerminalServices-AppServer-Client-FOD-Package*~arm64~*"; )}

  )

  ForEach ($item in $Script:Search_File_Order) { New-Variable -Name "Init_File_Type_$($item.Name)" -Value @() -Force }

  $ExcludeNameItem = @( "Fonts"; "RegionSpecific"; )

  ForEach ($WildCard in $Script:Init_Folder_All_File) {

    ForEach ($item in $Script:Search_File_Order) {

      ForEach ($NewRule in $item.Rule) {

        if ($WildCard -like "*$($NewRule)*") {

          Write-Host "`n   Fuzzy matching: " -NoNewline; Write-Host $NewRule -ForegroundColor Green; Write-Host "   File name: " -NoNewline; Write-Host $WildCard -ForegroundColor Green;

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

          $TempSave = @{ Match_Name = $NewRule; FileName = $WildCard }

          $new = $OSDefaultUser + $TempSave

          if ($item.name -Contains $ExcludeNameItem) { Write-Host "   Do not match, install directly" -ForegroundColor Yellow; New-Variable -Name "Init_File_Type_$($item.Name)" -Value $new -Force; $Script:Init_Folder_All_File_Match_Done += $WildCard; } else {

            ForEach ($Component in $Initl_install_Language_Component) { if ($Component -like "*$($NewRule)*") { Write-Host "   Component name: " -NoNewline; Write-Host $Component -ForegroundColor Green; New-Variable -Name "Init_File_Type_$($item.Name)" -Value $new -Force; $Script:Init_Folder_All_File_Match_Done += $WildCard; break; }}

          }

        }

      }

    }

  }

  Write-Host "`n   Grouping is complete, pending installation" -ForegroundColor Yellow; Write-Host "   $('-' * 80)";

  ForEach ($WildCard in $Script:Search_File_Order) {
```

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

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

    if ($OSDefaultUser.Count -gt 0) {  ForEach ($item in $OSDefaultUser) { Write-Host "   $($item.FileName)" -ForegroundColor Green; } } else { Write-Host "   Not available" -ForegroundColor Red; }

  }

  Write-Host "`n  Not matched, no longer installed" -ForegroundColor Yellow; Write-Host "  $('-' * 80)";

  ForEach ($item in $Script:Init_Folder_All_File) { if ($Script:Init_Folder_All_File_Match_Done -notcontains $item) { $Script:Init_Folder_All_File_Exclude += $item; Write-Host "$($item)" -ForegroundColor Red; }}

  Write-Host "`n  Install" -ForegroundColor Yellow; Write-Host "  $('-' * 80)";

ForEach ($WildCard in $Script:Search_File_Order) {

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

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

  if ($OSDefaultUser.Count -gt 0) {

    ForEach ($item in $OSDefaultUser) {

      Write-Host "  File name: " -NoNewline; Write-Host $item.FileName -ForegroundColor Green; Write-Host "  Installing ".PadRight(22) -NoNewline;

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

        try {

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

          Write-Host "Finish`n" -ForegroundColor Green

        } catch { Write-Host "Failed" -ForegroundColor Red; Write-Host "  $($_)" -ForegroundColor Red; }

      } else { Write-Host "Does not exist`n"; }

    }

    } else { Write-Host "  Not available`n" -ForegroundColor Red; }

  }

}

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

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:\ISOTEMP_Custom\Install\Install\Mount" /Set-AllIntl:zh-CN
```

4.2.2. 查看可用的语言设置

```
Dism /Image:"D:\ISOTEMP_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:\ISOTEMP_Custom\Install\Install\Mount" | Out-GridView
```

4.3.2. 导出到 Csv

```
$SaveTo = "D:\ISOTEMP_Custom\Install\Install\Report.Components.$(Get-Date -Format "yyyyMMddHHmmss").csv"; Get-WindowsPackage -Path "D:\ISOTEMP_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:\ISOTEMP_Custom\Install\Install\Update\windows10.0-kb5030216-x64_cbe587155f9818548b75f65d5cd41d341ed2fc61.msu

5.2. 添加

