Yi’ Solutions

Microsoft Windows Server 2022

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

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

## Summary

1. Encapsulation
2. Common problem
3. Known issues

Table of contents

1. Encapsulation Page 5
   * 1. Prerequisites Page 5
2. Running system Page 5
3. When using the DISM command to create a higher version image Page 5
4. Disk partition Page 5
5. Windows Security Page 5
6. Command line Page 6
7. Requirements Page 6
8. System installation package Page 6
9. Language Pack Page 6
   1. Learn Page 6
   2. Language pack: Download Page 7
      1. Language package: extract Page 7
10. Language pack: Ready Page 7
11. Language pack: Extract scheme Page 7
12. Execute the extract command Page 7
    * 1. Custom encapsulation Page 13
13. Custom encapsulation: Install.wim Page 13
14. View Install.wim details Page 13
15. Specify the path to mount install.wim Page 13
16. Start mounting Install.wim Page 13
    1. Custom encapsulation: WinRE.wim Page 13
       1. View WinRE.wim details Page 13
       2. Specify the path to mount WinRE.wim Page 14
       3. Start mounting WinRE.wim Page 14
       4. Language pack Page 14
          1. Language pack: add Page 14
          2. Offline image language: change Page 16
             1. Change default language, regional settings, and other international settings Page 16
             2. View available language settings Page 16
          3. Components: All packages installed in the image Page 16
       5. Cumulative updates Page 16
          1. Add Page 16
          2. Delete Page 16
          3. Solid update Page 16
             1. Clean components after curing and updating Page 16
       6. Drive Page 17
       7. Save image: WinRE.wim Page 17
       8. Unmount image: WinRE.wim Page 17
       9. After rebuilding WinRE.wim, the file size can be reduced Page 17
       10. Backup WinRE.wim Page 18
       11. Replace WinRE.wim within the Install.wim image Page 18
17. Language pack Page 18
    1. Language pack: add Page 18
    2. Offline image language: change Page 24
       1. Change default language, regional settings, and other international settings Page 24
       2. View available language settings Page 24
    3. Components: All packages installed in the image Page 24
18. Cumulative updates Page 25
    1. Download Page 25
    2. Add Page 25
    3. Solid update Page 25
       1. Clean up components after curing updates Page 25
19. Drive Page 25
20. Deployment engine: Add Page 26
21. Health Page 26
22. Replace the WinRE.wim Page 26
23. Save image: Install.wim Page 26
24. Unmount image: Install.wim Page 26
25. How to batch replace WinRE.wim in all index numbers in Install.wim Page 26
    1. Get WimLib Page 26
    2. How to extract and update WinRE.wim in Install.wim Page 26
26. Rebuilding Install.wim reduces file size Page 27
27. Split, merge, compress, and convert Page 28
    1. Splitting and merging Page 28
       1. Splitting Page 28
       2. Merge Page 29
    2. Solid compressed ESD format and WIM format conversion Page 30
       1. Solid compression Page 30
       2. Convert compressed files to WIM file format Page 30
28. Custom encapsulation: boot.wim Page 31
29. View Boot.wim details Page 31
30. Specify the path to mount Boot.wim Page 31
31. Start mounting Boot.wim Page 31
32. Language pack Page 31
    1. Language pack: add Page 32
    2. Offline image language: change Page 33
       1. Change default language, regional settings, and other international settings Page 33
       2. View available language settings Page 33
    3. Components: All packages installed in the image Page 33
    4. Language packs: sync to ISO installer Page 34
    5. Regenerate Lang.ini Page 34
       1. Regenerate the mounted directory lang.ini Page 34
       2. After regenerating lang.ini, synchronize to the installer Page 34
33. Cumulative updates Page 34
    1. Add Page 34
    2. Delete Page 34
    3. Solid update Page 34
       1. Clean components after curing and updating Page 34
34. Drive Page 35
35. Save image: Boot.wim Page 35
36. Unmount image: Boot.wim Page 35
37. Deployment engine Page 35
38. Add method Page 35
39. Deployment Engine: Advanced Page 38
    * 1. Generate ISO Page 40
40. Common problem Page 41
41. Clean all mounts to Page 41
42. Fix the problem of abnormal mounting Page 41
43. Clean up Page 41
44. Known issues Page 42

## Encapsulation

### Prerequisites

#### Running system

##### When using the DISM command to create a higher version image

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

* 1. Upgrade the running operating system or reinstall to a higher version (recommended);
  2. Upgrade or install a new version of ADK or PowerShell (not recommended)
     1. You can try to upgrade to the latest PowerShell 7 or higher version;
     2. After installing the latest version of ADK and replacing the DISM command, the problem of low DISM version can be solved. However, the command line mainly used by the packaging script is the PowerShell command line, so it is not recommended that you use the above method. The best method is to upgrade the running operating system or reinstall to a higher version.

##### Disk partition

* 1. After mounting an offline image to a REFS disk partition, some DISM commands may fail to execute properly. NTFS disk partitions are not affected by this.
  2. After the ISO is decompressed, its location is not affected by the REFS partition.

##### Windows Security

* When processing the encapsulation task, a large number of temporary files will be generated, and a large number of installation files will be released when installing the application in InBox Apps;
* Turning on Windows Security scans files and takes up a lot of CPU.
* In test: 1 hour and 22 minutes before shutdown, 20 minutes after shutdown.

How to close:

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

* 1. Open Windows Security or run: windowsdefender:
  2. Select "Virus & Threat Protection" or Run: windowsdefender://threat
  3. Find "Virus & Threat Protection Settings", click "Manage Settings" or Run: windowsdefender://threatsettings, we recommend that you turn off some features:
     1. Real-time protection
     2. Cloud=delivered protection
     3. Automatic sample submission
     4. Tamper Protection
  4. When you're not encapsulated, we recommend that you turn on Windows Security.

