



## MICROSOFT WINDOWS 10

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”、“InBox Apps：添加、更新、标记”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

章节 1    部署映像

目录

章节 1	部署映像	第 4 页
A.	先决条件	第 4 页
II	ISO 工具	第 4 页
III	要求	第 4 页
1.	系统安装包	第 4 页
2.	语言包	第 4 页
2.1.	学习	第 4 页
2.2.	语言包：下载	第 5 页
2.3.	功能包：下载	第 5 页
3.	InBox Apps	第 5 页
IV	命令行	第 5 页
B.	语言包：提取	第 5 页
II	语言包：准备	第 5 页
III	语言包：提取方案	第 5 页
IV	执行提取命令	第 6 页
C.	自定义部署映像	第 12 页
II	自定义部署映像：Install.wim	第 12 页
1.	查看 Install.wim 详细信息	第 12 页
2.	指定挂载 Install 路径	第 12 页
3.	开始挂载 Install.wim	第 12 页
3.1.	自定义部署映像：WinRE.wim	第 12 页
3.1.1.	查看 WinRE.wim 详细信息	第 12 页
3.1.2.	指定挂载 WinRE.wim 路径	第 12 页
3.1.3.	开始挂载 WinRE.wim	第 12 页
3.1.4.	语言包	第 13 页

3.1.4.1.	语言包：添加	第 13 页
3.1.4.2.	组件：映像中已安装的所有包	第 14 页
3.1.5.	保存映像 WinRE.wim	第 15 页
3.1.6.	卸载映像 WinRE.wim	第 15 页
3.1.7.	重建 WinRE.wim 后，可缩小文件大小	第 15 页
3.1.8.	备份 WinRE.wim	第 16 页
3.1.9.	替换 Install.wim 映像内的 WinRE.wim	第 16 页
4.	语言包	第 16 页
4.1.	语言包：添加	第 16 页
4.2.	组件：映像中已安装的所有包	第 22 页
5.	InBox Apps	第 22 页
5.1.	InBox Apps：已安装	第 22 页
5.2.	删除已安装的所有预应用程序	第 23 页
5.3.	区域标记：添加方式	第 23 页
5.4.	InBox Apps：安装	第 24 页
5.5.	InBox Apps：优化	第 31 页
6.	累积更新	第 31 页
6.1.	功能启用包	第 32 页
6.2.	初始版本	第 32 页
6.3.	其它版本	第 32 页
6.4.	固化更新，可选项	第 33 页
6.4.1.	固化更新后清理组件，可选项	第 33 页
7.	部署引擎：添加，可选	第 33 页
8.	健康	第 33 页

9.	替换 WinRE.wim	第 34 页
10.	保存映像 Install.wim	第 34 页
11.	卸载映像 Install.wim	第 34 页
12.	重建 Install.wim 后可缩小文件大小	第 34 页
13.	如何批量替换 Install.wim 里的所有索引号里的 WinRE.wim	第 35 页
13.1.	获取 WimLib	第 35 页
13.2.	如何在 Install.wim 里提取和更新 WinRE.wim	第 35 页
III	自定义部署映像：boot.wim	第 36 页
1.	查看 Boot.wim 文件信息	第 36 页
2.	指定挂载 Boot.wim 路径	第 36 页
3.	开始挂载 Boot.wim	第 36 页
4.	语言包	第 36 页
4.1.	语言包：添加	第 36 页
4.2.	组件：映像中已安装的所有包	第 38 页
4.3.	语言包：同步到 ISO 安装程序	第 38 页
4.4.	重新生成 Lang.ini	第 38 页
4.4.1.	重新生成已挂载目录 lang.ini	第 38 页
4.4.2.	重新生成 lang.ini 后，同步到安装程序	第 39 页
5.	保存映像 Boot.wim	第 39 页
6.	卸载映像 Boot.wim	第 39 页
IV	部署引擎	第 39 页
1.	添加方式	第 39 页
2.	部署引擎：进阶	第 42 页
D.	生成 ISO	第 45 页

章节 1 部署映像

A. 先决条件

II ISO 工具

准备一款可编辑 ISO 文件的软件，例如：[PowerISO](#)、[DAEMON Tools](#)、[ISO Workshop](#)；

III 要求

1. 系统安装包

关键词：[迭代](#)、[跨版本](#)、[大版本](#)、[累积更新](#)、[初始版本](#)

1.1. 说明

- 1.1.1. 每版本更新时请重新制作镜像，例如从 21H1 跨越到 22H2 时，应避免出现其它兼容性问题请勿在旧镜像基础上制作；
- 1.1.2. 该条例已经在某些 OEM 厂商，通过各种形式向封装师明确传达了该法令，不允许直接从迭代版本里直接升级；
- 1.1.3. 制作中请使用“初始版本”、“开发者版”制作。微软官方文档里曾短暂的出现过在制作中必须使用初始版本，后来这句在官方文档里却被删除了；
- 1.1.4. 安装语言包后，必须重新添加累积更新（可同一版本号），不添加累积更新会出现“乱码”、“界面闪退”等问题。
- 1.1.5. 进化过程：Windows 10 22H2, Build 19045.2006 + KB5027215 = OS Build 19045.3086

1.2. 准备下载初始版本或开发者版本

- 1.2.1. [en-us\\_windows\\_10\\_business\\_editions\\_version\\_22h2\\_x64\\_dvd\\_8cf17b79.iso](#)
- 1.2.2. [en-us\\_windows\\_10\\_consumer\\_editions\\_version\\_22h2\\_x64\\_dvd\\_8da72ab3.iso](#)

1.3. 示例下载 [en-us\\_windows\\_10\\_business\\_editions\\_version\\_22h2\\_x64\\_dvd\\_8cf17b79.iso](#) 后，解压到：[D:\en-us\\_windows\\_10\\_business\\_editions\\_version\\_22h2\\_x64\\_dvd\\_8cf17b79](#)

1.4. 解压完成后，将目录 [en-us\\_windows\\_10\\_business\\_editions\\_version\\_22h2\\_x64\\_dvd\\_8cf17b79](#) 更改为 [D:\OS\\_10](#)

1.5. 所有脚本、所有路径，已默认设置为 [D:\OS\\_10](#) 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. [语言概述](#)

2.1.2.     [将语言添加到 Windows 10 映像](#)

2.1.3.     [语言和区域按需功能 \(FOD\)](#)

2.2.     语言包：下载

[https://software-download.microsoft.com/download/pr/19041.1.191206-1406.vb\\_release\\_CLIENTLANGPACKDVD\\_OEM\\_MULTI.iso](https://software-download.microsoft.com/download/pr/19041.1.191206-1406.vb_release_CLIENTLANGPACKDVD_OEM_MULTI.iso)