```
Add-WindowsPackage -Path "D:\ISOTEMP_Custom\Install\Install\Mount" -PackagePath "D:\ISOTEMP_Custom\Install\Install\Update\windows10.0-kb5030216-x64_cbe587155f9818548b75f65d5cd41d341ed2fc61.msu"
```

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:\ISOTEMP_Custom\Install\Install\Mount" /cleanup-image /StartComponentCleanup /ResetBase
```

5.3.1. 固化更新后清理组件

```
$Mount = "D:\ISOTEMP_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:\ISOTEMP_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:\ISOTEMP_Custom\Install\Install\Update\Winlib\WinRE.wim"; $CopyTo =
   "D:\ISOTEMP_Custom\Install\Install\Mount\Windows\System32\Recovery"; Copy-Item -Path $WinRE -Destination $CopyTo -Force;
   ```

10. 保存映像

    ```
    Save-WindowsImage -Path "D:\ISOTEMP_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:\ISOTEMP_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
  - \Expand\Install\Install.WinRE.Extract.ps1
  - 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:\ISOTEMP\sources\install.wim", "1", "\Windows\System32\Recovery\Winre.wim", "--dest-dir=""D:\ISOTEMP_Custom\Install\Install\Update\Winlib""" ); New-Item -Path "D:\ISOTEMP_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
  - \Expand\Install\Install.WinRE.Replace.wim.ps1
  - 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:\ISOTEMP\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:\ISOTEMP\sources\install.wim", $_.ImageIndex, "--command=""add 'D:\ISOTEMP_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

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

- 复制代码

```
$InstallWim = "D:\ISOTEMP\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   Under reconstruction".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; }
```

14. 拆分、合并、压缩、互转

固实压缩为 ESD 文件格式，如果文件超过 4GB 时：无法使用拆分、无法复制到 FAT32 格式的磁盘，这是缺点。

使用 FAT32 格式存放 Windows 安装引导是最佳的解决方案，如果遇到 Install.wim 文件超过 4GB 时无法复制到 FAT32 格式的磁盘时，这时你需要将 Install.wim 拆分，小于 4GB 文件大小后才能复制到 FAT32 格式的磁盘里。

学会如何拆分和合并，固实压缩和互转，尤为重要。

14.1. 拆分和合并

14.1.1. 拆分

将 Install.wim 拆分为 4GB 文件大小后，获得新的文件名 Install.*.swm 后，删除旧的 Install.wim。

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

- 复制代码

```
Write-host "Split Install.wim into Install.*.swm"; Write-host "Splitting" -NoNewline; Split-WindowsImage -ImagePath "D:\ISOTEMP\sources\install.wim" -SplitImagePath "D:\ISOTEMP\sources\install.swm" -FileSize "4096" -CheckIntegrity -ErrorAction SilentlyContinue | Out-Null; Write-Host "Split Complete`n" -ForegroundColor Green; Write-host "`nVerify completion and delete old files"; if (Test-Path -Path "D:\ISOTEMP\sources\install.swm" -PathType leaf) { Remove-Item -Path "D:\ISOTEMP\sources\install.wim"; Write-Host "Done" -ForegroundColor Green; } else { Write-Host "Failed" -ForegroundColor Red; }
```

14.1.2. 合并

把所有 Install.*.swm 合并为 Install.wim 后，删除旧的 Install.*.swm。

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

- 复制代码

```
Write-host "Merge all Install.*.swm files into Install.wim"; Get-WindowsImage -ImagePath
"D:\ISOTEMP\Sources\install.swm" -ErrorAction SilentlyContinue | ForEach-Object { Write-Host "Image Name: " -
NoNewline; Write-Host $_.ImageName -ForegroundColor Yellow; Write-Host "Index Number: " -NoNewline;  Write-Host
$_.ImageIndex -ForegroundColor Yellow; Write-Host "Exporting".PadRight(28) -NoNewline; dism /export-image
/SourceImageFile:"D:\ISOTEMP\Sources\install.swm" /swmfile:"D:\ISOTEMP\sources\install*.swm"
/SourceIndex:"$($_.ImageIndex)" /DestinationImageFile:"D:\ISOTEMP\Sources\install.wim" /Compress:"Max"
/CheckIntegrity; Write-Host "Export Complete`n" -ForegroundColor Green; }; Write-host "`nVerify completion and delete
old files"; if (Test-Path -Path "D:\ISOTEMP\Sources\install.wim" -PathType leaf) { Get-ChildItem -Path
"D:\ISOTEMP\sources" -Recurse -include "*.swm" | ForEach-Object { Write-Host "Delete: $($_.Fullname)" -
ForegroundColor Green; Remove-Item -Path $_.Fullname; }; Write-Host "Done" -ForegroundColor Green; } else { Write-
Host "Falied" -ForegroundColor Green; }
```