##### Command line

* 1. Optional "Terminal" or "PowerShell ISE", if "Terminal" is not installed, please go to: <https://github.com/microsoft/terminal/releases> After downloading;
  2. Open "Terminal" or "PowerShell ISE" as administrator, it is recommended to set the PowerShell execution policy: bypass, PS command line:

Set-ExecutionPolicy -ExecutionPolicy Bypass -Force

* 1. In this article, PS command line, green part, please copy it, paste it into the "Terminal" dialog box, press Enter and start running;
  2. When there is .ps1, right-click the file and select Run with PowerShell, or copy the path and paste it into Terminal to run, the path with a colon, add the & character in the command line, example: & "D:\YiSolutions\\_Encapsulation\\_SIP.ps1"

#### Requirements

##### System installation package

* 1. Prepare to download the initial release or developer version
     1. x64
        1. Filename: [en-us\_windows\_server\_2022\_x64\_dvd\_620d7eac.iso](https://files.rg-adguard.net/file/9a0f4eb7-c3a9-e46b-3fc8-cdb71289dbfb)

List of files: <https://files.rg-adguard.net/file/9a0f4eb7-c3a9-e46b-3fc8-cdb71289dbfb>

* 1. For example, after downloading en-us\_windows\_server\_2022\_x64\_dvd\_620d7eac.iso, extract it to: D:\en-us\_windows\_server\_2022\_x64\_dvd\_620d7eac

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

* 1. After decompression, change the directory D:\en-us\_windows\_server\_2022\_x64\_dvd\_620d7eac to D:\OS\_2022
  2. All scripts and all paths have been set to D:\OS\_2022 by default as the image source.

##### Language Pack

###### Learn

* + 1. [Add languages to a Windows 11 image](https://learn.microsoft.com/en-us/azure/virtual-desktop/windows-11-language-packs)
    2. [Language and region Features on Demand (FOD)](https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/features-on-demand-language-fod?view=windows-11)
       1. Fonts

When adding a language pack, when the corresponding region is triggered, the required font functions need to be added, download "[List of all available language FODs](https://download.microsoft.com/download/7/6/0/7600F9DC-C296-4CF8-B92A-2D85BAFBD5D2/Windows-10-1809-FOD-to-LP-Mapping-Table.xlsx)" learn more.

In "[Language package: extract](#_Language_package:_extract)", the automatic recognition function has been added, and you can understand the functions: Function Match\_Required\_Fonts

* + - 1. Regional association

What are regional connections?

* When the image language is only in English, after adding the zh-HK language pack, the image language will not be added. You should install zh-TW first, and then install zh-HK to obtain the corresponding association.
* Please refer to Microsoft's official original version: Windows 10, Windows 11 Traditional Chinese version.

Known regional associations:

* + - * 1. Region: zh-TW, Optional associated areas: zh-HK

###### Language pack: Download

* + 1. Filename: <https://software-download.microsoft.com/download/sg/20348.1.210507-1500.fe_release_amd64fre_SERVER_LOF_PACKAGES_OEM.iso>

List of files: <https://files.rg-adguard.net/file/f4a036a7-5c8e-6bd6-764a-83655c1a9ce5>

### Language package: extract

#### Language pack: Ready

Mounted 20348.1.210507-1500.fe\_release\_amd64fre\_SERVER\_LOF\_PACKAGES\_OEM.iso or unzip it to any location;

#### Language pack: Extract scheme

##### Add

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

##### Delete

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

#### Execute the extract command

* Auto = automatically search all local disks, default;
* Customize the path, for example, specify the E drive: $ISO = "E:\"
* Extract.ps1
  + [\Expand\Extract.ps1](Expand/Extract.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Extract.ps1>
* Copy the code

$ISO = "Auto"

$SaveTo = "D:\OS\_2022\_Custom"

$Extract\_language\_Pack = @(

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

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

)

Function Extract\_Language

{

param( $Act, $NewLang, $Expand )

Function Match\_Required\_Fonts

{

param( $Lang )

$Fonts = @(

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

)

ForEach ($item in $Fonts) {

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

return $item.Name

}

}

return "Not\_matched"

}

Function Match\_Other\_Region\_Specific\_Requirements

{

param( $Lang )

$RegionSpecific = @(

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

)

ForEach ($item in $RegionSpecific) {

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

return $item.Name

}

}

return "Skip\_specific\_packages"

}

Function Extract\_Process

{

param( $Package, $Name, $NewSaveTo )

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

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

if ($ISO -eq "Auto") {

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

ForEach ($item in $Package) {

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

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

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

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

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

}

}

}

} else {

ForEach ($item in $Package) {

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

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

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

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

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

} else {

Write-host " Not found"

}

}

}

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

ForEach ($item in $Package) {

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

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

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

} else {

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

}

}

}

$AdvLanguage = @(

@{

Path = "Install\Install"

Rule = @(

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

"LanguagesAndOptionalFeatures\Microsoft-Windows-Server-Language-Pack\_x64\_{Lang}.cab"

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

)

}

@{

Path = "Install\WinRE"

Rule = @(

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

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

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-securestartup\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-atbroker\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-audiocore\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-audiodrivers\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-enhancedstorage\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-narrator\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-scripting\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-speech-tts\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-srh\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-srt\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-wds-tools\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-wmi\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-appxpackaging\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-storagewmi\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-wifi\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-rejuv\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-opcservices\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-hta\_{Lang}.cab"

)

}

@{

Path = "Boot\Boot"

Rule = @(

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

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

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\WinPE-Setup\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\WINPE-SETUP-Server\_{Lang}.CAB"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-securestartup\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-atbroker\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-audiocore\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-audiodrivers\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-enhancedstorage\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-narrator\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-scripting\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-speech-tts\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-srh\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-srt\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-wds-tools\_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE\_OCs\{Lang}\winpe-wmi\_{Lang}.cab"

)

}

)

$NewFonts = Match\_Required\_Fonts -Lang $NewLang

$SpecificPackage = Match\_Other\_Region\_Specific\_Requirements -Lang $NewLang

Foreach ($item in $Expand) {

$Language = @()

Foreach ($itemList in $AdvLanguage) {

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

Foreach ($PrintLang in $itemList.Rule) {

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

}

Extract\_Process -NewSaveTo $itemList.Path -Package $Language -Name $item

}

}

}

}

ForEach ($item in $Extract\_language\_Pack) { Extract\_Language -Act $item.Act -NewLang $item.Tag -Expand $item.Scope }

### Custom encapsulation

#### Custom encapsulation: Install.wim

##### View Install.wim details

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

$ViewFile = "D:\OS\_2022\Sources\Install.wim"

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

CYCLIC OPERATION AREA, START,

##### Specify the path to mount install.wim

New-Item -Path "D:\OS\_2022\_Custom\Install\Install\Mount" -ItemType directory

##### Start mounting Install.wim

Default index number: 1

Mount-WindowsImage -ImagePath "D:\OS\_2022\sources\install.wim" -Index "1" -Path "D:\OS\_2022\_Custom\Install\Install\Mount"