2.3.     功能包：下载

X64: [https://software-download.microsoft.com/download/pr/19041.1.191206-1406.vb\\_release\\_amd64fre\\_FOD-PACKAGES\\_OEM\\_PT1\\_amd64fre\\_MULTI.iso](https://software-download.microsoft.com/download/pr/19041.1.191206-1406.vb_release_amd64fre_FOD-PACKAGES_OEM_PT1_amd64fre_MULTI.iso)

3.     InBox Apps

[https://software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/19041.3031.230508-1728.vb\\_release\\_svc\\_prod3\\_amd64fre\\_InboxApps.iso](https://software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/19041.3031.230508-1728.vb_release_svc_prod3_amd64fre_InboxApps.iso)

IV     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”，设置 PowerShell 执行策略：绕过，PS 命令行：  
  
`Set-ExecutionPolicy -ExecutionPolicy Bypass -Force`
3.     在本文中，绿色部分属于 PS 命令行，请复制后，粘贴到“Terminal”对话框，按回车键（Enter）后开始运行；
4.     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:\Yi.Solutions\_Encapsulation\_SIP.ps1"`

B.     语言包：提取

II     语言包：准备

1.     语言包：挂载

挂载 [19041.1.191206-1406.vb\\_release\\_CLIENTLANGPACKDVD\\_OEM\\_MULTI.iso](#) 或解压到任意位置；

2.     功能包：挂载

请正确选择架构版本，提取错误的语言包，安装会报错误。

X64: [19041.1.191206-1406.vb\\_release\\_amd64fre\\_FOD-PACKAGES\\_OEM\\_PT1\\_amd64fre\\_MULTI.iso](#)

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

3. 架构: x64

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/\\_Learn/Packaging.tutorial/OS.10/22H2/Expand/Extract.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/Extract.ps1)

• 复制代码

```
$ISO = "Auto"

$SaveTo = "D:\OS_10_Custom"

$Extract_language_Pack = @(

    @{ Tag = "zh-CN"; Arch = "AMD64"; Act = "Add"; Scope = @"Install\Install"; "Install\WinRE"; "Boot\Boot" ) }

    @{ Tag = "en-US"; Arch = "AMD64"; Act = "Del"; Scope = @"Install\Install"; "Install\WinRE"; "Boot\Boot" }

)

Function Extract_Language

{

    param($NewArch, $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 = @(

            "Microsoft-Windows-LanguageFeatures-Fonts-{DiyLang}-Package~31bf3856ad364e35~AMD64~~.cab"

            "{ARCHC}\langpacks\Microsoft-Windows-Client-Language-Pack_x64_{Lang}.cab"

            "Microsoft-Windows-LanguageFeatures-Basic-{Lang}-Package~31bf3856ad364e35~AMD64~~.cab"

            "Microsoft-Windows-LanguageFeatures-Handwriting-{Lang}-Package~31bf3856ad364e35~AMD64~~.cab"

            "Microsoft-Windows-LanguageFeatures-OCR-{Lang}-Package~31bf3856ad364e35~AMD64~~.cab"

            "Microsoft-Windows-LanguageFeatures-Speech-{Lang}-Package~31bf3856ad364e35~AMD64~~.cab"

            "Microsoft-Windows-LanguageFeatures-TextToSpeech-{Lang}-Package~31bf3856ad364e35~AMD64~~.cab"

            "Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

            "Microsoft-Windows-MSPaint-FoD-Package~31bf3856ad364e35~{ARCH}~{Lang}~.cab"

            "Microsoft-Windows-Notepad-FoD-Package~31bf3856ad364e35~{ARCH}~{Lang}~.cab"

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

            "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

            "Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

            "Microsoft-Windows-Printing-WFS-FoD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

            "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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


```

```
"Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

"Microsoft-Windows-InternationalFeatures-{Specific}-Package~31bf3856ad364e35~amd64~~.cab"

)

}

@{

    Path = "Install\WinRE"

    Rule = @{

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    }

}

@{

    Path = "Boot\Boot"

    Rule = @{

        "Windows Preinstallation Environment\x64\WinPE_OC\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-CLIENT_{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

$NewArchC = $NewArch.Replace("AMD64", "x64")

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).Replace("{ARCH}", $NewArch).Replace("{ARCHC}", $NewArchC)

            }

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

        }

    }

}

}

}

Foreach ($item in $Extract_language_Pack) { Extract_Language -Act $item.Act -NewLang $item.Tag -NewArch $item.Arch -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_10\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 路径

```
New-Item -Path "D:\OS_10_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_10\sources\install.wim" -Index "1" -Path "D:\OS_10_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_10_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_10_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 "D:\OS_10_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim" -
Index "1" -Path "D:\OS_10_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/\\_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/WinRE/WinRE.Instl.lang.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/WinRE/WinRE.Instl.lang.ps1)

- 复制代码

```
$Mount = "D:\OS_10_Custom\Install\WinRE\Mount"

$Sources = "D:\OS_10_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 = @(

    @{ 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) {

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

    ForEach ($Component in $InitL_install_Language_Component) {

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

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

            Write-host "  Language pack file: " -NoNewline; Write-host "$($Sources)\$($Rule.File)" -
ForegroundColor Green

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

            try{

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

                Write-host "Finish" -ForegroundColor Green

            } catch {

                Write-host "Failed" -ForegroundColor Red

                Write-host "  $($_) " -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_10_Custom\Install\WinRE\Mount" | Out-GridView
```

3.1.4.2.2. 导出到 Csv

```
$SaveTo = "D:\OS_10_Custom\Install\WinRE\Report.$(Get-Date -Format
"yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS_10_Custom\Install\WinRE\Mount" | Export-CSV -NoType
-Path $SaveTo

Write-host $SaveTo -ForegroundColor Green
```

3.1.5. 保存映像 WinRE.wim

```
Save-WindowsImage -Path "D:\OS_10_Custom\Install\WinRE\Mount"
```

3.1.6. 卸载映像 WinRE.wim

关闭所有可能正在访问映像中文件的应用程序，包括文件资源管理器。

```
Dismount-WindowsImage -Path "D:\OS_10_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/\\_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/WinRE/WinRE.Rebuild.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/WinRE/WinRE.Rebuild.ps1)

- 复制代码

```
$FileName = "D:\OS_10_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/\\_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/WinRE/WinRE.Backup.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/WinRE/WinRE.Backup.ps1)

- 复制代码

```
$WimLibPath = "D:\OS_10_Custom\Install\Install\Update\Winlib"  
  
$FileName = "D:\OS_10_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/\\_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/Install.Instl.lang.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/Install.Instl.lang.ps1)

- 复制代码

```
Function Language_Install  
  
{
```

```

param($Mount, $Sources, $Lang)

$InitL_install_Language_Component = @()

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

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

} else {

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

    return

}

$Script:Init_Folder_All_File = @()

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

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

        $Script:Init_Folder_All_File += $_.FullName

    }

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

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

        ForEach ($item in $Script:Init_Folder_All_File) {

            Write-host "  $($item)"