14.2. 固实压缩 ESD 格式和互转 WIM 格式

14.2.1. 固实压缩

固实压缩后可以编辑版本信息和应用文件等；不可以挂载映像等，获得新的文件 install.esd 后，删除旧的 Install.wim。

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

- 复制代码

```
Write-host "Solid compressed Install.wim"; Get-WindowsImage -ImagePath "D:\ISOTEMP\Sources\install.wim" -
ErrorAction SilentlyContinue | ForEach-Object { Write-Host "Image Name: " -NoNewline; Write-Host $_.ImageName -
ForegroundColor Yellow; Write-Host "Index Number: " -NoNewline; Write-Host $_.ImageIndex -ForegroundColor Yellow;
Write-Host "Compressing".PadRight(28) -NoNewline; dism /export-image
/SourceImageFile:"D:\ISOTEMP\Sources\install.wim" /SourceIndex:"$($_.ImageIndex)"
/DestinationImageFile:"D:\ISOTEMP\Sources\install.esd" /Compress:recovery /CheckIntegrity; Write-Host
"Compression completed`n" -ForegroundColor Green; }; Write-host "`nVerify completion and delete old files"; if (Test-
Path -Path "D:\ISOTEMP\Sources\install.esd" -PathType leaf) { Remove-Item -Path "D:\ISOTEMP\Sources\install.wim";
Write-Host "Done" -ForegroundColor Green; } else { Write-Host "Falied" -ForegroundColor Green; }
```

14.2.2. 压缩文件转换为 WIM 文件格式

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

- 复制代码

```
Write-host "Convert ESD to WIM"; Get-WindowsImage -ImagePath "D:\ISOTEMP\Sources\install.esd" -ErrorAction
```

```
SilentlyContinue | ForEach-Object { Write-Host "Image Name: " -NoNewline; Write-Host $_.ImageName -
ForegroundColor Yellow; Write-Host "Index Number: " -NoNewline; Write-Host $_.ImageIndex -ForegroundColor Yellow;
Write-Host "Exporting".PadRight(28) -NoNewline; try { Export-WindowsImage -SourceImagePath
"D:\ISOTEMP\Sources\install.esd" -SourceIndex $_.ImageIndex -DestinationImagePath
"D:\ISOTEMP\Sources\install.wim" -CompressionType "Max" -CheckIntegrity -ErrorAction SilentlyContinue | Out-Null;
Write-Host "Done`n" -ForegroundColor Green; } catch { Write-Host $_ -ForegroundColor Yellow; Write-host "Falied`n" -
ForegroundColor Red; }}; Write-host "`nVerify completion and delete old files"; if (Test-Path -Path
"D:\ISOTEMP\Sources\install.wim" -PathType leaf) { Remove-Item -Path "D:\ISOTEMP\Sources\install.esd"; Write-Host
"Done" -ForegroundColor Green; } else { Write-Host "Falied" -ForegroundColor Green; }
```

II. 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:\ISOTEMP\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:\ISOTEMP_Custom\Boot\Boot\Mount" -ItemType directory
```

3. 开始挂载 Boot.wim

默认索引号：2

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

- 验证挂载状态

  挂载完成后，查看“已挂载的 Windows 映像信息，包括挂载目录、映像名称、状态”等，运行命令：

  ```
  Get-WindowsImage -Mounted
  ```

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
  - \Expand\Boot\Boot.Instl.lang.ps1

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

- 复制代码

```
$Mount = "D:\ISOTEMP_Custom\Boot\Boot\Mount"; $Sources = "D:\ISOTEMP_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:\ISOTEMP_Custom\Boot\Boot\Mount" /Set-AllIntl:zh-CN
```

4.2.2. 查看可用的语言设置