PROCESS FILES INSIDE THE INSTALL.WIM IMAGE, OPTIONALLY, START

###### Custom encapsulation: WinRE.wim

WARNING:

* WinRE.wim is a file within the Install.wim image;
* When Install.wim has multiple index numbers, only process any WinRE.wim;
* Synchronizing to all index numbers reduces the Install.wim volume, Learn "[How to extract and update WinRE.wim in Install.wim](#_How_to_batch)".
  + 1. View WinRE.wim details

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

$ViewFile = "D:\OS\_2022\_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

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

* + 1. Specify the path to mount WinRE.wim

New-Item -Path "D:\OS\_2022\_Custom\Install\WinRE\Mount" -ItemType directory

* + 1. Start mounting WinRE.wim

Default index number: 1

$FileName = "D:\OS\_2022\_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

Mount-WindowsImage -ImagePath $FileName -Index "1" -Path "D:\OS\_2022\_Custom\Install\WinRE\Mount"

* + 1. Language pack
* Automatically install language packs: Get "Component: All installed packages in the image" and match them. After matching the corresponding names, install the local corresponding language pack files.
* When adding languages, different schema versions must be corresponded, and if not, errors are reported during the addition process.
  + - 1. Language pack: add

WinRE.Instl.lang.ps1

[\Expand\Install\WinRE\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>

Copy the code

$Mount = "D:\OS\_2022\_Custom\Install\WinRE\Mount"

$Sources = "D:\OS\_2022\_Custom\Install\WinRE\Language\Add\zh-CN"

$Initl\_install\_Language\_Component = @()

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

$Initl\_install\_Language\_Component += $\_.PackageName

}

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

$Language\_List = @(

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

@{ Match = "\*SecureStartup\*"; File = "winpe-securestartup\_zh-CN.cab"; }

@{ Match = "\*ATBroker\*"; File = "winpe-atbroker\_zh-CN.cab"; }

@{ Match = "\*AudioCore\*"; File = "winpe-audiocore\_zh-CN.cab"; }

@{ Match = "\*AudioDrivers\*"; File = "winpe-audiodrivers\_zh-CN.cab"; }

@{ Match = "\*EnhancedStorage\*"; File = "winpe-enhancedstorage\_zh-CN.cab"; }

@{ Match = "\*Narrator\*"; File = "winpe-narrator\_zh-CN.cab"; }

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

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

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

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

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

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

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

@{ Match = "\*StorageWMI\*"; File = "winpe-storagewmi\_zh-CN.cab"; }

@{ Match = "\*WiFi\*"; File = "winpe-wifi\_zh-CN.cab"; }

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

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

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

)

ForEach ($Rule in $Language\_List) {

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

ForEach ($Component in $Initl\_install\_Language\_Component) {

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

Write-host " Component name: " -NoNewline

Write-host $Component -ForegroundColor Green

Write-host " Language pack file: " -NoNewline

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

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

try {

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

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

}

break

}

}

}

* + - 1. Offline image language: change
         1. Change default language, regional settings, and other international settings

Language Tag: zh-CN

Dism /Image:"D:\OS\_2022\_Custom\Install\WinRE\Mount" /Set-AllIntl:zh-CN

* + - * 1. View available language settings

Dism /Image:"D:\OS\_2022\_Custom\Install\WinRE\Mount" /Get-Intl

* + - 1. Components: All packages installed in the image
         1. View

Get-WindowsPackage -Path "D:\OS\_2022\_Custom\Install\WinRE\Mount" | Out-GridView

* + - * 1. Export to Csv

$SaveTo = "D:\OS\_2022\_Custom\Install\WinRE\Report.Components.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS\_2022\_Custom\Install\WinRE\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green

* + 1. Cumulative updates

To prepare the cumulative updates file available, change the example file name: KB\_WinRE.cab

* + - 1. Add

$KBPath = "D:\OS\_2022\_Custom\Install\WinRE\Update\KB\_WinRE.cab"

Add-WindowsPackage -Path "D:\OS\_2022\_Custom\Install\WinRE\Mount" -PackagePath $KBPath

* + - 1. Delete

$KBPath = "D:\OS\_2022\_Custom\Install\WinRE\Update\KB\_WinRE.cab"

Remove-WindowsPackage -Path "D:\OS\_2022\_Custom\Install\WinRE\Mount" -PackagePath $KBPath

* + - 1. Solid update

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

Dism /image:"D:\OS\_2022\_Custom\Install\WinRE\Mount" /cleanup-image /StartComponentCleanup /ResetBase

* + - * 1. Clean components after curing and updating

$Mount = "D:\OS\_2022\_Custom\Install\WinRE\Mount"

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

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

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

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

}

}

* + 1. Drive
    2. Save image: WinRE.wim

Save-WindowsImage -Path "D:\OS\_2022\_Custom\Install\WinRE\Mount"

* + 1. Unmount image: WinRE.wim

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

Dismount-WindowsImage -Path "D:\OS\_2022\_Custom\Install\WinRE\Mount" -Discard

* + 1. After rebuilding WinRE.wim, the file size can be reduced
* WinRE.Rebuild.ps1
  + [\Expand\Install\WinRE\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>
* Copy the code

$FileName = "D:\OS\_2022\_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

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

Write-Host " Image name: " -NoNewline; Write-Host "$($\_.ImageName)" -ForegroundColor Yellow

Write-Host " The index number: " -NoNewline; Write-Host "$($\_.ImageIndex)" -ForegroundColor Yellow

Write-Host "`n 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

}

* + 1. Backup WinRE.wim
* WinRE.Backup.ps1
  + [\Expand\Install\WinRE\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>
* Copy the code

$WimLibPath = "D:\OS\_2022\_Custom\Install\Install\Update\Winlib"

$FileName = "D:\OS\_2022\_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

New-Item -Path $WimLibPath -ItemType Directory

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

* + 1. Replace WinRE.wim within the Install.wim image
* After each installation of Install.wim, use item "[Replace the WinRE.wim](#_Replace_the_WinRE.wim)";
* Learning "[Get all index numbers of Install.wim and replace the old WinRE.wim](#_Get_all_index)".

PROCESS FILES INSIDE THE INSTALL.WIM IMAGE, END

##### Language pack

* Automatically install language packs: Get "Component: All installed packages in the image" and match them. After matching the corresponding names, install the local corresponding language pack files.
* When adding languages, different schema versions must be corresponded, and if not, errors are reported during the addition process.

