Yi’ Solutions

# Microsoft Windows 11 25H2

PACKAGING TUTORIAL: Different system versions have different packaging methods. The packaging process includes: "Language pack: add, associate, delete", "Drive: add, delete", "Cumulative update: add, delete", "InBox Apps: add, Update, mark" 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. N ways to speed up Windows operating system Page 5
   1. Turn off Windows Security Center Page 5
   2. Turn off virtualization-based security Page 6
6. ISO tools Page 6
7. Command line Page 6
8. Requirements Page 7
9. System installation package Page 7
10. Language pack Page 9
    1. Learn Page 9
    2. Language pack: Download Page 11
11. InBox Apps Page 11
    * 1. Language Pack: Extract Page 12
12. Language pack: Ready Page 12
13. Language pack: Scheme Page 12
14. Execute the extract command Page 13
    * 1. Custom encapsulation Page 19
15. Custom encapsulation Install.wim Page 19
16. View Install.wim details Page 19
17. Specify the path to mount Install.wim Page 19
18. Start mounting Install.wim Page 19
    1. Custom encapsulation WinRE.wim Page 20
       1. View WinRE.wim details Page 20
       2. Specify the path to mount WinRE.wim Page 20
       3. Start mounting WinRE.wim Page 20
       4. Language pack: WinRE Page 20
          1. Language pack: add Page 20
          2. Offline image language: change Page 22
             1. Change default language, regional settings, and other international settings Page 22
             2. View available language settings Page 22
          3. Components: All packages installed in the image Page 22
       5. Cumulative updates Page 22
          1. Add Page 23
          2. Delete Page 23
          3. Solid update Page 23
             1. Clean components after curing and updating Page 23
       6. Drive Page 23
       7. Save image: WinRE.wim Page 23
       8. Unmount image: WinRE.wim Page 23
       9. After rebuilding WinRE.wim, the file size can be reduced Page 23
       10. Backup WinRE.wim Page 24
       11. Replace WinRE.wim within the Install.wim image Page 25

1. Language pack Page 25

* 1. Language pack: add Page 25
  2. Offline image language: change Page 31
     1. Change default language, regional settings, and other international settings Page 31
     2. View available language settings Page 31
  3. Components: All packages installed in the image Page 31

1. InBox Apps Page 32
   1. Learn Page 32
   2. InBox Apps: Pre-install the application Page 32
   3. InBox Apps: Installed Page 34
   4. Remove all installed pre-applications Page 34
   5. Installing "Local Language Experience Packs (LXPs)" to offline images marks: installed languages Page 35
   6. Offline image has language tag installed: How to add it Page 35
   7. InBox Apps: Install Page 38
   8. ffline image has language tag installed: Remove Page 48
   9. InBox Apps: optimization Page 49
2. Cumulative updates Page 49
   1. Feature Enablement Package Page 49
   2. Initial release cumulative update Page 50
   3. Other versions Page 50
   4. Solidify Updated Page 51
      1. Clean components after curing and updating Page 51
3. Drive Page 51
4. Deployment engine: Add Page 52
5. Health Page 52
6. Replace WinRE.wim Page 52
7. Save image: Install.wim Page 52
8. Unmount image: Install.wim Page 52
9. How to batch replace WinRE.wim in all index numbers in Install.wim Page 52
   1. Obtain WimLib Page 52
   2. How to extract and update WinRE.wim in Install.wim Page 52
10. Rebuilding Install.wim reduces file size Page 53
11. Split, merge, compress, and convert Page 54
    1. Splitting and merging Page 54
       1. Splitting Page 54
       2. Merge Page 55
    2. Solid compressed ESD format and WIM format conversion Page 56
       1. Solid compression Page 56
       2. Convert compressed files to WIM file format Page 56
12. Custom encapsulation boot.wim Page 57
13. View Boot.wim details Page 57
14. Specify the path to mount Boot.wim Page 57
15. Start mounting Boot.wim Page 57
16. Language pack: Boot Page 57
    1. Language pack: Add Page 58
    2. Offline image language: change Page 59
       1. Change default language, regional settings, and other international settings Page 59
       2. View available language settings Page 59
    3. Components: All packages installed in the image Page 59
    4. Language packs: sync to ISO installer Page 59
    5. Regenerate Lang.ini Page 60
       1. Regenerate the mounted directory lang.ini Page 60
       2. After regenerating lang.ini, sync to the installer Page 60
17. Cumulative updates Page 60
    1. Add Page 60
    2. Delete Page 60
    3. Solid update Page 60
       1. Clean components after curing and updating Page 60
18. Drive Page 61
19. Other Page 61
    1. Bypass TPM check during installation Page 61
20. Save image: Boot.wim Page 61
21. Unmount image: Boot.wim Page 61
22. Deployment engine Page 61
23. Add method Page 62
24. Deployment Engine: Advanced Page 65
    * 1. ISO Page 67
25. Generate ISO Page 67
26. Bypass TPM installation check Page 68
27. Common problem Page 68
28. Clean all mounts to Page 68
29. Fix the problem of abnormal mounting Page 68
30. Clean up Page 68
31. Known issues Page 69

## 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 25H2, 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.

##### N ways to speed up Windows operating system

When processing packaging tasks, installing cumulative updates, installing drivers, and installing applications in InBox Apps, a large number of temporary files will be generated. The following methods can be used to speed up the system:

###### Turn off Windows Security Center

* Turning on Windows Security Center will scan files and take up a lot of CPU.
* During the test: It took 1 hour and 22 minutes before it was turned off, and 20 minutes after it was turned off.