```
Dism /Image:"D:\ISOTEMP_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:\ISOTEMP_Custom\Boot\Boot\Mount" | Out-GridView
```

4.3.2. 导出到 Csv

```
$SaveTo = "D:\ISOTEMP_Custom\Boot\Boot\Report.Components.$(Get-Date -Format "yyyyMMddHHmmss").csv"; Get-WindowsPackage -Path "D:\ISOTEMP_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:\ISOTEMP_Custom\Boot\Boot\Mount\sources\zh-CN" -Destination "D:\ISOTEMP\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:\ISOTEMP_Custom\Boot\Boot\Mount\Sources\lang.ini

```
Dism /image:"D:\ISOTEMP_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\ISOTEMP_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:\ISOTEMP\Sources\lang.ini

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

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

```
Add-WindowsPackage -Path "D:\ISOTEMP_Custom\Boot\Boot\Mount" -PackagePath "D:\ISOTEMP_Custom\Boot\Boot\Update\KB_Boot.cab"
```

5.2. 删除

```
Remove-WindowsPackage -Path "D:\ISOTEMP_Custom\Boot\Boot\Mount" -PackagePath
```

"D:\ISOTEMP_Custom\Boot\Boot\Update\KB_Boot.cab"

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:\ISOTEMP_Custom\Boot\Boot\Mount" /cleanup-image /StartComponentCleanup /ResetBase

5.3.1. 固化更新后清理组件

$Mount = "D:\ISOTEMP_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:\ISOTEMP_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:\ISOTEMP_Custom\Boot\Boot\Mount" -Discard

III. 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:\ISOTEMP\Autounattend.xml

```
Copy-Item "D:\Multilingual\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\ISOTEMP\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:\ISOTEMP\Sources\Unattend.xml

```
Copy-Item "D:\Multilingual\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\ISOTEMP\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:\ISOTEMP\sources\`$OEM$\`$$\Panther" -ItemType Directory
```

1.1.1.3.2. 复制

复制 D:\Multilingual_Learn\Unattend\Mul.Unattend.xml 到 D:\ISOTEMP\Sources\OEM\Panther\Unattend.xml

```
Copy-Item "D:\Multilingual\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\ISOTEMP\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:\ISOTEMP\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:\ISOTEMP\Sources\$OEM$\$1\Yi\Engine

```
Copy-Item "D:\Multilingual\Engine" -Destination "D:\ISOTEMP\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:\ISOTEMP\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -ErrorAction SilentlyContinue | Out-Null
  Out-File -FilePath "D:\ISOTEMP\sources\`$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:\ISOTEMP_Custom\Install\Install\Mount

1.2.1. 无人值守

复制 D:\Multilingual_Learn\Unattend\Mul.Unattend.xml 到 D:\ISOTEMP_Custom\Install\Install\Mount\Panther\Unattend.xml

```
Copy-Item "D:\Multilingual\_Learn\Unattend\Mul.Unattend.xml" -Destination
"D:\ISOTEMP_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:\ISOTEMP_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:\ISOTEMP_Custom\Install\Install\Mount\Yi\Engine

```
Copy-Item "D:\Multilingual\Engine" -Destination "D:\ISOTEMP_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:\ISOTEMP_Custom\Install\Install\Mount\Yi\Engine\Deploy\Allow" -ItemType Directory -ErrorAction SilentlyContinue | Out-Null
  Out-File -FilePath "D:\ISOTEMP_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:\ISOTEMP\Autounattend.xml
- D:\ISOTEMP\Sources\Unattend.xml
- D:\ISOTEMP\sources\$OEM$\$$\Panther\unattend.xml
- D:\ISOTEMP_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

1. 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/refs/heads/main/_Encapsulation/Modules/AIO/Oscdimg/amd64/oscdimg.exe

1.2. x86

https://github.com/ilikeyi/Solutions/raw/refs/heads/main/_Encapsulation/Modules/AIO/Oscdimg/x86/oscdimg.exe

1.3. arm64

https://github.com/ilikeyi/Solutions/raw/refs/heads/main/_Encapsulation/Modules/AIO/Oscdimg/arm64/oscdimg.exe