        }

    } else {

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

        return

    }

} else {

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

    return

}

$Script:Init_Folder_All_File_Match_Done = @()

$Script:Init_Folder_All_File_Exclude = @()

$Script:Search_File_Order = @(

    @{

        Name = "Fonts"

        Description = "Fonts"

        Rule = @(

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


```

```
)

}

@{

    Name = "Basic"

    Description = "Basic"

    Rule = @(

        @{ Match_Name = "*LanguageFeatures-Basic*"; IsMatch = "Yes"; Capability = "Language.Basic~~~$($Lang)~0.0.1.0"; }

        @{ Match_Name = "*Client*Language*Pack*"; IsMatch = "Yes"; Capability = "Language.Basic~~~$($Lang)~0.0.1.0"; }

    )

}

@{

    Name = "OCR"

    Description = "Optical character recognition"

    Rule = @(

        @{ Match_Name = "*LanguageFeatures-OCR*"; IsMatch = "Yes"; Capability = "Language.OCR~~~$($Lang)~0.0.1.0"; }

    )

}

@{

    Name = "Handwriting"

    Description = "Handwriting recognition"

    Rule = @(

        @{ Match_Name = "*LanguageFeatures-Handwriting*"; IsMatch = "Yes"; Capability =
"Language.Handwriting~~~$($Lang)~0.0.1.0"; }

    )

}

@{

    Name = "TextToSpeech"

    Description = "Text-to-speech"

    Rule = @(

        @{ Match_Name = "*LanguageFeatures-TextToSpeech*"; IsMatch = "Yes"; Capability =
"Language.TextToSpeech~~~$($Lang)~0.0.1.0"; }

    )

}

@{

    Name = "Speech"
```

```

    Description = "Speech recognition"

    Rule = @(

        @{ Match_Name = "*LanguageFeatures-Speech*"; IsMatch = "Yes"; Capability = "Language.Speech~~~$($Lang)~0.0.1.0"; }

    )

}

@{

    Name = "RegionSpecific"

    Description = "Other region-specific requirements"

    Rule = @(

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

    )

}

@{

    Name = "Retail"

    Description = "Retail demo experience"

    Rule = @(

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

    )

}

@{

    Name = "Features_On_Demand"

    Description = "Features on demand"

    Rule = @(

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

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

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

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

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

        @{ Match_Name = "*MediaPlayer*amd64*"; IsMatch = "Yes"; Capability = "Media.WindowsMediaPlayer~~~~0.0.12.0"; }

        @{ Match_Name = "*MediaPlayer*wow64*"; IsMatch = "Yes"; Capability = "Media.WindowsMediaPlayer~~~~0.0.12.0"; }

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

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

        @{ Match_Name = "*Printing*PMCPPC*amd64*"; IsMatch = "Yes"; Capability = "Print.Management.Console~~~~0.0.1.0"; }
```

```

@{ Match_Name = "*Printing*WFS*amd64*"; IsMatch = "Yes"; Capability = "Print.Management.Console~~~~0.0.1.0"; }

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

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

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

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

@{ Match_Name = "*WMIC*FoD*Package*amd64*"; IsMatch = "Yes"; Capability = "WMIC~~~~"; }

@{ Match_Name = "*WMIC*FoD*Package*wow64*"; IsMatch = "Yes"; Capability = "WMIC~~~~"; }

)

}

)

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

ForEach ($Wildcard in $Script:Init_Folder_All_File) {

    ForEach ($item in $Script:Search_File_Order) {

        ForEach ($NewRule in $item.Rule) {

            if ($Wildcard -like "*$( $NewRule.Match_Name )*") {

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

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

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

                $TempSave = @{ Match_Name = $NewRule.Match_Name; Capability = $NewRule.Capability; FileName = $Wildcard }

                $new = $OSDefaultUser + $TempSave

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

                    ForEach ($Component in $Initl_install_Language_Component) {

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

                            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

                        }

                    }

                } else {

                    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

                }
            }
        }
    }
}

```

```

    }

}

}

}

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

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

```

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

    try {

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

        Write-host "Finish`n" -ForegroundColor Green

    } catch {

        Write-host "Failed" -ForegroundColor Red

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

    }

} else {

    Write-host "Does not exist`n"

}

}

} else {

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

}

}

}
```

Language\_Install -Mount "D:\OS\_10\_Custom\Install\Install\Mount" -Sources "D:\OS\_10\_Custom\Install\Install\Language\Add" -Lang "zh-CN"

#### 4.2. 组件：映像中已安装的所有包

4.2.1. [查看](#)

```
Get-WindowsPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Out-GridView
```

#### 4.2.2. 导出到 Csv

```
$SaveTo = "D:\OS_10_Custom\Install\Install\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green
```

## 5. InBox Apps

### 5.1. InBox Apps: 已安装

5.1.1. [查看](#)

```
Get-AppXProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Out-GridView
```

5.1.2. 导出到 Csv

```
$SaveTo = "D:\OS_10_Custom\Install\Install\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-AppXProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green
```

5.2. 删除已安装的所有预应用程序

- Install.InBox.Appx.Clear.all.ps1
  - [\Expand\Install\Install.InBox.Appx.Clear.all.ps1](#)
  - [https://github.com/ilikeyi/solutions/blob/main/\\_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/Install.InBox.Appx.Clear.all.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/Install.InBox.Appx.Clear.all.ps1)

- 复制代码

```
Get-AppXProvisionedPackage -path "D:\OS_10_Custom\Install\Install\Mount" -ErrorAction SilentlyContinue | ForEach-Object {

    Write-host "`n $($_.DisplayName)"

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

    try{

        Remove-AppxProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" -PackageName $_.PackageName -ErrorAction
        SilentlyContinue | Out-Null

        Write-host "Finish" -ForegroundColor Green

    } catch {

        Write-host "Failed" -ForegroundColor Red

    }

}
```

5.3. 区域标记：添加方式

5.3.1. 执行“语言包：添加”

5.3.2. 安装“本地语言体验包（LXPs）”

微软官方向 Windows 10 提供了本地语言体验包（LXPS）安装文件，Windows 11 不再提供，想获取：

5.3.2.1. 使用“Windows 本地语言体验包（LXPs）下载器”下载

了解：<https://github.com/ilikeyi/LXPs>

下载后保存到：D:\OS\_10\_Custom\Install\Install\InBox.Appx，文件格式：[LanguageExperiencePack.zh-CN.Neutral.Appx](#)



5.3.2.2. 手动下载

5.3.2.2.1. 区域

下载区域：zh-CN，应用程序 ID：9NRMNT6GMZ70，商店连接：  
<https://www.microsoft.com/store/productId/9NRMNT6GMZ70>

5.3.2.2.2. 打开网站：<https://store.rg-adguard.net>

5.3.2.2.2.1. 搜索关键词：  
  
<https://www.microsoft.com/store/productId/9NRMNT6GMZ70>

5.3.2.2.2.2. 网页内搜索 22621 内容，搜索结果：  
  
[Microsoft.LanguageExperiencePackzh-CN\\_22621.\\*.neutral\\_8wekyb3d8bbwe.appx](#)