How to turn off:

Green is the command line, hold down the Windows key and press R to launch Run.

* + 1. Open Windows Security Center or run: windowsdefender:
    2. Select "Virus & threat protection" or run: windowsdefender://threat
    3. Find "Virus and Threat Protection Settings", click "Manage Settings" or run: windowsdefender://threatsettings. It is recommended that you turn off some features:
       1. Real-time protection
       2. Cloud-provided protection
       3. Automatically submit samples
       4. Tamper Protection
    4. When not in the package, it is recommended that you turn on Windows Security Center.

###### Turn off virtualization-based security

Even after closing Windows Security Center, virtualization-based security is still running, and the system running speed will be greatly reduced. The speed improvement is obvious after closing it.

* + 1. After running, restart your computer

dism /Online /Disable-Feature:microsoft-hyper-v-all /NoRestart

dism /Online /Disable-Feature:IsolatedUserMode /NoRestart

dism /Online /Disable-Feature:Microsoft-Hyper-V-Hypervisor /NoRestart

dism /Online /Disable-Feature:Microsoft-Hyper-V-Online /NoRestart

dism /Online /Disable-Feature:HypervisorPlatform /NoRestart

mountvol X: /s

cmd /c copy /y %WINDIR%\System32\SecConfig.efi X:\EFI\Microsoft\Boot\SecConfig.efi

bcdedit /create {0cb3b571-2f2e-4343-a879-d86a476d7215} /d "DebugTool" /application osloader

bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} path "\EFI\Microsoft\Boot\SecConfig.efi"

bcdedit /set {bootmgr} bootsequence {0cb3b571-2f2e-4343-a879-d86a476d7215}

bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} loadoptions DISABLE-LSA-ISO,DISABLE-VBS

bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} device partition=X:

mountvol X: /d

bcdedit /set hypervisorlaunchtype off

* + 1. View Status

Run Msinfo32 and check the "Virtualization-based Security" status in the system summary.

##### ISO tools

Use a software that can edit ISO files, such as: PowerISO, DAEMON Tools, ISO Workshop;

##### 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

Keywords: iteration, cross-version, major version, cumulative update, initial release

* 1. illustrate
     1. Please re-create the image for each version update. For example, when jumping from 23H2 to 25H2, avoid other compatibility issues and do not create based on the old image;
     2. The regulation has been clearly communicated to packagers in various forms by some OEMs, and direct upgrades from iterative versions are not allowed;
     3. Please use "Initial Version" and "Developer Edition" for production. There was a brief appearance in the official Microsoft documentation that the initial version must be used in production, but this sentence was later deleted in the official documentation;
     4. After installing the language pack, you must re-add the cumulative update (the same version number), and if you do not add the cumulative update, problems such as "garbled characters" and "interface flashback" will occur.
     5. Evolution process: Windows 11 25H2, Build 26200.6584 + ( " KB5054156 Checkpoint Update" + " KB5065789 Monthly Cumulative Update" ) = OS Build 26200.6725
  2. Prepare to download the initial or developer version
     1. x64
        1. Filename: en-us\_windows\_11\_consumer\_editions\_version\_25h2\_x64\_dvd\_9934ee4c.iso

Download Link: <https://software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/26200.6584.250915-1905.25h2_ge_release_svc_refresh_CLIENT_CONSUMER_x64FRE_en-us.iso>

List of files: <https://files.rg-adguard.net/file/8d625f81-4e0a-a542-a5ac-177e3fc4548e>

* + - 1. Filename: en-us\_windows\_11\_business\_editions\_version\_25h2\_x64\_dvd\_41c521e7.iso

List of files: <https://files.rg-adguard.net/file/136ef757-5e4a-5964-95a3-3803f2de0c8b>

* + - 1. Filename: en-us\_windows\_11\_iot\_enterprise\_version\_25h2\_x64\_dvd\_67098cd6.iso

List of files: <https://files.rg-adguard.net/file/fce4ab0f-1033-19b9-42e7-860501a6d3a1>

* + - 1. Filename: [en-us\_windows\_11\_enterprise\_ltsc\_2024\_x64\_dvd\_965cfb00.iso](https://files.rg-adguard.net/file/142ca376-487f-e858-a606-e120e70b9d02)

List of files: <https://files.rg-adguard.net/file/142ca376-487f-e858-a606-e120e70b9d02>

* + - 1. Filename: [en-us\_windows\_11\_iot\_enterprise\_ltsc\_2024\_x64\_dvd\_f6b14814.iso](https://files.rg-adguard.net/file/9160d5cf-f480-a1c9-a62c-b75ab0708d2c)

List of files: <https://files.rg-adguard.net/file/9160d5cf-f480-a1c9-a62c-b75ab0708d2c>

* + 1. arm64
       1. Filename: en-us\_windows\_11\_consumer\_editions\_version\_25h2\_arm64\_dvd\_de2d8bf8.iso

Download Link: [https ://software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/26200.6584.250915-1905.25h2\_ge\_release\_svc\_refresh\_CLIENT\_CONSUMER\_A64FRE\_ko-kr.iso](https%20:/software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/26200.6584.250915-1905.25h2_ge_release_svc_refresh_CLIENT_CONSUMER_A64FRE_ko-kr.iso)

List of files: <https://files.rg-adguard.net/file/a6329578-0cd5-4256-4377-fc8e82882184>

* + - 1. Filename: en-us\_windows\_11\_business\_editions\_version\_25h2\_arm64\_dvd\_8afc9b39.iso