2. 使用 oscdimg 命令行生成一个 ISO 文件，保存到：D:\ISOTEMP.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:\ISOTemp"; $Volume = "ISOTemp"; $SaveTo = "D:\ISOTemp.iso"; $Arguments = @("-m", "-o", "-u2", "-udfver102", "-l""$($Volume)""", "-bootdata:2#p0,e,b""$($ISO)\boot\etfsboot.com""#pEF,e,b""$($ISO)\efi\microsoft\boot\efisys.bin""", """$($ISO)""", """$($SaveTo)"""); Start-Process -FilePath $Oscdimg -ArgumentList $Arguments -wait -nonewwindow;
```

章节 2 部署

A. 部署前应注意

1. 在选择安装 Windows 11 的位置时

先选择“磁盘分区”，再选择“格式化分区”，再“下一步”，如果未格式化磁盘，覆盖安装会导致已知问题：

- 使用 Administator 用户：
  - 应用程序图标会出现 UAC 安全提示；
  - 点右键运行 PS1 以管理员身份运行无效；

2. 运行安装程序时

- 你不得在正在使用的系统里运行 ISO 下的 Setup.exe 进入安装程序，否则会导致安装过程中复制 OEM 目录异常，此问题属于安装程序的 BUG。
- 安装系统到物理设备时，您只能使用 U 盘、CD-ROM、PE、通过网络安装（PXE 启动）等方式进入安装程序，必须“全新安装”。

B. 部署操作系统到物理设备

物理设备进入系统安装引导前，你必须选择“准备引导安装程序前先决条件”中的“创建可引导的安装物理存储介质、CD-ROM、通过网络安装（PXE 启动）”其中一项并完成。

1. 准备引导安装程序前先决条件

部署系统安装文件到物理存储设备时，应准备可移动驱动器、CD-ROM 来存放 Windows 操作系统的安装文件。

1.1. 创建可引导的安装物理存储介质

准备存储大于 16G 的可移动硬盘、USB 驱动器，购买可移动驱动器时，应选择双接口（Type-C、USB3.1），选择带有 Type-C、USB3.1 双接口的好处有：

- 在安装系统过程中若缺少驱动时：可通过“手机”下载后，插上带有 Type-C 的可移动驱动器进行文件管理，将已下载的“驱动”复制到“可移动驱动器”里；
- 在日常过程中，可使用 Type-C 连接至手机用于存储临时文件和备份数据等操作。

1.1.1. 磁盘分区

- 在 MacOS（除 M 系列）、PC 里安装 Windows 系统，使用 FAT32 格式存放 Windows 系统安装文件是最佳的解决方案。
- 存储设备小于 16G 时，建议您直接划分 1 个分区；
- 存储设备大于 32G 以上时，建议您划分 3 个分区，划分方案：

  分区 1，分配第 2 分区 6 GB 后，剩余的所有磁盘空间用于：存放 Windows 系统安装文件（将 Yi.Deploy.Rule.iso 解压后可自定义部署）、日常可用于存储临时文件和备份数据等操作。

  分区 2：划分磁盘空间 6G，用于存放 PE 系统，推荐：

  - Sergei Strelec | https://sergeistrelec.name/winpe-10-8-sergei-strelec-english
  - Hirens BootCD | https://www.hirensbootcd.org

1.1.2. 复制系统安装文件到磁盘分区里

格式化分区请选择 Fat32 后，再复制 ISO 下的所有文件到 U 盘根目录即可完成创建。

1.2. CD-ROM

1.2.1. 准备一台“可刻录光盘机”

1.2.2. 准备“空白光盘”

1.2.3. 选择“ISO 文件”后，选择右键“刻录”功能后点击开始刻录，等待完成。

1.3. 通过网络安装（PXE 启动）

每软件使用方法不同，使用前请学习，可选择：

1.3.1. Serva | https://www.vercot.com/~serva

1.3.2. TinyPXE Server | http://labalec.fr/erwan/?page_id=958

1.3.3. iventory | https://www.iventoy.com

2. 物理设备进入系统安装引导

- 开机时，根据不同的主板按不同的键进入“启动菜单”“BIOS 菜单里选择磁盘引导”，常见的 BIOS 启动热键：F2、F8、F9、F11、F12、ESC
- 根据启动介质选择不同的菜单：CD-ROM、PXE、选择 U 盘已识别出来的分区。

C. 部署到正在使用的系统里，将原生启动 VHD 添加到现有到启动菜单里

1. 创建 VHD/VHDX 文件

1.1. 磁盘管理交互式

打开“磁盘管理（diskmgmt.msc）”，选择“动作”，选择“创建 VHD”，弹出“创建或附加虚拟磁盘”对话框后：

- 设置“位置：D:\OS.vhdx”
- 设置“虚拟磁盘大小：120，选择：GB”
- 选择“虚拟磁盘格式：VHDX”
- 选择“动态扩展”

点击“确定”后，显示磁盘区域会新增一个磁盘，依次：

- 选择“磁盘 2（选择时请查看是否选择正确的磁盘）”后点击“右键”，选择“初始化磁盘”
- 选择“磁盘 2 分区”后点击“右键”，选择“新建简单卷向导”并完成。

1.2. 命令行创建

快速创建（保存到：D:\OS.vhdx，虚拟磁盘大小：120GB，动态扩展，分配盘符：Q），命令行：

- 输入 Diskpart 后“回车”，在此对话框里依次运行：