5.3.2.2.2.3. 下载后保存到 D:\OS\_10\_Custom\Install\Install\InBox.Appx 目录里，重命名：[LanguageExperiencePack.zh-cn.Neutral.Appx](#)

5.3.2.3. 执行安装命令安装本地语言体验包（LXPs）

了解区域标记添加方式后，获得 [LanguageExperiencePack.zh-cn.Neutral](#) 后，执行安装命令：

```
Add-AppxProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" -PackagePath "D:\OS_10_Custom\Install\Install\InBox.appx\LanguageExperiencePack.zh-cn.Neutral.appx" -SkipLicense
```

5.3.2.4. InBox Apps：已安装的应用程序包

5.3.2.4.1. 查看

```
Get-AppXProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Out-GridView
```

5.3.2.4.2. 导出到 Csv

```
$SaveTo = "D:\OS_10_Custom\Install\Install\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-AppXProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green
```

5.4. InBox Apps：安装

5.4.1. 挂载或解压 InBox Apps 安装文件

挂载 [19041.3031.230508-1728.vb\\_release\\_svc\\_prod3\\_amd64fre\\_InboxApps.iso](#) 或解压到任意位置；

#### 5.4.2. 执行安装命令后安装 InBox Apps 到: Install.wim

- **Auto** = 自动搜索本地所有磁盘，默认；
- 自定义路径，例如指定为 F 盘: **\$ISO = "F:\packages"**
- Install.Inst.InBox.Appx.ps1
  - [\Expand\Install\Install.Inst.InBox.Appx.ps1](#)
  - [https://github.com/ilikeyi/solutions/blob/main/\\_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/Install.Inst.InBox.Appx.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/Install.Inst.InBox.Appx.ps1)
- 复制代码

```
$ISO = "Auto"

$Mount = "D:\OS_10_Custom\Install\Install\Mount"

try{

    Write-host "`n  Offline image version: " -NoNewline

    $Current_Edition_Version = (Get-WindowsEdition -Path $Mount).Edition

    Write-Host $Current_Edition_Version -ForegroundColor Green

} catch {

    Write-Host "Error" -ForegroundColor Red

    Write-Host "  $($_) " -ForegroundColor Yellow

    return

}

$Pre_Config_Rules = @(

    @{

        Name = @("Core"; "CoreN"; "CoreSingleLanguage");

        Apps = @(

            "Microsoft.UI.Xaml.2.3"; "Microsoft.UI.Xaml.2.4"; "Microsoft.UI.Xaml.2.7"; "Microsoft.NET.Native.Framework.2.2";
            "Microsoft.NET.Native.Runtime.2.2"; "Microsoft.VCLibs.140.00"; "Microsoft.VCLibs.140.00.UWPDesktop";
            "Microsoft.HEIFImageExtension"; "Microsoft.HEVCVideoExtension"; "Microsoft.SecHealthUI";
            "Microsoft.VP9VideoExtensions"; "Microsoft.WebplImageExtension"; "Microsoft.WindowsStore"; "Microsoft.GamingApp";
            "Microsoft.Sticky.Notes"; "Microsoft.Paint"; "Microsoft.PowerAutomateDesktop"; "Microsoft.ScreenSketch";
            "Microsoft.WindowsNotepad"; "Microsoft.WindowsTerminal"; "Clipchamp.Clipchamp"; "Microsoft.Solitaire.Collection";
            "Microsoft.WindowsAlarms"; "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsMaps"; "Microsoft.ZuneMusic";
            "Microsoft.BingNews"; "Microsoft.DesktopAppInstaller"; "Microsoft.WindowsCamera"; "Microsoft.Getstarted";
            "Microsoft.Cortana"; "Microsoft.BingWeather"; "Microsoft.GetHelp"; "Microsoft.MicrosoftOfficeHub"; "Microsoft.People";
            "Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.WebMediaExtensions"; "Microsoft.Windows.Photos";
            "Microsoft.WindowsCalculator"; "Microsoft.Windows.CommunicationsApps"; "Microsoft.WindowsSoundRecorder";
            "Microsoft.Xbox.TCUI"; "Microsoft.XboxGameOverlay"; "Microsoft.XboxGamingOverlay"; "Microsoft.XboxIdentityProvider";
            "Microsoft.XboxSpeechToTextOverlay";

            "Microsoft.YourPhone"; "Microsoft.ZuneVideo"; "MicrosoftCorporationII.QuickAssist";
            "MicrosoftWindows.Client.WebExperience"; "Microsoft.RawImageExtension"; "MicrosoftCorporationII.MicrosoftFamily";
```

```

    )

}

@{

    Name = @("Education"; "Professional"; "ProfessionalEducation"; "ProfessionalWorkstation"; "Enterprise";
"loTEnterprise"; "ServerRdsh");

    Apps = @(

        "Microsoft.UI.Xaml.2.3"; "Microsoft.UI.Xaml.2.4"; "Microsoft.UI.Xaml.2.7"; "Microsoft.NET.Native.Framework.2.2";
"Microsoft.NET.Native.Runtime.2.2"; "Microsoft.VCLibs.140.00"; "Microsoft.VCLibs.140.00.UWPDesktop";
"Microsoft.HEIFImageExtension"; "Microsoft.HEVCVideoExtension"; "Microsoft.SecHealthUI";
"Microsoft.VP9VideoExtensions"; "Microsoft.WebplImageExtension"; "Microsoft.WindowsStore"; "Microsoft.GamingApp";
"Microsoft.Sticky.Notes"; "Microsoft.Paint"; "Microsoft.PowerAutomateDesktop"; "Microsoft.ScreenSketch";
"Microsoft.WindowsNotepad"; "Microsoft.WindowsTerminal"; "Clipchamp.Clipchamp"; "Microsoft.Solitaire.Collection";
"Microsoft.WindowsAlarms"; "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsMaps"; "Microsoft.ZuneMusic";
"Microsoft.BingNews"; "Microsoft.DesktopAppInstaller"; "Microsoft.WindowsCamera"; "Microsoft.Getstarted";
"Microsoft.Cortana"; "Microsoft.BingWeather"; "Microsoft.GetHelp"; "Microsoft.MicrosoftOfficeHub"; "Microsoft.People";
"Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.WebMediaExtensions"; "Microsoft.Windows.Photos";
"Microsoft.WindowsCalculator"; "Microsoft.Windows.CommunicationsApps"; "Microsoft.WindowsSoundRecorder";
"Microsoft.Xbox.TCUI"; "Microsoft.XboxGameOverlay"; "Microsoft.XboxGamingOverlay"; "Microsoft.XboxIdentityProvider";
"Microsoft.XboxSpeechToTextOverlay"; "Microsoft.YourPhone"; "Microsoft.ZuneVideo";
"MicrosoftCorporationII.QuickAssist"; "MicrosoftWindows.Client.WebExperience"; "Microsoft.RawImageExtension";

    )

}

@{

    Name = @("EnterpriseN"; "EnterpriseGN"; "EnterpriseSN"; "ProfessionalN"; "EducationN"; "ProfessionalWorkstationN";
"ProfessionalEducationN"; "CloudN"; "CloudEN"; "CloudEditionN"; "CloudEditionLN"; "StarterN");

    Apps = @(

        "Microsoft.UI.Xaml.2.3"; "Microsoft.UI.Xaml.2.4"; "Microsoft.UI.Xaml.2.7"; "Microsoft.NET.Native.Framework.2.2";
"Microsoft.NET.Native.Runtime.2.2"; "Microsoft.VCLibs.140.00"; "Microsoft.VCLibs.140.00.UWPDesktop";
"Microsoft.SecHealthUI"; "Microsoft.WindowsStore"; "Microsoft.Sticky.Notes"; "Microsoft.Paint";
"Microsoft.PowerAutomateDesktop"; "Microsoft.ScreenSketch"; "Microsoft.WindowsNotepad";
"Microsoft.WindowsTerminal"; "Clipchamp.Clipchamp"; "Microsoft.Solitaire.Collection"; "Microsoft.WindowsAlarms";
"Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsMaps"; "Microsoft.BingNews"; "Microsoft.DesktopAppInstaller";
"Microsoft.WindowsCamera"; "Microsoft.Getstarted"; "Microsoft.Cortana"; "Microsoft.BingWeather"; "Microsoft.GetHelp";
"Microsoft.MicrosoftOfficeHub"; "Microsoft.People"; "Microsoft.StorePurchaseApp"; "Microsoft.Todos";
"Microsoft.Windows.Photos"; "Microsoft.WindowsCalculator"; "Microsoft.Windows.CommunicationsApps";
"Microsoft.XboxGameOverlay"; "Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay";
"Microsoft.YourPhone"; "MicrosoftCorporationII.QuickAssist"; "MicrosoftWindows.Client.WebExperience";

    )

}

)

$Allow_Install_App = @()

ForEach ($item in $Pre_Config_Rules) {

    if ($item.Name -contains $Current_Edition_Version) {

        Write-host "`n  Match to: "-NoNewline