List of files: <https://files.rg-adguard.net/file/e7e9a4e0-7bc8-1bdd-5671-fb2feaf7e46e>

* + - 1. Filename: en-us\_windows\_11\_iot\_enterprise\_version\_25h2\_arm64\_dvd\_59e0d737.iso

List of files: <https://files.rg-adguard.net/file/c1600ec1-ccf1-c08d-bb46-d9daca25c10c>

* + - 1. Filename: [en-us\_windows\_11\_iot\_enterprise\_ltsc\_2024\_arm64\_dvd\_ec517836.iso](https://files.rg-adguard.net/file/de2234af-1514-ff50-50af-5a3395549c42)

List of files: <https://files.rg-adguard.net/file/de2234af-1514-ff50-50af-5a3395549c42>

* 1. After the sample download en-us\_windows\_11\_consumer\_editions\_version\_25h2\_x64\_dvd\_9934ee4c.iso, Unzip to: D:\en-us\_windows\_11\_consumer\_editions\_version\_25h2\_x64\_dvd\_9934ee4c

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 is complete, change the directory D:\en-us\_windows\_11\_consumer\_editions\_version\_25h2\_x64\_dvd\_9934ee4c change to D:\OS\_11
  2. All scripts and all paths are set to D:\OS\_11 as the image source.
  3. Installation configuration
     1. Learn: [Windows Setup Edition Configuration and Product ID Files ( EI.cfg and PID.txt )](https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/windows-setup-edition-configuration-and-product-id-files--eicfg-and-pidtxt?view=windows-11)
     2. Known issues
        1. When there is no Ei.cfg, ISO boot installation will report an error when selecting certain versions, prompting: Windows cannot find the Microsoft Software License terms. Make sure the insttallation sources are valid and restart the installation.
        2. How to solve it? Add ei.cfg to D:\OS\_11\Sources and create ei.cfg:

@"

[Channel]

volume

[VL]

1

"@ | Out-File -FilePath "D:\OS\_11\sources\EI.CFG" -Encoding Ascii

##### Language pack

###### Learn

As you read, please understand the important highlights of "Blue".

* + 1. [Languages overview](https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/languages-overview?view=windows-11)
    2. [Add languages to a Windows 11 image](https://learn.microsoft.com/en-us/azure/virtual-desktop/windows-11-language-packs)
    3. [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 Pack: 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.

Known regional associations:

* + - * 1. Region: ar-sa, Optional associated areas: en-US, associated installation package:

Microsoft-Windows-LanguageFeatures-Basic-es-us-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~.cab

* + - * 1. Region: bg-bg, Optional associated areas: en-US, associated installation package:

Microsoft-Windows-LanguageFeatures-Basic-es-us-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~.cab

* + - * 1. Region: da-dk, Optional associated areas: en-US, associated installation package:

Microsoft-Windows-LanguageFeatures-Basic-es-us-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~.cab

* + - * 1. Region: el-gr, Optional associated areas: en-US, associated installation package:

Microsoft-Windows-LanguageFeatures-Basic-es-us-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~.cab

* + - * 1. Region: fr-ca, Optional associated areas: en-US, fr-fr, associated installation package:

Microsoft-Windows-LanguageFeatures-Basic-es-us-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-Basic-fr-fr-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-Handwriting-fi-fi-Package~31bf3856ad364e35~amd64~~.cab

* + - * 1. Region: he-il, Optional associated areas: en-US, associated installation package:

Microsoft-Windows-LanguageFeatures-Basic-es-us-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~.cab

* + - * 1. Region: ru-ru, Optional associated areas: en-US, associated installation package:

Microsoft-Windows-LanguageFeatures-Basic-es-us-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~.cab

* + - * 1. Region: th-th, Optional associated areas: en-US, associated installation package:

Microsoft-Windows-LanguageFeatures-Basic-es-us-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~.cab

* + - * 1. Region: uk-ua, Optional associated areas: en-US, associated installation package:

Microsoft-Windows-LanguageFeatures-Basic-es-us-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~.cab

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

Microsoft-Windows-LanguageFeatures-Basic-zh-hk-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-Speech-zh-hk-Package~31bf3856ad364e35~amd64~~.cab

Microsoft-Windows-LanguageFeatures-TextToSpeech-zh-hk-Package~31bf3856ad364e35~amd64~~.cab

* + - 1. Other region-specific requirements

When triggering a known area, a specific "package" needs to be added.

* + - * 1. Region: zh-TW

Package: Microsoft-Windows-InternationalFeatures-Taiwan-Package~31bf3856ad364e35~amd64~~.cab

Description: Supplemental support for Taiwan date formatting requirements. Package will be provided to customers located in Taiwan.

Recommendation: Preinstall only on devices shipping to the Taiwan market. Not installing this capability on devices causes any API calls to that use the Taiwan calendar to fail.

THERE IS CONTROVERSY:

* During the test, it was found that this package was not installed in the original image of Microsoft's official original Windows 11 However, there are known issues in the recommended items. In the end, it is consistent with the official Microsoft original version, and the packager can freely choose to install it. or not.