###### Language pack: add

* Install.Instl.lang.ps1
  + [\Expand\Install\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>
* Copy the code

Function Language\_Install

{

param($Mount, $Sources, $Lang)

$Initl\_install\_Language\_Component = @()

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

Get-WindowsPackage -Path $Mount | ForEach-Object { $Initl\_install\_Language\_Component += $\_.PackageName }

} else {

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

return

}

$Script:Init\_Folder\_All\_File = @()

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

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

$Script:Init\_Folder\_All\_File += $\_.FullName

}

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

if ($Script:Init\_Folder\_All\_File.Count -gt 0 ) {

ForEach ($item in $Script:Init\_Folder\_All\_File) {

Write-host " $($item)"

}

} else {

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

return

}

} else {

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

return

}

$Script:Init\_Folder\_All\_File\_Match\_Done = @()

$Script:Init\_Folder\_All\_File\_Exclude = @()

$Global:Search\_File\_Order = @(

@{

Name = "Fonts"

Description = "Fonts"

Rule = @(

@{ Match\_Name = "\*Fonts\*"; IsMatch = "No"; Capability = ""; }

)

}

@{

Name = "Basic"

Description = "Basic"

Rule = @(

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

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

)

}

@{

Name = "OCR"

Description = "Optical character recognition"

Rule = @(

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

)

}

@{

Name = "Handwriting"

Description = "Handwriting recognition"

Rule = @(

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

)

}

@{

Name = "TextToSpeech"

Description = "Text-to-speech"

Rule = @(

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

)

}

@{

Name = "Speech"

Description = "Speech recognition"

Rule = @(

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

)

}

@{

Name = "RegionSpecific"

Description = "Other region-specific requirements"

Rule = @(

@{ Match\_Name = "\*InternationalFeatures\*zh-TW\*"; IsMatch = "Yes"; Capability = ""; }

)

}

@{

Name = "Retail"

Description = "Retail demo experience"

Rule = @(

@{ Match\_Name = "\*RetailDemo\*"; IsMatch = "Yes"; Capability = ""; }

)

}

@{

Name = "Features\_On\_Demand"

Description = "Features on demand"

Rule = @(

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

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

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

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

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

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

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

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

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

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

)

}

)

ForEach ($item in $Global:Search\_File\_Order) {

New-Variable -Scope global -Name "Init\_File\_Type\_$($item.Name)" -Value @() -Force

}

ForEach ($WildCard in $Script:Init\_Folder\_All\_File) {

ForEach ($item in $Global:Search\_File\_Order) {

ForEach ($TTT in $item.Rule) {

if ($WildCard -like "\*$($TTT.Match\_Name)\*") {

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

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

$OSDefaultUser = (Get-Variable -Scope global -Name "Init\_File\_Type\_$($item.Name)" -ErrorAction SilentlyContinue).Value

$TempSave = @{ Match\_Name = $TTT.Match\_Name; Capability = $TTT.Capability; FileName = $WildCard }

$new = $OSDefaultUser + $TempSave

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

ForEach ($Component in $Initl\_install\_Language\_Component) {

if ($Component -like "\*$($TTT.Match\_Name)\*") {

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

New-Variable -Scope global -Name "Init\_File\_Type\_$($item.Name)" -Value $new -Force

$Script:Init\_Folder\_All\_File\_Match\_Done += $WildCard

break

}

}

} else {

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

New-Variable -Scope global -Name "Init\_File\_Type\_$($item.Name)" -Value $new -Force

$Script:Init\_Folder\_All\_File\_Match\_Done += $WildCard

}

}

}

}

}

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

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

ForEach ($WildCard in $Global:Search\_File\_Order) {

$OSDefaultUser = (Get-Variable -Scope global -Name "Init\_File\_Type\_$($WildCard.Name)" -ErrorAction SilentlyContinue).Value

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

if ($OSDefaultUser.Count -gt 0) {

ForEach ($item in $OSDefaultUser) {

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

}

} else {

Write-host " Not available" -ForegroundColor Red

}

}

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

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

ForEach ($item in $Script:Init\_Folder\_All\_File) {

if ($Script:Init\_Folder\_All\_File\_Match\_Done -notcontains $item) {

$Script:Init\_Folder\_All\_File\_Exclude += $item

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

}

}

Write-host "`n Install" -ForegroundColor Yellow

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

ForEach ($WildCard in $Global:Search\_File\_Order) {

$OSDefaultUser = (Get-Variable -Scope global -Name "Init\_File\_Type\_$($WildCard.Name)" -ErrorAction SilentlyContinue).Value

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

if ($OSDefaultUser.Count -gt 0) {

ForEach ($item in $OSDefaultUser) {

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

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

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

try {

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

Write-host "Finish`n" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

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

}

} else {

Write-host "Does not exist`n"

}

}

} else {

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

}

}

}

Language\_Install -Mount "D:\OS\_2022\_Custom\Install\Install\Mount" -Sources "D:\OS\_2022\_Custom\Install\Install\Language\Add" -Lang "zh-CN"

###### Offline image language: change

* Starting Windows 11, the [default System UI Language](https://learn.microsoft.com/en-us/windows/win32/intl/user-interface-language-management) set by DISM is left unaltered on all editions except for Home edition. For all [commercial editions](https://www.microsoft.com/windowsforbusiness/compare) the language chosen during the Out-of-Box Experience (OOBE) is set as the [System Preferred UI language](https://learn.microsoft.com/en-us/windows/win32/intl/user-interface-language-management) and Windows will be displayed in this language and for Home edition the language chosen at OOBE will continue to be the default System UI Language.
* As of Windows 10, version 2004, if an .appx-based Language Experience Pack (LXP) backed language is passed as an argument then the language will be set as the System Preferred UI language and its parent language will be set as the Default System UI language. In prior versions only .cab based language packs were supported.
  + 1. Change default language, regional settings, and other international settings

Language Tag: zh-CN

Dism /Image:"D:\OS\_2022\_Custom\Install\Install\Mount" /Set-AllIntl:zh-CN

* + 1. View available language settings

Dism /Image:"D:\OS\_2022\_Custom\Install\Install\Mount" /Get-Intl

###### Components: All packages installed in the image

* + 1. View

Get-WindowsPackage -Path "D:\OS\_2022\_Custom\Install\Install\Mount" | Out-GridView