```

```

        Write-host $Current_Edition_Version -ForegroundColor Green

        $Allow_Install_App = $item.Apps

        break
    }
}

Write-host "`n The app to install ( $($Allow_Install_App.Count) item )" -ForegroundColor Yellow

Write-host " ${'-' * 80}"

ForEach ($item in $Allow_Install_App) {

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

}

$InBoxApps = @(

    @{ Name = "Microsoft.UI.Xaml.2.3"; File = "Microsoft.UI.Xaml.x64.2.3.appx"; License = ""; }

    @{ Name = "Microsoft.UI.Xaml.2.4"; File = "Microsoft.UI.Xaml.x64.2.4.appx"; License = ""; }

    @{ Name = "Microsoft.UI.Xaml.2.7"; File = "Microsoft.UI.Xaml.x64.2.7.appx"; License = ""; }

    @{ Name = "Microsoft.NET.Native.Framework.2.2"; File = "Microsoft.NET.Native.Framework.x64.2.2.appx"; License = ""; }

    @{ Name = "Microsoft.NET.Native.Runtime.2.2"; File = "Microsoft.NET.Native.Runtime.x64.2.2.appx"; License = ""; }

    @{ Name = "Microsoft.VCLibs.140.00"; File = "Microsoft.VCLibs.x64.14.00.appx"; License = ""; }

    @{ Name = "Microsoft.VCLibs.140.00.UWPDesktop"; File = "Microsoft.VCLibs.x64.14.00.UWPDesktop.appx"; License =
""; }

    @{ Name = "Microsoft.WindowsStore"; File = "Microsoft.WindowsStore_8wekyb3d8bbwe.msixbundle"; License =
"Microsoft.WindowsStore_8wekyb3d8bbwe.xml"; }

    @{ Name = "Microsoft.HEIFImageExtension"; File = "Microsoft.HEIFImageExtension_8wekyb3d8bbwe.appxbundle";
License = "Microsoft.HEIFImageExtension_8wekyb3d8bbwe.xml"; }

    @{ Name = "Microsoft.HEVCVideoExtension"; File = "Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.x64.appx"; License =
"Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.x64.xml"; }

    @{ Name = "Microsoft.SecHealthUI"; File = "Microsoft.SecHealthUI_8wekyb3d8bbwe.x64.appx"; License =
"Microsoft.SecHealthUI_8wekyb3d8bbwe.x64.xml"; }

    @{ Name = "Microsoft.VP9VideoExtensions"; File = "Microsoft.VP9VideoExtensions_8wekyb3d8bbwe.x64.appx"; License =
"Microsoft.VP9VideoExtensions_8wekyb3d8bbwe.x64.xml"; }

    @{ Name = "Microsoft.WebplImageExtension"; File = "Microsoft.WebplImageExtension_8wekyb3d8bbwe.x64.appx";
License = "Microsoft.WebplImageExtension_8wekyb3d8bbwe.x64.xml"; }

    @{ Name = "Microsoft.GamingApp"; File = "Microsoft.GamingApp_8wekyb3d8bbwe.msixbundle"; License =
"Microsoft.GamingApp_8wekyb3d8bbwe.xml"; }

    @{ Name = "Microsoft.Sticky.Notes"; File = "Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe.msixbundle"; License =
"Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe.xml"; }

    @{ Name = "Microsoft.Paint"; File = "Microsoft.Paint_8wekyb3d8bbwe.msixbundle"; License =
"Microsoft.Paint_8wekyb3d8bbwe.xml"; }

    @{ Name = "Microsoft.PowerAutomateDesktop"; File = "Microsoft.PowerAutomateDesktop_8wekyb3d8bbwe.msixbundle";
License = "Microsoft.PowerAutomateDesktop_8wekyb3d8bbwe.xml"; }