```
Create Vdisk File="D:\OS.vhdx" Maximum=122880 Type=expandable
Select Vdisk file="D:\OS.vhdx"
Attach Vdisk
Create Partition Primary
```

```
Format Fs=NTFS Label="VOS" Quick

Assign Letter=Q

Exit
```

2. 应用 Install.wim 里的系统到 VHD/VHDX 文件里

完成“创建 VHD/VHDX 文件”后，可应用 Install.wim 里指定的索引号到指定的盘符，已设置：映像文件：D:\ISOTemp\Sources\Install.wim，索引号：1，应用到盘符：Q，命令行：

```
Expand-WindowsImage -ImagePath "D:\ISOTemp\Sources\install.wim" -ApplyPath "Q:\" -Index 1
```

3. 将原生启动 VHD 添加到现有 Windows 10/11 启动菜单中

3.1. 备份 BCD

使用 BCDedit 工具 /export 选项备份您的 BCD 存储。在命令提示符下，运行：bcdedit /export c:\bcdbackup

3.2. 复制一个现有的 Windows 10/11 安装启动项。然后修改副本，使其用作 VHD 启动项。在命令提示符下，运行：

```
bcdedit /copy {default} /d "VHD New Windows 11"
```

当 BCDedit 命令成功完成时，它会在命令提示符窗口中返回 {GUID} 作为输出。

3.3. 在上一个命令的命令提示符输出中找到 {GUID}。复制 GUID（包括大括号），以便在后续步骤中使用。

3.4. 设置 VHD 启动项的设备和操作系统设备选项，必须替换 {GUID}。在命令提示符下，运行：

```
bcdedit /set {default} device vhd="[D:]\OS.vhdx"

bcdedit /set {default} osdevice vhd="[D:]\OS.vhdx"
```

3.5. 可选项：

3.5.1. 将 VHD 的启动项设置为默认启动项。计算机重启后，启动菜单将显示计算机上所有 Windows 安装，并在操作系统选择倒计时结束后启动 VHD。在命令提示符下，输入：

```
bcdedit /default {guid}
```

3.5.2. 某些基于 x86 的系统需要内核的启动配置选项，以便检测某些硬件信息并成功地从 VHD 进行本地启动。在命令提示符下，输入：

```
bcdedit /set {guid} detecthal on
```

有关如何使用 BCDedit 工具的更多信息，请参阅此 Microsoft 网站。

D. 部署到虚拟机

常见的有 Windows 自带的 Hyper-V、VMware Workstation Pro、VirtualBox 等。

1. 启用 Hyper-V

- 确保您的系统符合要求：Windows 10/11 专业版或企业版，配备具有二级地址转换（SLAT）的 64 位处理器，并且至少有 4 GB 内存。Hyper-V 不适用于 Windows 家庭版。
- 启用 Hyper-V 后，您可以使用 Hyper-V 管理器或 PowerShell 命令开始创建虚拟机。
- Hyper-V 是内置于 Windows 的虚拟化平台，允许您创建和管理虚拟机。以下是在系统上启用 Hyper-V 的方法。

1.1. 使用 PowerShell

以管理员身份打开 Windows PowerShell：按 Win + S，输入“PowerShell”，右键点击，然后选择“以管理员身份运行”，运行以下命令以启用 Hyper-V：

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All
```