* + 1. Export to Csv

$SaveTo = "D:\OS\_2022\_Custom\Install\Install\Report.Components.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS\_2022\_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green

##### Cumulative updates

###### Download

Check the "[Windows Server 2022 Update History](https://support.microsoft.com/en-gb/topic/windows-server-2022-update-history-e1caa597-00c5-4ab9-9f3e-8212fe80b2ee) ", for example, install the cumulative update: KB5030216

Go to the download page: <https://www.catalog.update.microsoft.com/Search.aspx?q=Kb5030216> Or "[Direct download](https://catalog.s.download.windowsupdate.com/c/msdownload/update/software/secu/2023/09/windows10.0-kb5030216-x64_cbe587155f9818548b75f65d5cd41d341ed2fc61.msu)" (If you cannot download, please go to the download page), save to

D:\OS\_2022\_Custom\Install\Install\Update\windows10.0-kb5030216-x64\_cbe587155f9818548b75f65d5cd41d341ed2fc61.msu

###### Add

$KBPath = "D:\OS\_2022\_Custom\Install\Install\Update\windows10.0-kb5030216-x64\_cbe587155f9818548b75f65d5cd41d341ed2fc61.msu"

Add-WindowsPackage -Path "D:\OS\_2022\_Custom\Install\Install\Mount" -PackagePath $KBPath

###### Solid update

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

Dism /Image:"D:\OS\_2022\_Custom\Install\Install\Mount" /cleanup-image /StartComponentCleanup /ResetBase

* + 1. Clean up components after curing updates
* Install.Update.Curing.ps1
  + [\Expand\Install\Install.Update.Curing.ps1](Expand/Install/Install.Update.Curing.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/Install/Install.Update.Curing.ps1>
* Copy the code

$Mount = "D:\OS\_2022\_Custom\Install\Install\Mount"

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

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

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

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

}

}

##### Drive

##### Deployment engine: Add

* Learn "[Deployment engine](#_Deployment_engine)", if added to ISO installation media, can skip adding to mounted.
* After adding the deployment engine, continue at the current location.

##### Health

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

Repair-WindowsImage -Path "D:\OS\_2022\_Custom\Install\Install\Mount" -ScanHealth

##### Replace the WinRE.wim

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

$WinRE = "D:\OS\_2022\_Custom\Install\Install\Update\Winlib\WinRE.wim"

$CopyTo = "D:\OS\_2022\_Custom\Install\Install\Mount\Windows\System32\Recovery"

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

##### Save image: Install.wim

Save-WindowsImage -Path "D:\OS\_2022\_Custom\Install\Install\Mount"

##### Unmount image: Install.wim

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

Dismount-WindowsImage -Path "D:\OS\_2022\_Custom\Install\Install\Mount" -Discard

CYCLIC OPERATION AREA, END.

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

###### Get WimLib

After going to the official website of <https://wimlib.net>, select a different version: [arm64](https://wimlib.net/downloads/wimlib-1.14.3-windows-aarch64-bin.zip), [x64](https://wimlib.net/downloads/wimlib-1.14.3-windows-x86_64-bin.zip), [x86](https://wimlib.net/downloads/wimlib-1.14.3-windows-i686-bin.zip), and extract it to: D:Wimlib after downloading.

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

* + 1. Extract the WinRE.wim file from Install.wim
* Install.WinRE.Extract.ps1
  + [\Expand\Install\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>
* Copy the code

$Arguments = @(

"extract",

"D:\OS\_2022\sources\install.wim", "1",

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

"--dest-dir=""D:\OS\_2022\_Custom\Install\Install\Update\Winlib"""

)

New-Item -Path "D:\OS\_2022\_Custom\Install\Install\Update\Winlib" -ItemType Directory

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

* + 1. Get all index numbers of Install.wim and replace the old WinRE.wim
* Install.WinRE.Replace.wim.ps1
  + [\Expand\Install\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>
* Copy the code

Get-WindowsImage -ImagePath "D:\OS\_2022\sources\install.wim" -ErrorAction SilentlyContinue | ForEach-Object {

Write-Host " Image name: " -NoNewline

Write-Host $\_.ImageName -ForegroundColor Yellow

Write-Host " The index number: " -NoNewline

Write-Host $\_.ImageIndex -ForegroundColor Yellow

Write-Host "`n Replacement "

$Arguments = @(

"update",

"D:\OS\_2022\sources\install.wim",

$\_.ImageIndex,

"--command=""add 'D:\OS\_2022\_Custom\Install\Install\Update\Winlib\WinRE.wim' '\Windows\System32\Recovery\WinRe.wim'"""

)

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

Write-Host " Finish`n" -ForegroundColor Green

}

##### Rebuilding Install.wim reduces file size

* Install.Rebuild.wim.ps1
  + [\Expand\Install\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>
* Copy the code

$InstallWim = "D:\OS\_2022\sources\install.wim"

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

Write-Host " Image name: " -NoNewline

Write-Host $\_.ImageName -ForegroundColor Yellow

Write-Host " The index number: " -NoNewline

Write-Host $\_.ImageIndex -ForegroundColor Yellow

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

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

Write-Host "Finish`n" -ForegroundColor Green

}

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

Remove-Item -Path $InstallWim

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

Write-Host "Finish" -ForegroundColor Green

} else {

Write-host "Failed" -ForegroundColor Red

}

##### Split, merge, compress, and convert

Solid compression is in ESD file format. If the file exceeds 4GB, it cannot be split and cannot be copied to a FAT32 disk. This is a disadvantage.

Using FAT32 format to store Windows installation boot is the best solution. If the Install.wim file exceeds 4GB and cannot be copied to a FAT32 disk, you need to split the Install.wim file and copy it to a FAT32 disk after the file size is less than 4GB.

It is particularly important to learn how to split and merge, solid compression and conversion.

###### Splitting and merging

* + 1. Splitting

After splitting Install.wim into 4GB file sizes and getting new file names Install.\*.swm, delete the old Install.wim.