```

```
@{ Name = "Microsoft.ScreenSketch"; File = "Microsoft.ScreenSketch_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.ScreenSketch_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsNotepad"; File = "Microsoft.WindowsNotepad_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.WindowsNotepad_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsTerminal"; File = "Microsoft.WindowsTerminal_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.WindowsTerminal_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Clipchamp.Clipchamp"; File = "Clipchamp.Clipchamp_yxz26nhyzhsrt.msixbundle"; License =  
"Clipchamp.Clipchamp_yxz26nhyzhsrt.xml"; }
```

```
@{ Name = "Microsoft.Solitaire.Collection"; File = "Microsoft.MicrosoftSolitaireCollection_8wekyb3d8bbwe.msixbundle";  
License = "Microsoft.MicrosoftSolitaireCollection_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsAlarms"; File = "Microsoft.WindowsAlarms_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.WindowsAlarms_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsFeedbackHub"; File = "Microsoft.WindowsFeedbackHub_8wekyb3d8bbwe.msixbundle";  
License = "Microsoft.WindowsFeedbackHub_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsMaps"; File = "Microsoft.WindowsMaps_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.WindowsMaps_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.ZuneMusic"; File = "Microsoft.ZuneMusic_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.ZuneMusic_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "MicrosoftCorporationII.MicrosoftFamily"; File =  
"MicrosoftCorporationII.MicrosoftFamily_8wekyb3d8bbwe.msixbundle"; License =  
"MicrosoftCorporationII.MicrosoftFamily_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.BingNews"; File = "Microsoft.BingNews_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.BingNews_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.DesktopAppInstaller"; File = "Microsoft.DesktopAppInstaller_8wekyb3d8bbwe.msixbundle";  
License = "Microsoft.DesktopAppInstaller_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsCamera"; File = "Microsoft.WindowsCamera_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.WindowsCamera_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Getstarted"; File = "Microsoft.Getstarted_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.Getstarted_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Cortana"; File = "Microsoft.CortanaApp_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.CortanaApp_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.BingWeather"; File = "Microsoft.BingWeather_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.BingWeather_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.GetHelp"; File = "Microsoft.GetHelp_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.GetHelp_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.MicrosoftOfficeHub"; File = "Microsoft.MicrosoftOfficeHub_8wekyb3d8bbwe.appxbundle"; License  
= "Microsoft.MicrosoftOfficeHub_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.People"; File = "Microsoft.People_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.People_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.StorePurchaseApp"; File = "Microsoft.StorePurchaseApp_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.StorePurchaseApp_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Todos"; File = "Microsoft.Todos_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.Todos_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WebMediaExtensions"; File = "Microsoft.WebMediaExtensions_8wekyb3d8bbwe.appxbundle";  
License = "Microsoft.WebMediaExtensions_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Windows.Photos"; File = "Microsoft.Windows.Photos_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.Windows.Photos_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsCalculator"; File = "Microsoft.WindowsCalculator_8wekyb3d8bbwe.appxbundle"; License  
= "Microsoft.WindowsCalculator_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Windows.CommunicationsApps"; File =  
"Microsoft.WindowsCommunicationsApps_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.WindowsCommunicationsApps_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsSoundRecorder"; File =  
"Microsoft.WindowsSoundRecorder_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.WindowsSoundRecorder_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Xbox.TCUI"; File = "Microsoft.Xbox.TCUI_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.Xbox.TCUI_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.XboxGameOverlay"; File = "Microsoft.XboxGameOverlay_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.XboxGameOverlay_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.XboxGamingOverlay"; File = "Microsoft.XboxGamingOverlay_8wekyb3d8bbwe.appxbundle"; License  
= "Microsoft.XboxGamingOverlay_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.XboxIdentityProvider"; File = "Microsoft.XboxIdentityProvider_8wekyb3d8bbwe.appxbundle";  
License = "Microsoft.XboxIdentityProvider_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.XboxSpeechToTextOverlay"; File =  
"Microsoft.XboxSpeechToTextOverlay_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.XboxSpeechToTextOverlay_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.YourPhone"; File = "Microsoft.YourPhone_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.YourPhone_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.ZuneVideo"; File = "Microsoft.ZuneVideo_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.ZuneVideo_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "MicrosoftCorporationII.QuickAssist"; File =  
"MicrosoftCorporationII.QuickAssist_8wekyb3d8bbwe.appxbundle"; License =  
"MicrosoftCorporationII.QuickAssist_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "MicrosoftWindows.Client.WebExperience"; File =  
"MicrosoftWindows.Client.WebExperience_cw5n1h2txyewy.appxbundle"; License =  
"MicrosoftWindows.Client.WebExperience_cw5n1h2txyewy.xml"; }
```

```
@{ Name = "Microsoft.RawImageExtension"; File = "Microsoft.RawImageExtension_8wekyb3d8bbwe.appxbundle"; License  
= "Microsoft.RawImageExtension_8wekyb3d8bbwe.xml"; }
```

```
)
```

```
Function Install_Appx
```

```
{
```

```
param($File, $License)
```

```
Write-host " ${'-' * 80}"
```

```
Write-host " Installing: " -NoNewline; Write-host $File -ForegroundColor Yellow
```

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

```

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

    Write-host "  License: " -NoNewline; Write-host $License -ForegroundColor Yellow

    Write-host "  With License".PadRight(22) -NoNewline -ForegroundColor Green

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

    try {

        Add-AppxProvisionedPackage -Path $Mount -PackagePath $File -LicensePath $License -ErrorAction SilentlyContinue
    } | Out-Null

    Write-Host "Done" -ForegroundColor Green

    } catch {

        Write-Host "Failed" -ForegroundColor Red

        Write-Host "  $($_) " -ForegroundColor Yellow

    }

} else {

    Write-host "  No License".PadRight(22) -NoNewline -ForegroundColor Red

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

    try {

        Add-AppxProvisionedPackage -Path $Mount -PackagePath $File -SkipLicense -ErrorAction SilentlyContinue | Out-
Null

        Write-Host "Done" -ForegroundColor Green

    } catch {

        Write-Host "Failed" -ForegroundColor Red

        Write-Host "  $($_) " -ForegroundColor Yellow

    }

}

} else {

    Write-host "  The installation package does not exist" -ForegroundColor Red

}

}

ForEach ($Rule in $InBoxApps) {

    Write-host "`n  Name: " -NoNewline

    Write-host $Rule.Name -ForegroundColor Yellow

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

    if($Allow_Install_App -contains $Rule.Name) {

        Write-host "  Search for apps: " -NoNewline

        Write-host $Rule.File -ForegroundColor Yellow

```

```

Write-host " Search for License: " -NoNewLine

Write-host $Rule.File -ForegroundColor Yellow

if ($ISO -eq "Auto"){

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

        $AppPath = Join-Path -Path $_.Root -ChildPath "packages\${$Rule.File}" -ErrorAction SilentlyContinue

        $LicensePath = Join-Path -Path $_.Root -ChildPath "packages\${$Rule.License}" -ErrorAction SilentlyContinue

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

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

            Write-host " Discover apps: " -NoNewLine; Write-host $AppPath -ForegroundColor Green

            if (Test-Path $LicensePath -PathType Leaf) {; Write-host " Discover License: " -NoNewLine

                Write-host $LicensePath -ForegroundColor Green

            } else {

                Write-host " License: " -NoNewLine

                Write-host "Not found" -ForegroundColor Red

            }

            Install_Appx -File $AppPath -License $LicensePath

            return

        }

    }

} else {

    Install_Appx -File "$($ISO)\${$Rule.File}" -License "$($ISO)\${$Rule.License}"

}

} else {

    Write-host " Skip the installation" -ForegroundColor Red

}

}