###### Language pack: Download

* + 1. Download: <https://software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/26100.1.240331-1435.ge_release_amd64fre_CLIENT_LOF_PACKAGES_OEM.iso>

List of files: <https://files.rg-adguard.net/file/025cfc5d-f5fa-7d00-246e-76c04a40e210>

##### InBox Apps

* 1. Download: <https://software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/26100.6584.250904-1728.ge_release_svc_prod1_amd64fre_InboxApps.iso>

List of files: <https://files.rg-adguard.net/file/ada764e8-717a-d080-099b-d3aa44b84dab>

* 1. Fix missing items
     1. Microsoft.HEVCVideoExtension

26100.6584.250904-1728.ge\_release\_svc\_prod1\_amd64fre\_InboxApps.iso does not contain the Microsoft.HEVCVideoExtension\_8wekyb3d8bbwe.appxbundle installation package file, you must use this version: 2.4.13.0, which can be obtained in the following ways:

* + - 1. If the file cannot be obtained, please download it manually:
         1. Microsoft.HEVCVideoExtension\_8wekyb3d8bbwe.appxbundle

Direct download

* <https://github.com/ilikeyi/Solutions/raw/refs/heads/main/_Learn/Download.Package/Windows.11.25H2/Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.appxbundle>

Open the website: <https://store.rg-adguard.net>

a). Select a URL (link). b). Select a channel (Retail). c). Enter keywords: https://www.microsoft.com/store/productId/9N4WGH0Z6VHQ

Search results: Microsoft.HEVCVideoExtension\_2.4.13.0.\*.appxbundle

After downloading, save it to the "Desktop" and rename it: Microsoft.HEVCVideoExtension\_8wekyb3d8bbwe.appxbundle

* + - * 1. License

Pre-saved licenses

Copy {location}:\YiSolutions\\_Learn\Packaging.tutorial\OS.11\25H2\Expand\License\15a6383d188a454ca98e15931ce48dc4\_License1.xml to the desktop. If it is not available locally, download it online.

After downloading the file through the online download link, save it to the desktop

<https://github.com/ilikeyi/Solutions/raw/refs/heads/main/_Learn/Download.Package/Windows.11.25H2/15a6383d188a454ca98e15931ce48dc4_License1.xml>

* + 1. After downloading through the above method, you will get the following 4 files:
       1. Microsoft.HEVCVideoExtension\_8wekyb3d8bbwe.appxbundle
       2. 15a6383d188a454ca98e15931ce48dc4\_License1.xml
    2. Use ISO editing tools to edit 26100.6584.250904-1728.ge\_release\_svc\_prod1\_amd64fre\_InboxApps.iso and add the prepared files to the [ISO]:\packages directory;

### Language Pack: Extract

#### Language pack: Ready

Mount 26100.1.240331-1435.ge\_release\_amd64fre\_CLIENT\_LOF\_PACKAGES\_OEM.iso or unzipped to any location;

#### Language pack: Scheme

##### Add

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

##### Delete

* 1. Language name: English - United States, Region: 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.11/25H2/Expand/Extract.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Extract.ps1)
* Copy the code

Function Extract\_Language

{

param($Act, $NewLang, $Expand, $ISO, $SaveTo)

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-Client-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-InternetExplorer-Optional-Package~31bf3856ad364e35~AMD64~{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-Notepad-System-FoD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

"LanguagesAndOptionalFeatures\Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

"LanguagesAndOptionalFeatures\Microsoft-Windows-MSPaint-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

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

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

"LanguagesAndOptionalFeatures\Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

"LanguagesAndOptionalFeatures\HyperV-OptionalFeature-VirtualMachinePlatform-Client-Disabled-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-DirectoryServices-ADAM-Client-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-DirectoryServices-ADAM-Client-FOD-Package~31bf3856ad364e35~wow64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-EnterpriseClientSync-Host-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

"LanguagesAndOptionalFeatures\Microsoft-Windows-ProjFS-OptionalFeature-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-SenseClient-FoD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-SnippingTool-FoD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-SimpleTCP-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-SmbDirect-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-Telnet-Client-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-TerminalServices-AppServer-Client-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-TerminalServices-AppServer-Client-FOD-Package~31bf3856ad364e35~wow64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-TFTP-Client-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"LanguagesAndOptionalFeatures\Microsoft-Windows-VBSCRIPT-FoD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

"LanguagesAndOptionalFeatures\Microsoft-Windows-WinOcr-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

"LanguagesAndOptionalFeatures\Microsoft-Windows-InternationalFeatures-{Specific}-Package~31bf3856ad364e35~amd64~~.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-connectivity\_{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-appxdeployment\_{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-windowsupdate\_{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-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

Foreach ($item in $Expand) {

$Language = @()

Foreach ($itemList in $AdvLanguage) {

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

Foreach ($PrintLang in $itemList.Rule) {

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

}

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

}

}

}

}

$Extract\_language\_Pack = @(

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

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

)

ForEach ($item in $Extract\_language\_Pack) {

Extract\_Language -Act $item.Act -NewLang $item.Tag -Expand $item.Scope -ISO "Auto" -SaveTo "D:\OS\_11\_Custom"

}

### Custom encapsulation

#### Custom encapsulation Install.wim

##### View Install.wim details

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

$ViewFile = "D:\OS\_11\Sources\Install.wim"

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

LOOP OPERATING AREA, START,

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

New-Item -Path "D:\OS\_11\_Custom\Install\Install\Mount" -ItemType directory

##### Start mounting Install.wim

Default index number: 1

Mount-WindowsImage -ImagePath "D:\OS\_11\sources\install.wim" -Index "1" -Path "D:\OS\_11\_Custom\Install\Install\Mount"

PROCESS FILES WITHIN 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;
* Synchronize to all index numbers to reduce the size of Install.wim, learn "[How to batch replace WinRE.wim in all index numbers 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\_11\_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\_11\_Custom\Install\WinRE\Mount" -ItemType directory