* Install.Split.ps1
* [\Expand\Install\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>
* Copy the code

Write-host "Split Install.wim into Install.\*.swm";

Write-host "Splitting" -NoNewline;

Split-WindowsImage -ImagePath "D:\OS\_2022\sources\install.wim" -SplitImagePath "D:\OS\_2022\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:\OS\_2022\sources\install.swm" -PathType leaf) {

Remove-Item -Path "D:\OS\_2022\sources\install.wim"

Write-Host "Done" -ForegroundColor Green

} else {

Write-Host "Failed" -ForegroundColor Red

}

* + 1. Merge

After merging all Install.\*.swm into Install.wim, delete the old Install.\*.swm.

* Install.Merging.ps1
* [\Expand\Install\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>
* Copy the code

Write-host "Merge all Install.\*.swm files into Install.wim";

Get-WindowsImage -ImagePath "D:\OS\_2022\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:\OS\_2022\Sources\install.swm" /swmfile:"D:\OS\_2022\sources\install\*.swm" /SourceIndex:"$($\_.ImageIndex)" /DestinationImageFile:"D:\OS\_2022\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:\OS\_2022\Sources\install.wim" -PathType leaf) {

Get-ChildItem -Path "D:\OS\_2022\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

}

###### Solid compressed ESD format and WIM format conversion

* + 1. Solid compression

After solid compression, you can edit version information and application files, etc.; you cannot mount images, etc. After obtaining the new file install.esd, delete the old Install.wim.

* Install.Compress.ps1
* [\Expand\Install\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>
* Copy the code

Write-host "Solid compressed Install.wim";

Get-WindowsImage -ImagePath "D:\OS\_2022\Sources\install.wim" -ErrorAction SilentlyContinue | ForEach-Object {

Write-Host "Image Name: " -NoNewline; Write-Host $\_.ImageName -ForegroundColor Yellow;

Write-Host "Index Number: " -NoNewline; Write-Host $\_.ImageIndex -ForegroundColor Yellow;

Write-Host "Compressing".PadRight(28) -NoNewline

dism /export-image /SourceImageFile:"D:\OS\_2022\Sources\install.wim" /SourceIndex:"$($\_.ImageIndex)" /DestinationImageFile:"D:\OS\_2022\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:\OS\_2022\Sources\install.esd" -PathType leaf) {

Remove-Item -Path "D:\OS\_2022\Sources\install.wim"

Write-Host "Done" -ForegroundColor Green

} else {

Write-Host "Falied" -ForegroundColor Green

}

* + 1. Convert compressed files to WIM file format
* Install.Convert.ps1
* [\Expand\Install\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>
* Copy the code

Write-host "Convert ESD to WIM";

Get-WindowsImage -ImagePath "D:\OS\_2022\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:\OS\_2022\Sources\install.esd" -SourceIndex $\_.ImageIndex -DestinationImagePath "D:\OS\_2022\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:\OS\_2022\Sources\install.wim" -PathType leaf) {

Remove-Item -Path "D:\OS\_2022\Sources\install.esd"

Write-Host "Done" -ForegroundColor Green

} else {

Write-Host "Falied" -ForegroundColor Green

}

#### Custom encapsulation: boot.wim

##### View Boot.wim details

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

$ViewFile = "D:\OS\_2022\Sources\Boot.wim"

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

##### Specify the path to mount Boot.wim

New-Item -Path "D:\OS\_2022\_Custom\Boot\Boot\Mount" -ItemType directory

##### Start mounting Boot.wim

Default index number: 2

Mount-WindowsImage -ImagePath "D:\OS\_2022\sources\boot.wim" -Index "2" -Path "D:\OS\_2022\_Custom\Boot\Boot\Mount"

##### Language pack

* Automatically install language packs: Get "Component: All installed packages in the image" and match them. After matching the corresponding names, install the local corresponding language pack files.
* When adding languages, different schema versions must be corresponded, and if not, errors are reported during the addition process.

###### Language pack: add

* Boot.Instl.lang.ps1
  + [\Expand\Boot\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>
* Copy the code

$Mount = "D:\OS\_2022\_Custom\Boot\Boot\Mount"

$Sources = "D:\OS\_2022\_Custom\Boot\Boot\Language\Add\zh-CN"

$Initl\_install\_Language\_Component = @()

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

$Initl\_install\_Language\_Component += $\_.PackageName

}

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

$Language = @(

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

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

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

@{ Match = "\*SecureStartup\*"; File = "winpe-securestartup\_zh-CN.cab"; }

@{ Match = "\*ATBroker\*"; File = "winpe-atbroker\_zh-CN.cab"; }

@{ Match = "\*AudioCore\*"; File = "winpe-audiocore\_zh-CN.cab"; }

@{ Match = "\*AudioDrivers\*"; File = "winpe-audiodrivers\_zh-CN.cab"; }

@{ Match = "\*EnhancedStorage\*"; File = "winpe-enhancedstorage\_zh-CN.cab"; }

@{ Match = "\*Narrator\*"; File = "winpe-narrator\_zh-CN.cab"; }

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

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

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

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

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

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

)

ForEach ($Rule in $Language) {

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

ForEach ($Component in $Initl\_install\_Language\_Component) {

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

Write-host " Component name: " -NoNewline

Write-host $Component -ForegroundColor Green

Write-host " Language pack file: " -NoNewline

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

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

try {

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

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

}

break

}

}

}

###### Offline image language: change

* + 1. Change default language, regional settings, and other international settings

Language Tag: zh-CN

Dism /Image:"D:\OS\_2022\_Custom\Boot\Boot\Mount" /Set-AllIntl:zh-CN

* + 1. View available language settings

Dism /Image:"D:\OS\_2022\_Custom\Boot\Boot\Mount" /Get-Intl

###### Components: All packages installed in the image

* + 1. View

Get-WindowsPackage -Path "D:\OS\_2022\_Custom\Boot\Boot\Mount" | Out-GridView

* + 1. Export to Csv

$SaveTo = "D:\OS\_2022\_Custom\Boot\Boot\Report.Components.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS\_2022\_Custom\Boot\Boot\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green

###### Language packs: sync to ISO installer

Copy-Item -Path "D:\OS\_2022\_Custom\Boot\Boot\Mount\sources\zh-CN" -Destination "D:\OS\_2022\sources\zh-CN" -Recurse -Force

###### Regenerate Lang.ini

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

* + 1. Regenerate the mounted directory lang.ini

Re-generated Lang.ini file location: D:\OS\_2022\_Custom\Boot\Boot\Mount\Sources\lang.ini

Dism /image:"D:\OS\_2022\_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS\_2022\_Custom\Boot\Boot\Mount"