```

#### 5.5. InBox Apps：优化

在安装应用后，应优化预配 Appx 包，通过用硬链接替换相同的文件来减少应用的磁盘使用量，仅针对脱机映像。

```
Dism /Image:"D:\OS_10_Custom\Install\Install\Mount" /Optimize-ProvisionedAppxPackages
```

### 6. 累积更新

- 不同版本、旧版本升级到最新版时，需优先添加“功能启用包”后才能添加最新版累积更新；
- 添加语言包后，可安装与初始版本相同的累积更新，以解决安装后未刷新“组件：映像中已安装的所有包”状态的已知问题；



- 为保持最新，推荐您下载最新版。

## 6.1. 功能启用包

- 学习：[KB5015684：使用启用包对 Windows 10 版本 22H2 的特别推荐更新](#)
- 注意：[使用 Windows 10 22H2 19045.2006 相同版本和以上版本号时，可跳过功能启用包。](#)
- 制作 Windows 10 版本 2004、20H2、21H1、21H2 时，需提前安装“功能启用包”后才能安装 19045.\* 的累积更新，下载后保存到：[D:\OS\\_10\\_Custom\Install\Install\Update](#)，根据架构选择下载：

### 6.1.1. x64, 默认

- 直连下载

[https://catalog.s.download.windowsupdate.com/c/upgr/2022/07/windows10.0-kb5015684-x64\\_523c039b86ca98f2d818c4e6706e2cc94b634c4a.msu](https://catalog.s.download.windowsupdate.com/c/upgr/2022/07/windows10.0-kb5015684-x64_523c039b86ca98f2d818c4e6706e2cc94b634c4a.msu)

- 安装

```
$KBPath = "D:\OS_10_Custom\Install\Install\Update\windows10.0-kb5015684-x64_523c039b86ca98f2d818c4e6706e2cc94b634c4a.msu"

Add-WindowsPackage -Path "D:\OS_10_Custom\Install\Install\Mount" -PackagePath $KBPath
```

## 6.2. 初始版本

累积更新 KB5017308 已无法从 <https://www.catalog.update.microsoft.com/Search.aspx?q=KB5017308> 里搜索到，通过直连方式下载后保存到：[D:\OS\\_10\\_Custom\Install\Install\Update](#)，根据架构选择下载：

### 6.2.1. x64, 默认

- 直连下载

[https://catalog.s.download.windowsupdate.com/c/msdownload/update/software/secu/2022/09/windows10.0-kb5017308-x64\\_2027053968a06948b45d139d95475ab4feee5654.msu](https://catalog.s.download.windowsupdate.com/c/msdownload/update/software/secu/2022/09/windows10.0-kb5017308-x64_2027053968a06948b45d139d95475ab4feee5654.msu)

- 安装

```
$KBPath = "D:\OS_10_Custom\Install\Install\Update\windows10.0-kb5017308-x64_2027053968a06948b45d139d95475ab4feee5654.msu"

Add-WindowsPackage -Path "D:\OS_10_Custom\Install\Install\Mount" -PackagePath $KBPath
```

## 6.3. 其它版本

查阅“[Windows 10 版本信息](#)”，例如下载累积更新：[KB5032278](#)，前往下载页面：

<https://www.catalog.update.microsoft.com/Search.aspx?q=KB5032278>，下载后保存到：D:\OS\_10\_Custom\Install\Install\Update，  
或通过直连下载，根据架构选择下载：

6.3.1. x64，默认

- 直连下载

[https://catalog.s.download.windowsupdate.com/d/msdownload/update/software/updt/2023/11/windows10-kb5032278-x64\\_3a2b384f690eaa92c5030c535ac4a7e47c71a635.msu](https://catalog.s.download.windowsupdate.com/d/msdownload/update/software/updt/2023/11/windows10-kb5032278-x64_3a2b384f690eaa92c5030c535ac4a7e47c71a635.msu)

- 添加

```
$KBPath = "D:\OS_10_Custom\Install\Install\Update\windows10.0-kb5032278-x64_3a2b384f690eaa92c5030c535ac4a7e47c71a635.msu"
```

```
Add-WindowsPackage -Path "D:\OS_10_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.4. 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_10_Custom\Install\Install\Mount" /cleanup-image /StartComponentCleanup /ResetBase
```

6.4.1. 固化更新后清理组件，可选项

```
$Mount = "D:\OS_10_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

    }

}
```

7. 部署引擎：添加，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。

8. 健康

保存前应检查是否损坏，健康状态异常时，中止保存

```
Repair-WindowsImage -Path "D:\OS_10_Custom\Install\Install\Mount" -ScanHealth
```

9. 替换 WinRE.wim

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

```
$WinRE = "D:\OS_10_Custom\Install\Install\Update\Winlib\WinRE.wim"

$CopyTo = "D:\OS_10_Custom\Install\Install\Mount\Windows\System32\Recovery"

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

10. 保存映像 Install.wim

```
Save-WindowsImage -Path "D:\OS_10_Custom\Install\Install\Mount"
```

11. 卸载映像 Install.wim

关闭所有可能正在访问映像中文件的应用程序，包括文件资源管理器。

```
Dismount-WindowsImage -Path "D:\OS_10_Custom\Install\Install\Mount" -Discard
```

循环操作区域，结束。

12. 重建 Install.wim 后可缩小文件大小

- Install.Rebuild.wim.ps1
  - [\Expand\Install\Install.Rebuild.wim.ps1](#)
  - [https://github.com/ilikeyi/solutions/blob/main/\\_Learn/Packaging/tutorial/OS.10/22H2/Expand/Install/Install.Rebuild.wim.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging/tutorial/OS.10/22H2/Expand/Install/Install.Rebuild.wim.ps1)

- 复制代码

```
$InstallWim = "D:\OS_10\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

}

```

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

#### 13.1. 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](#)

#### 13.2. 如何在 Install.wim 里提取和更新 WinRE.wim

##### 13.2.1. 从 Install.wim 里提取 WinRE.wim 文件 Install.wim

- Install.WinRE.Replace.wim.ps1
  - [\Expand\Install\Install.WinRE.Replace.wim.ps1](#)
  - [https://github.com/ilikeyi/solutions/blob/main/\\_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/Install.WinRE.Replace.wim.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/Install.WinRE.Replace.wim.ps1)

- 复制代码

```

$Arguments = @(

    "extract",

    "D:\OS_10\sources\install.wim", "1",

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

    "--dest-dir=""D:\OS_10_Custom\Install\Install\Update\Winlib""

)

New-Item -Path "D:\OS_10_Custom\Install\Install\Update\Winlib" -ItemType Directory -ea SilentlyContinue

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

```

##### 13.2.2. 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.10/22H2/Expand/Install/Install.WinRE.Replace.wim.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/Install/Install.WinRE.Replace.wim.ps1)

- 复制代码