* + 1. Start mounting WinRE.wim

Default index number: 1

Mount-WindowsImage -ImagePath "D:\OS\_11\_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim" -Index "1" -Path "D:\OS\_11\_Custom\Install\WinRE\Mount"

* + 1. Language pack: WinRE
* 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.11/25H2/Expand/Install/WinRE/WinRE.Instl.lang.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/WinRE/WinRE.Instl.lang.ps1)
* Copy the code

$Mount = "D:\OS\_11\_Custom\Install\WinRE\Mount"

$Sources = "D:\OS\_11\_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 = "\*WinPE-appxdeployment\*"; File = "winpe-appxdeployment\_zh-CN.cab"; }

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

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

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

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

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

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

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

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

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

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

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

@{ Match = "\*WinPE-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

}

}

}

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

Region: zh-CN

Dism /Image:"D:\OS\_11\_Custom\Install\WinRE\Mount" /Set-AllIntl:"zh-CN"

* + - * 1. View available language settings

Dism /Image:"D:\OS\_11\_Custom\Install\WinRE\Mount" /Get-Intl

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

Get-WindowsPackage -Path "D:\OS\_11\_Custom\Install\WinRE\Mount" | Out-GridView

* + - * 1. Export to csv

$SaveTo = "D:\OS\_11\_Custom\Install\WinRE\Report.Components.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS\_11\_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\_11\_Custom\Install\WinRE\Update\KB\_WinRE.cab"

Add-WindowsPackage -Path "D:\OS\_11\_Custom\Install\WinRE\Mount" -PackagePath $KBPath

* + - 1. Delete

$KBPath = "D:\OS\_11\_Custom\Install\WinRE\Update\KB\_WinRE.cab"

Remove-WindowsPackage -Path "D:\OS\_11\_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\_11\_Custom\Install\WinRE\Mount" /cleanup-image /StartComponentCleanup /ResetBase

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

$Mount = "D:\OS\_11\_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\_11\_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\_11\_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.11/25H2/Expand/Install/WinRE/WinRE.Rebuild.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/WinRE/WinRE.Rebuild.ps1)
* Copy the code

$FileName = "D:\OS\_11\_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.11/25H2/Expand/Install/WinRE/WinRE.Backup.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/WinRE/WinRE.Backup.ps1)
* Copy the code

$WimLibPath = "D:\OS\_11\_Custom\Install\Install\Update\Winlib"

$FileName = "D:\OS\_11\_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 mount Install.wim "[Replace WinRE.wim](#_Replace_WinRE.wim)";
* Learn "[Get all index numbers of Install.wim and replace the old WinRE.wim](#_Get_all_index)".

Process the files in the Install.wim image and 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.11/25H2/Expand/Install/Install.Instl.lang.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/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 = @()

$Script:Search\_File\_Order = @(

@{

Name = "Fonts"

Description = "Fonts"

Rule = @( "\*Fonts\*"; )

}

@{

Name = "Basic"

Description = "Basic"

Rule = @( "\*LanguageFeatures-Basic\*"; "\*Client\*Language\*Pack\*"; )

}

@{

Name = "OCR"

Description = "Optical character recognition"

Rule = @( "\*LanguageFeatures-OCR\*"; )

}

@{

Name = "Handwriting"

Description = "Handwriting recognition"

Rule = @( "\*LanguageFeatures-Handwriting\*"; )

}

@{

Name = "TextToSpeech"

Description = "Text-to-speech"

Rule = @( "\*LanguageFeatures-TextToSpeech\*"; )

}

@{

Name = "Speech"

Description = "Speech recognition"

Rule = @( "\*LanguageFeatures-Speech\*"; )

}

@{

Name = "RegionSpecific"

Description = "Other region-specific requirements"

Rule = @( "\*InternationalFeatures\*"; )

}

@{

Name = "Retail"

Description = "Retail demo experience"

Rule = @( "\*RetailDemo\*"; )

}

@{

Name = "Features\_On\_Demand"

Description = "Features on demand"

Rule = @(

"\*InternetExplorer\*~x86~\*"; "\*InternetExplorer\*~x64~\*"; "\*InternetExplorer\*~amd64~\*"; "\*InternetExplorer\*~wow64~\*"; "\*InternetExplorer\*~arm64.x86~\*"; "\*InternetExplorer\*~arm64~\*";

"\*MSPaint\*~x86~\*"; "\*MSPaint\*~x64~\*"; "\*MSPaint\*~amd64~\*"; "\*MSPaint\*~wow64~\*"; "\*MSPaint\*~arm64.x86~\*"; "\*MSPaint\*~arm64~\*";

"\*Notepad-FoD-Package\*~x86~\*"; "\*Notepad-FoD-Package\*~x64~\*"; "\*Notepad-FoD-Package\*~amd64~\*"; "\*Notepad-FoD-Package\*~wow64~\*"; "\*Notepad-FoD-Package\*~arm64.x86~\*"; "\*Notepad-FoD-Package\*~arm64~\*";

"\*Notepad-System-FoD-Package\*~x86~\*"; "\*Notepad-System-FoD-Package\*~x64~\*"; "\*Notepad-System-FoD-Package\*~amd64~\*"; "\*Notepad-System-FoD-Package\*~wow64~\*"; "\*Notepad-System-FoD-Package\*~arm64.x86~\*"; "\*Notepad-System-FoD-Package\*~arm64~\*";