* + 1. After regenerating lang.ini, synchronize to the installer

Re-generated Lang.ini file location: D:\OS\_2022\Sources\lang.ini

Dism /image:"D:\OS\_2022\_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS\_2022"

##### Cumulative updates

To prepare the cumulative updates file available, change the example file name: KB\_Boot.cab

###### Add

$KBPath = "D:\OS\_2022\_Custom\Boot\Boot\Update\KB\_Boot.cab"

Add-WindowsPackage -Path "D:\OS\_2022\_Custom\Boot\Boot\Mount" -PackagePath $KBPath

###### Delete

$KBPath = "D:\OS\_2022\_Custom\Boot\Boot\Update\KB\_Boot.cab"

Remove-WindowsPackage -Path "D:\OS\_2022\_Custom\Boot\Boot\Mount" -PackagePath $KBPath

###### Solid update

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

Dism /image:"D:\OS\_2022\_Custom\Boot\Boot\Mount" /cleanup-image /StartComponentCleanup /ResetBase

* + 1. Clean components after curing and updating

$Mount = "D:\OS\_2022\_Custom\Boot\Boot\Mount"

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

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

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

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

}

}

##### Drive

##### Save image: Boot.wim

Save-WindowsImage -Path "D:\OS\_2022\_Custom\Boot\Boot\Mount"

##### Unmount image: Boot.wim

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

Dismount-WindowsImage -Path "D:\OS\_2022\_Custom\Boot\Boot\Mount" -Discard

#### Deployment engine

* + - * Learn about "Automatically Adding Languages Installed in Windows Systems", learn: <https://github.com/ilikeyi/Multilingual>, how to download:
      * After entering the website, click "Code", "Download Compressed Package", and after the download is completed, you will get the main.zip compressed package file.
      * Go to the <https://github.com/ilikeyi/Multilingual/releases> download page, select the available version: 1.1.1.1, select the download source code format: zip, and get the Multilingual-1.1.1.1.zip compressed package file after the download is completed;
      * Unzip the downloaded main.zip or Multilingual-1.1.1.1.zip to: D:\Multilingual-1.1.1.1, and rename: D:\Multilingual
      * Learn "[Unattended Windows Setup Reference](https://learn.microsoft.com/en-us/windows-hardware/customize/desktop/unattend/)", Intervene in the installation process by leaving it unattended.

##### Add method

###### Add to ISO installation media

* + 1. Unattended
       1. Add to: [ISO]:\Autounattend.xml

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

Copy D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml to D:\OS\_2022\Autounattend.xml

Copy-Item "D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS\_2022\Autounattend.xml" -Force

* + - 1. Add to: [ISO]:\Sources\Unattend.xml

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

Copy D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml to D:\OS\_2022\Sources\Unattend.xml

Copy-Item "D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS\_2022\Sources\Unattend.xml" -Force

* + - 1. Add to: [ISO]:\sources\$OEM$\$$\Panther\unattend.xml

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

* + - * 1. Create $OEM$ path

New-Item -Path "D:\OS\_2022\sources\`$OEM$\`$$\Panther" -ItemType Directory

* + - * 1. Copy

Copy D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml to D:\OS\_2022\Sources\$OEM$\Panther\Unattend.xml

Copy-Item "D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS\_2022\sources\`$OEM$\`$$\Panther\Unattend.xml" -Force

* + 1. Deployment engine: add

Add "Automatically add installed languages for Windows systems" to D:\OS\_2022\sources\$OEM$\$1\Yi\Engine in the directory.

* + - 1. Deployment Engine: Copy

Copy D:\Multilingual\Engine to D:\OS\_2022\Sources\$OEM$\$1\Yi\Engine

Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS\_2022\sources\`$OEM$\`$1\Yi\Engine" -Recurse -Force

* + - 1. Deployment engine: custom deployment tags

$Flag = @(

"Is\_Mark\_Sync" # Allow global search and synchronization of deployment tags

# Prerequisite deployment

# "Auto\_Update" # Allow automatic updates

# "Use\_UTF8" # Beta: Global language support using Unicode UTF-8

"Disable\_Network\_Location\_Wizard" # Network Location Wizard

"Disable\_Cleanup\_Appx\_Tasks" # Appx Cleanup and maintenance tasks

"Disable\_Cleanup\_On\_Demand\_Language" # Prevent cleanup of unused on-demand feature language packs

"Disable\_Cleanup\_Unsed\_Language" # Prevent cleaning of unused language packs

"Prerequisites\_Reboot" # Restart your computer

# Complete first deployment

# "Popup\_Engine" # Allow the deployment engine main interface to pop up for the first time

# "Allow\_First\_Pre\_Experience" # Allow first preview, as planned

"Reset\_Execution\_Policy" # Restore PowerShell execution policy: Restricted

"Clear\_Solutions" # Delete the entire solution

"Clear\_Engine" # Delete the deployment engine and keep the others

# "First\_Experience\_Reboot" # Restart your computer

)

ForEach ($item in $Flag) {

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

New-Item -Path "D:\OS\_2022\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -ErrorAction SilentlyContinue | Out-Null

Out-File -FilePath "D:\OS\_2022\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow\$($item)" -Encoding utf8 -ErrorAction SilentlyContinue

}

###### Add to mounted

Through "[Custom encapsulation: Install.wim](#_Custom_encapsulation:_Install.wim)", execute "[Start mounting Install.wim](#_Start_mounting_Install.wim)" and mount to: D:\OS\_2022\_Custom\Install\Install\Mount

* + 1. Unattended

Copy D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml to D:\OS\_2022\_Custom\Install\Install\Mount\Panther\Unattend.xml

Copy-Item "D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS\_2022\_Custom\Install\Install\Mount\Panther" -Force

* + 1. Deployment engine: add

Add "Automatically add languages installed on Windows systems" to the D:\OS\_2022\_Custom\Install\Install\Mount\Yi\Engine directory.