```
Get-WindowsImage -ImagePath "D:\OS_10\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_10\sources\install.wim", $_.ImageIndex,  
  
        "--command="""add 'D:\OS_10_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_10\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_10_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_10\sources\boot.wim" -Index "2" -Path "D:\OS_10_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/\\_Learn/Packaging.tutorial/OS.10/22H2/Expand/Boot/Boot.Instl.lang.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/Boot/Boot.Instl.lang.ps1)

- 复制代码

```
$Mount = "D:\OS_10_Custom\Boot\Boot\Mount"

$Sources = "D:\OS_10_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*Client*Package*"; File = "WINPE-SETUP-CLIENT_zh-CN.CAB"; }

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

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

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

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

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

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

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

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

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

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

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

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

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

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

)

ForEach ($Rule in $Language) {

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

    ForEach ($Component in $Initl_install_Language_Component) {

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

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

```
Write-host "  Language pack file: " -NoNewline; Write-host "$($Sources)\$($Rule.File)" -ForegroundColor Green

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

try{

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

    Write-host "Finish" -ForegroundColor Green

} catch {

    Write-host "Failed" -ForegroundColor Red

    Write-host "  $($_) " -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_10_Custom\Boot\Boot\Mount" | Out-GridView
```

##### 4.2.2. 导出到 Csv

```
$SaveTo = "D:\OS_10_Custom\Boot\Boot\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS_10_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_10_Custom\Boot\Boot\Mount\sources\zh-CN" -Destination "D:\OS_10\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:\en-  
us_windows_server_2022_x64_dvd_620d7eac_Custom\Boot\Boot\Mount\Sources\lang.ini  
  
Dism /image:"D:\OS_10_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS_10_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:\en-us\\_windows\\_server\\_2022\\_x64\\_dvd\\_620d7eac\Sources\lang.ini](#)

```
Dism /image:"D:\OS_10_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS_10"
```

5. 保存映像 Boot.wim

```
Save-WindowsImage -Path "D:\OS_10_Custom\Boot\Boot\Mount"
```

6. 卸载映像 Boot.wim

关闭所有可能正在访问映像中文件的应用程序，包括文件资源管理器。

```
Dismount-WindowsImage -Path "D:\OS_10_Custom\Boot\Boot\Mount" -Discard
```

IV 部署引擎

- 了解“自动添加 Windows 系统已安装的语言”，学习：<https://github.com/ilikeyi/Multilingual>，如何下载：
  - 进入网站后，点击“代码”，“下载压缩包”，下载完成后得到 main.zip 压缩包文件。
  - 前往 <https://github.com/ilikeyi/Multilingual/releases> 下载页面，选择可用版本：1.1.1.1，选择下载源代码格式：zip，下载完成后得到 Multilingual-1.1.1.1.zip 压缩包文件；
- 将已下载的 main.zip 或 Multilingual-1.1.1.1.zip，解压到：D:\Multilingual-1.1.1.1，重命名：D:\Multilingual
- 学习“[无人值守 Windows 安装参考](#)”，通过无人值守来干预安装过程。

1. 添加方式

1.1. 添加到 ISO 安装介质

1.1.1. 无人值守

1.1.1.1. 添加到：[\[ISO\]:\Autounattend.xml](#)

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

复制 [D:\Multilingual\Learn\Unattend\Mul.Unattend.xml](#) 到 [D:\OS\\_10\Autounattend.xml](#)

```
Copy-Item "D:\Multilingual\Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS_10\Autounattend.xml" -Force
```

1.1.1.2. 添加到：[\[ISO\]:\Sources\Unattend.xml](#)

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

复制 [D:\Multilingual\Learn\Unattend\Mul.Unattend.xml](#) 到 [D:\OS\\_10\Sources\Unattend.xml](#)



```
Copy-Item "D:\Multilingual\Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS_10\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_10\sources\`$OEM$\`$$\Panther" -ItemType Directory
```

1.1.1.3.2. 复制

复制 D:\Multilingual\Learn\Unattend\Mul.Unattend.xml 到  
D:\OS\_10\Sources\\$OEM\$\Panther\Unattend.xml

```
Copy-Item "D:\Multilingual\Learn\Unattend\Mul.Unattend.xml" -Destination  
"D:\OS_10\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\_10\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\_10\Sources\\$OEM\$\\$1\Yi\Engine

```
Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS_10\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_10\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -ErrorAction
    SilentlyContinue | Out-Null

    Out-File -FilePath "D:\OS_10\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\\_10\\_Custom\Install\Install\Mount](#)

1.2.1. 无人值守

复制 [D:\Multilingual\Learn\Unattend\Mul.Unattend.xml](#) 到 [D:\OS\\_10\\_Custom\Install\Install\Mount\Panther\Unattend.xml](#)

```
Copy-Item "D:\Multilingual\Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS_10_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\\_10\\_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\\_10\\_Custom\Install\Install\Mount\Yi\Engine](#)

```
Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS_10_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_10_Custom\Install\Install\Mount\Yi\Engine\Deploy\Allow" -ItemType Directory -
    ErrorAction SilentlyContinue | Out-Null

    Out-File -FilePath "D:\OS_10_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\_10\Autounattend.xml
- D:\OS\_10\Sources\Unattend.xml
- D:\OS\_10\sources\\$\OEM\$\\$\$\Panther\unattend.xml
- D:\OS\_10\_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 OScding

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

1.1. x64

[https://github.com/ilikeyi/solutions/raw/main/\\_Software/Oscding/amd64/oscdimg.exe](https://github.com/ilikeyi/solutions/raw/main/_Software/Oscding/amd64/oscdimg.exe)

1.2. x86

[https://github.com/ilikeyi/solutions/raw/main/\\_Software/Oscding/x86/oscdimg.exe](https://github.com/ilikeyi/solutions/raw/main/_Software/Oscding/x86/oscdimg.exe)

1.3. arm64

[https://github.com/ilikeyi/solutions/raw/main/\\_Software/Oscding/arm64/oscdimg.exe](https://github.com/ilikeyi/solutions/raw/main/_Software/Oscding/arm64/oscdimg.exe)

2. 使用 oscding 命令行生成一个 ISO 文件，保存到：D:\Win11.iso

- ISO.ps1
  - [\Expand\ISO.ps1](#)
  - [https://github.com/ilikeyi/solutions/blob/main/\\_Learn/Packaging.tutorial/OS.10/22H2/Expand/ISO.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.10/22H2/Expand/ISO.ps1)

• 复制代码

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

$ISO = "D:\OS_10"

$Volume = "OS_10"

$SaveTo = "D:\OS_10.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 $Oscding -ArgumentList $Arguments -wait -nonewwindow
```



作者: Yi

网站: <https://fengyi.tel>

邮箱: [775159955@qq.com](mailto:775159955@qq.com), [ilikeyi@outlook.com](mailto:ilikeyi@outlook.com)

文档版本: 1.0

文档模型: 精简版

更新日期: 2024 - 4

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