"\*MediaPlayer\*~x86~\*"; "\*MediaPlayer\*~x64~\*"; "\*MediaPlayer\*~amd64~\*"; "\*MediaPlayer\*~wow64~\*"; "\*MediaPlayer\*~arm64.x86~\*"; "\*MediaPlayer\*~arm64~\*";

"\*PowerShell\*ISE\*~x86~\*"; "\*PowerShell\*ISE\*~x64~\*"; "\*PowerShell\*ISE\*~amd64~\*"; "\*PowerShell\*ISE\*~wow64~\*"; "\*PowerShell\*ISE\*~arm64.x86~\*"; "\*PowerShell\*ISE\*~arm64~\*";

"\*StepsRecorder\*~x86~\*"; "\*StepsRecorder\*~x64~\*"; "\*StepsRecorder\*~amd64~\*"; "\*StepsRecorder\*~wow64~\*"; "\*StepsRecorder\*~arm64.x86~\*"; "\*StepsRecorder\*~arm64~\*";

"\*SnippingTool\*~x86~\*"; "\*SnippingTool\*~x64~\*"; "\*SnippingTool\*~amd64~\*"; "\*SnippingTool\*~wow64~\*"; "\*SnippingTool\*~arm64.x86~\*"; "\*SnippingTool\*~arm64~\*";

"\*WMIC\*~x86~\*"; "\*WMIC\*~x64~\*"; "\*WMIC\*~amd64~\*"; "\*WMIC\*~wow64~\*"; "\*WMIC\*~arm64.x86~\*"; "\*WMIC\*~arm64~\*";

"\*WordPad\*~x86~\*"; "\*WordPad\*~x64~\*"; "\*WordPad\*~amd64~\*"; "\*WordPad\*~wow64~\*"; "\*WordPad\*~arm64.x86~\*"; "\*WordPad\*~arm64~\*";

"\*Printing-WFS\*~x86~\*"; "\*Printing-WFS\*~x64~\*"; "\*Printing-WFS\*~amd64~\*"; "\*Printing-WFS\*~wow64~\*"; "\*Printing-WFS\*~arm64.x86~\*"; "\*Printing-WFS\*~arm64~\*";

"\*Printing-PMCPPC\*~x86~\*"; "\*Printing-PMCPPC\*~x64~\*"; "\*Printing-PMCPPC\*~amd64~\*"; "\*Printing-PMCPPC\*~wow64~\*"; "\*Printing-PMCPPC\*~arm64.x86~\*"; "\*Printing-PMCPPC\*~arm64~\*";

"\*Telnet-Client\*~x86~\*"; "\*Telnet-Client\*~x64~\*"; "\*Telnet-Client\*~amd64~\*"; "\*Telnet-Client\*~wow64~\*"; "\*Telnet-Client\*~arm64.x86~\*"; "\*Telnet-Client\*~arm64~\*";

"\*TFTP-Client\*~x86~\*"; "\*TFTP-Client\*~x64~\*"; "\*TFTP-Client\*~amd64~\*"; "\*TFTP-Client\*~wow64~\*"; "\*TFTP-Client\*~arm64.x86~\*"; "\*TFTP-Client\*~arm64~\*";

"\*VBSCRIPT\*~x86~\*"; "\*VBSCRIPT\*~x64~\*"; "\*VBSCRIPT\*~amd64~\*"; "\*VBSCRIPT\*~wow64~\*"; "\*VBSCRIPT\*~arm64.x86~\*"; "\*VBSCRIPT\*~arm64~\*";

"\*WinOcr-FOD-Package\*~x86~\*"; "\*WinOcr-FOD-Package\*~x64~\*"; "\*WinOcr-FOD-Package\*~amd64~\*"; "\*WinOcr-FOD-Package\*~wow64~\*"; "\*WinOcr-FOD-Package\*~arm64.x86~\*"; "\*WinOcr-FOD-Package\*~arm64~\*";

"\*ProjFS-OptionalFeature-FOD-Package\*~x86~\*"; "\*ProjFS-OptionalFeature-FOD-Package\*~x64~\*"; "\*ProjFS-OptionalFeature-FOD-Package\*~amd64~\*"; "\*ProjFS-OptionalFeature-FOD-Package\*~wow64~\*"; "\*ProjFS-OptionalFeature-FOD-Package\*~arm64.x86~\*"; "\*ProjFS-OptionalFeature-FOD-Package\*~arm64~\*";

"\*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package\*~x86~\*"; "\*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package\*~x64~\*"; "\*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package\*~amd64~\*"; "\*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package\*~wow64~\*"; "\*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package\*~arm64.x86~\*"; "\*ServerCoreFonts-NonCritical-Fonts-BitmapFonts-FOD-Package\*~arm64~\*";

"\*ServerCoreFonts-NonCritical-Fonts-Support-FOD-Package\*~x86~\*"; "\*ServerCoreFonts-NonCritical-Fonts-Support-FOD-Package\*~x64~\*"; "\*ServerCoreFonts-NonCritical-Fonts-Support-FOD-Package\*~amd64~\*"; "\*ServerCoreFonts-NonCritical-Fonts-Support-FOD-Package\*~wow64~\*"; "\*ServerCoreFonts-NonCritical-Fonts-Support-FOD-Package\*~arm64.x86~\*"; "\*ServerCoreFonts-NonCritical-Fonts-Support-FOD-Package\*~arm64~\*";

"\*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package\*~x86~\*"; "\*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package\*~x64~\*"; "\*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package\*~amd64~\*"; "\*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package\*~wow64~\*"; "\*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package\*~arm64.x86~\*"; "\*ServerCoreFonts-NonCritical-Fonts-TrueType-FOD-Package\*~arm64~\*";