* + - 1. Deployment Engine: Copy

Copy D:\Multilingual\Engine to D:\OS\_2022\_Custom\Install\Install\Mount\Yi\Engine

Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS\_2022\_Custom\Install\Install\Mount\Yi\Engine" -Recurse -Force

* + - 1. Deployment engine: custom deployment tags

$Flag = @(

"Is\_Mark\_Sync" # Allow global search and synchronization of deployment tags

# Prerequisite deployment

# "Auto\_Update" # Allow automatic updates

# "Use\_UTF8" # Beta: Global language support using Unicode UTF-8

"Disable\_Network\_Location\_Wizard" # Network Location Wizard

"Disable\_Cleanup\_Appx\_Tasks" # Appx Cleanup and maintenance tasks

"Disable\_Cleanup\_On\_Demand\_Language" # Prevent cleanup of unused on-demand feature language packs

"Disable\_Cleanup\_Unsed\_Language" # Prevent cleaning of unused language packs

"Prerequisites\_Reboot" # Restart your computer

# Complete first deployment

# "Popup\_Engine" # Allow the deployment engine main interface to pop up for the first time

# "Allow\_First\_Pre\_Experience" # Allow first preview, as planned

"Reset\_Execution\_Policy" # Restore PowerShell execution policy: Restricted

"Clear\_Solutions" # Delete the entire solution

"Clear\_Engine" # Delete the deployment engine and keep the others

# "First\_Experience\_Reboot" # Restart your computer

)

ForEach ($item in $Flag) {

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

New-Item -Path "D:\OS\_2022\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -ErrorAction SilentlyContinue | Out-Null

Out-File -FilePath "D:\OS\_2022\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow\$($item)" -Encoding utf8 -ErrorAction SilentlyContinue

}

##### Deployment Engine: Advanced

###### Deployment engine: adding process

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

###### Unattended solution

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

* D:\OS\_2022\Autounattend.xml
* D:\OS\_2022\Sources\Unattend.xml
* D:\OS\_2022\sources\$OEM$\$$\Panther\unattend.xml
* D:\OS\_2022\_Custom\Install\Install\Mount\Panther\Unattend.xml
  + 1. Multilingual or monolingual

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

* + - 1. Multi-language

<UILanguage>%OSDUILanguage%</UILanguage>

<InputLocale>%OSDInputLocale%</InputLocale>

<SystemLocale>%OSDSystemLocale%</SystemLocale>

<UILanguage>%OSDUILanguage%</UILanguage>

<UILanguageFallback>%OSDUILanguageFallback%</UILanguageFallback>

<UserLocale>%OSDUserLocale%</UserLocale>

* + - 1. Monolingual

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

<UILanguage>zh-CN</UILanguage>

<InputLocale>zh-CN</InputLocale>

<SystemLocale>zh-CN</SystemLocale>

<UILanguage>zh-CN</UILanguage>

<UILanguageFallback>zh-CN</UILanguageFallback>

<UserLocale>zh-CN</UserLocale>

* + 1. User plan

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

* + - 1. Self-created user Administrator

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

<UserAccounts>

<LocalAccounts>

<LocalAccount wcm:action="add">

<Password>

<Value></Value>

<PlainText>true</PlainText>

</Password>

<Description>Administrator</Description>

<DisplayName>Administrator</DisplayName>

<Group>Administrators</Group>

<Name>Administrator</Name>

</LocalAccount>

</LocalAccounts>

</UserAccounts>

<AutoLogon>

<Password>

<Value></Value>

<PlainText>true</PlainText>

</Password>

<Enabled>true</Enabled>

<Username>Administrator</Username>

</AutoLogon>

* + - 1. Custom user

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

* + - 1. Delete

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

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

* + - 1. Replace

From the beginning <OOBE> to </OOBE>

<OOBE>

<ProtectYourPC>3</ProtectYourPC>

<HideEULAPage>true</HideEULAPage>

<HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>

</OOBE>

### Generate ISO

#### Download OScdimg

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

* 1. x64

<https://github.com/ilikeyi/Solutions/raw/refs/heads/main/_Encapsulation/Modules/AIO/Oscdimg/amd64/oscdimg.exe>

* 1. x86

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

* 1. arm64

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

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

* ISO.ps1
  + [\Expand\ISO.ps1](Expand/ISO.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.2022/Expand/ISO.ps1>
* Copy the code

$Oscdimg = "D:\Oscdimg.exe"

$ISO = "D:\Win2022"

$Volume = "Win2022"

$SaveTo = "D:\Win2022.iso"

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

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

## Common problem

### Clean all mounts to

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

Dismount-WindowsImage -Path "D:\OS\_2022\_Custom\Install\Install\Mount" -Discard

Dismount-WindowsImage -Path "D:\OS\_2022\_Custom\Install\WinRE\Mount" -Discard

Dismount-WindowsImage -Path "D:\OS\_2022\_Custom\Boot\Boot\Mount" -Discard

### Fix the problem of abnormal mounting

1. View mounted

Get-WindowsImage -Mounted

1. Delete the DISM mount record saved in the registry

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

1. Delete all resources associated with the corrupted mounted image

Clear-WindowsCorruptMountPoint

Dism /cleanup-wim

### Clean up

A large number of temporary files will be generated during the packaging process. Installation files will be temporarily released when installing InBox Apps applications, installing cumulative updates, and installing language packs. Therefore, unscheduled cleaning of outdated ones will occupy a large amount of disk space for a long time. It is recommended that you try the following methods to achieve this. Cleanup plan to free up more space:

1. Common logs
   1. Clean using the command line

$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. Manual deletion
     1. DISM log

Using the "Disk Cleanup" function, the logs generated by DISM cannot be cleaned and need to be deleted manually. Path: {system disk}:\Windows\Logs\DISM

* + 1. Temporary directory

Using the "Disk Cleanup" function, files in the temporary directory cannot be cleaned and manual operation is required. Run: %Temp% to quickly locate and open the temporary directory. Path: {system disk}:\Users\{username}\AppData\Local\ Temp

* + 1. Clear the command line records of "Terminal"

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

After cleaning up command line records, you need to restart the "Terminal" to take effect.

1. Disk cleanup

Run cleanmgr, selecting the disks and types to clean.

## Known issues

1. Add Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~zh-CN~.cab to Windows Server 2022 Standard Core, Windows Server 2022 Datacenter Core will add Microsoft-Windows-PowerShell-ISE-FOD -Package~31bf3856ad364e35~amd64~zh-CN~10.0.20348.1, an error will be reported when deleting it, and the operation is not recommended for the time being.

Yi’ Solutions

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

* Yi's official website | <https://fengyi.tel/solutions>
* Github | <https://github.com/ilikeyi/solutions>

Author: Yi

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

Document version: 1.7

Translation: Chinese to English version

Initial public offering time: 4 / 2023

All scripts included in the document, last tested: 11 / 2024

Document last updated: 11 / 2024

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