当系统提示时，输入 Y 重启计算机以完成安装，当系统提示时，重启计算机。

1.2. 在“启用或关闭 Windows 功能”里选择

1.2.1. 打开“启用或关闭 Windows 功能”对话框

1.2.1.1. 运行 OptionalFeatures 后弹出“启用或关闭 Windows 功能”。

1.2.1.2. 使用控制面板

- 打开控制面板：按下 Win 键 + S，输入“控制面板”，然后打开它。
- 导航到程序 > 程序和功能 > 启用或关闭 Windows 功能。

1.2.2. 打开后勾选 Hyper-V 复选框，然后点击确定。

1.2.3. 重启计算机以完成安装。

2. VMware Workstation Pro

官方网站 | https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion

3. VirtualBox

官方网站 | https://www.virtualbox.org

E. 进阶部署

1. 全自动安装

- 批量部署过程中，想实现全自动安装，需要修改应预答配置；
- 多硬盘

批量安装优先判断多少个硬盘并初始化，根据不同的硬盘需求去实现不同的解决方案。

2. 部署引擎

如果添加了部署引擎（Multilingual、YiSuite），在部署中，您可以“自定义”部署过程，下载模板：Yi.Engine.Deploy.Rule.iso，下载完成后，解压到任意磁盘，或在首次部署时，挂载 ISO 或修改 ISO 内容，了解更多：

- Multilingual | https://github.com/ilikeyi/Multilingual
- YiSuite | https://github.com/ilikeyi/YiSuite

2.1. 自定义部署 Office

下载模板 Yi.Engine.Deploy.Rule.iso 后，解压到任意磁盘，示例解压到：D:\Yi，选择版本后运行下载脚本：

2.1.1. Office 365

```
& "D:\Yi\Office\365\amd64\Download.Office.amd64.ps1"
& "D:\Yi\Office\365\amd64\Download.Office.x86.ps1"
```

2.1.2. Office 2024

```
& "D:\Yi\Office\2024\amd64\Download.Office.amd64.ps1"
& "D:\Yi\Office\2024\amd64\Download.Office.x86.ps1"
```

2.1.3. Office 2021

```
& "D:\Yi\Office\2021\amd64\Download.Office.amd64.ps1"
& "D:\Yi\Office\2021\amd64\Download.Office.x86.ps1"
```

安装前请下载 Office 安装包，下载完成后：按此顺序搜索：365、2024、2021；

发现了可用的安装包：首次部署会搜索 Yi.ps1 文件并执行 Office 安装进程；

手动运行：D:\Yi\Office\{ 365 | 2024 | 2021 }\{ amd64 | x86 }\Office\Install.Office.ps1 可进入安装进程。

可持续性：

自定义部署：在使用移动 U 盘，可将 Yi 目录保存到 U 盘分区，在安装系统过程中，一直保持连接状态，自动运行自定义部署策略。

2.2. 应预答用户方案

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。

2.2.1. 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. OOBE 交互式创建新用户

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。

2.2.2.1. 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. 替换

从开始处 <OOBE> 到 </OOBE>

```
<OOBE>
  <ProtectYourPC>3</ProtectYourPC>
  <HideEULAPage>true</HideEULAPage>
  <HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>
</OOBE>
```

章节 3 常见问题

A. 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:\ISOTEMP_Custom\Install\Install\Mount" -Discard
Dismount-WindowsImage -Path "D:\ISOTEMP_Custom\Install\WinRE\Mount" -Discard
Dismount-WindowsImage -Path "D:\ISOTEMP_Custom\Boot\Boot\Mount" -Discard
```

B. 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

C. 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 4 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'S 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：4.6

首次公开发行时间：4 / 2023

文档里包含的所有脚本，最后测试时间：1 / 2026

文档最近更新日期：1 / 2026

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