"\*SimpleTCP-FOD-Package\*~x86~\*"; "\*SimpleTCP-FOD-Package\*~x64~\*"; "\*SimpleTCP-FOD-Package\*~amd64~\*"; "\*SimpleTCP-FOD-Package\*~wow64~\*"; "\*SimpleTCP-FOD-Package\*~arm64.x86~\*"; "\*SimpleTCP-FOD-Package\*~arm64~\*";

"\*VirtualMachinePlatform-Client-Disabled-FOD-Package\*~x86~\*"; "\*VirtualMachinePlatform-Client-Disabled-FOD-Package\*~x64~\*"; "\*VirtualMachinePlatform-Client-Disabled-FOD-Package\*~amd64~\*"; "\*VirtualMachinePlatform-Client-Disabled-FOD-Package\*~wow64~\*"; "\*VirtualMachinePlatform-Client-Disabled-FOD-Package\*~arm64.x86~\*"; "\*VirtualMachinePlatform-Client-Disabled-FOD-Package\*~arm64~\*";

"\*DirectoryServices-ADAM-Client-FOD-Package\*~x86~\*"; "\*DirectoryServices-ADAM-Client-FOD-Package\*~x64~\*"; "\*DirectoryServices-ADAM-Client-FOD-Package\*~amd64~\*"; "\*DirectoryServices-ADAM-Client-FOD-Package\*~wow64~\*"; "\*DirectoryServices-ADAM-Client-FOD-Package\*~arm64.x86~\*"; "\*DirectoryServices-ADAM-Client-FOD-Package\*~arm64~\*";

"\*EnterpriseClientSync-Host-FOD-Package\*~x86~\*"; "\*EnterpriseClientSync-Host-FOD-Package\*~x64~\*"; "\*EnterpriseClientSync-Host-FOD-Package\*~amd64~\*"; "\*EnterpriseClientSync-Host-FOD-Package\*~wow64~\*"; "\*EnterpriseClientSync-Host-FOD-Package\*~arm64.x86~\*"; "\*EnterpriseClientSync-Host-FOD-Package\*~arm64~\*";

"\*SenseClient-FoD-Package\*~x86~\*"; "\*SenseClient-FoD-Package\*~x64~\*"; "\*SenseClient-FoD-Package\*~amd64~\*"; "\*SenseClient-FoD-Package\*~wow64~\*"; "\*SenseClient-FoD-Package\*~arm64.x86~\*"; "\*SenseClient-FoD-Package\*~arm64~\*";

"\*SmbDirect-FOD-Package\*~x86~\*"; "\*SmbDirect-FOD-Package\*~x64~\*"; "\*SmbDirect-FOD-Package\*~amd64~\*"; "\*SmbDirect-FOD-Package\*~wow64~\*"; "\*SmbDirect-FOD-Package\*~arm64.x86~\*"; "\*SmbDirect-FOD-Package\*~arm64~\*";

"\*TerminalServices-AppServer-Client-FOD-Package\*~x86~\*"; "\*TerminalServices-AppServer-Client-FOD-Package\*~x64~\*"; "\*TerminalServices-AppServer-Client-FOD-Package\*~amd64~\*"; "\*TerminalServices-AppServer-Client-FOD-Package\*~wow64~\*"; "\*TerminalServices-AppServer-Client-FOD-Package\*~arm64.x86~\*"; "\*TerminalServices-AppServer-Client-FOD-Package\*~arm64~\*";

)

}

)

ForEach ($item in $Script:Search\_File\_Order) { New-Variable -Name "Init\_File\_Type\_$($item.Name)" -Value @() -Force }

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

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

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

ForEach ($NewRule in $item.Rule) {

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

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

Write-host " File name: " -NoNewline; Write-host $WildCard -ForegroundColor Green

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

$TempSave = @{ Match\_Name = $NewRule; FileName = $WildCard }

$new = $OSDefaultUser + $TempSave

if ($item.name -Contains $ExcludeNameItem) {

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

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

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

} else {

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

if ($Component -like "\*$($NewRule)\*") {

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

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

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

break

}

}

}

}

}

}

}

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

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

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

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

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

if ($OSDefaultUser.Count -gt 0) {

ForEach ($item in $OSDefaultUser) { Write-host " $($item.FileName)" -ForegroundColor Green }

} else { Write-host " Not available" -ForegroundColor Red }

}

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

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

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

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

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

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

}

}

Write-host "`n Install" -ForegroundColor Yellow

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

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

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

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

if ($OSDefaultUser.Count -gt 0) {

ForEach ($item in $OSDefaultUser) {

Write-host " File name: " -NoNewline; Write-host $item.FileName -ForegroundColor Green

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

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

try {

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

Write-host "Finish`n" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

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

}

} else { Write-host "Does not exist`n" }

}

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

}

}

Language\_Install -Mount "D:\OS\_11\_Custom\Install\Install\Mount" -Sources "D:\OS\_11\_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

Region: zh-CN

Dism /Image:"D:\OS\_11\_Custom\Install\Install\Mount" /Set-AllIntl:zh-CN

* + 1. View available language settings

Dism /Image:"D:\OS\_11\_Custom\Install\Install\Mount" /Get-Intl

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

* + 1. View

Get-WindowsPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" | Out-GridView

* + 1. Export to csv

$SaveTo = "D:\OS\_11\_Custom\Install\Install\Report.Components.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green

##### InBox Apps

###### Learn

* + 1. [DISM App Package (.appx or .appxbundle) Servicing Command-Line Options](https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/dism-app-package--appx-or-appxbundle--servicing-command-line-options)

Important:

DISM /add-provisionedappxpackage Starting in Windows 10 version 1803, when using this option, you can preinstall apps without pinning them to the Start menu /region. When preinstalling an app, you can choose to exclude the app from LayoutModification.xml so that the app will install successfully and not appear as a Start menu tile. If no region list is specified, the app will only be provisioned when it is pinned to the Start layout.

Again, you are warned that using the command line to add must add the parameter: /Region

###### InBox Apps: Pre-install the application

* Group abbreviations
* Group A: Core, CoreSingleLanguage
* Group B: CoreN
* Group C: Education, Professional, ProfessionalEducation, ProfessionalWorkstation, Enterprise, IoTEnterprise, IoTEnterpriseK, ServerRdsh, ServerStandard, ServerDatacenter
* Group D: EducationN, ProfessionalN, ProfessionalEducationN, ProfessionalWorkstationN, EnterpriseN, EnterpriseGN, EnterpriseSN, CloudN, CloudEN, CloudEditionLN, StarterN

Name Products ID Grouping

1. Microsoft.AV1VideoExtension 9MVZQVXJBQ9V A, C
2. Microsoft.AVCEncoderVideoExtension 9PB0TRCNRHFX A, C
3. Microsoft.GamingApp 9MV0B5HZVK9Z A, C
4. Microsoft.HEIFImageExtension 9PMMSR1CGPWG A, C
5. Microsoft.HEVCVideoExtension 9NMZLZ57R3T7 A, C
6. Microsoft.RawImageExtension 9NCTDW2W1BH8 A, C
7. Microsoft.VP9VideoExtensions 9N4D0MSMP0PT A, C
8. Microsoft.WebMediaExtensions 9N5TDP8VCMHS A, C
9. Microsoft.WebpImageExtension 9PG2DK419DRG A, C
10. Microsoft.MPEG2VideoExtension 9N95Q1ZZPMH4 A, C
11. Microsoft.Windows.DevHome 9N8MHTPHNGVV A, C
12. Microsoft.WindowsSoundRecorder 9WZDNCRFHWKN A, C
13. Microsoft.Xbox.TCUI 9NKNC0LD5NN6 A, C
14. Microsoft.XboxGamingOverlay 9NZKPSTSNW4P A, C
15. Microsoft.ZuneMusic 9WZDNCRFJ3PT A, C
16. Microsoft.CrossDevice 9NTXGKQ8P7N0 A, C, D
17. MSTeams XP8BT8DW290MPQ A, C, D
18. MicrosoftCorporationII.MicrosoftFamily 9PDJDJS743XF A, B
19. Microsoft.UI.Xaml.2.8 A, B, C, D
20. Microsoft.WindowsAppRuntime.1.5 A, B, C, D
21. Microsoft.WindowsAppRuntime.1.6 A, B, C, D
22. Microsoft.WindowsAppRuntime.1.7 A, B, C, D
23. Microsoft.NET.Native.Framework.2.2 A, B, C, D
24. Microsoft.NET.Native.Runtime.2.2 A, B, C, D
25. Microsoft.VCLibs.140.00 A, B, C, D
26. Microsoft.VCLibs.140.00.UWPDesktop A, B, C, D
27. Microsoft.Services.Store.Engagement A, B, C, D
28. Microsoft.DesktopAppInstaller 9NBLGGH4NNS1 A, B, C, D
29. Microsoft.WindowsStore 9WZDNCRFJBMP A, B, C, D
30. MicrosoftWindows.Client.WebExperience 9MSSGKG348SP A, B, C, D
31. Clipchamp.Clipchamp 9P1J8S7CCWWT A, B, C, D
32. ApplicationCompatibilityEnhancements 9PCSD6N03BKV A, B, C, D
33. Microsoft.BingNews 9WZDNCRFHVFW A, B, C, D
34. Microsoft.BingSearch 9NZBF4GT040C A, B, C, D
35. Microsoft.BingWeather 9WZDNCRFJ3Q2 A, B, C, D
36. Microsoft.Copilot 9NHT9RB2F4HD A, B, C, D
37. Microsoft.GetHelp 9PKDZBMV1H3T A, B, C, D
38. Microsoft.MicrosoftOfficeHub 9WZDNCRD29V9 A, B, C, D
39. Microsoft.MicrosoftSolitaireCollection 9WZDNCRFHWD2 A, B, C, D
40. Microsoft.StickyNotes 9NBLGGH4QGHW A, B, C, D
41. Microsoft.OutlookForWindows 9NRX63209R7B A, B, C, D
42. Microsoft.Paint 9PCFS5B6T72H A, B, C, D
43. Microsoft.PowerAutomateDesktop 9NFTCH6J7FHV A, B, C, D
44. Microsoft.ScreenSketch 9MZ95KL8MR0L A, B, C, D
45. Microsoft.SecHealthUI 9NK0JLZT7JD0 A, B, C, D
46. Microsoft.StorePurchaseApp 9NBLGGH4LS1F A, B, C, D
47. Microsoft.Todos 9NBLGGH5R558 A, B, C, D
48. Microsoft.Windows.Photos 9WZDNCRFJBH4 A, B, C, D
49. Microsoft.WindowsAlarms 9WZDNCRFJ3PR A, B, C, D
50. Microsoft.WindowsCalculator 9WZDNCRFHVN5 A, B, C, D
51. Microsoft.WindowsCamera 9WZDNCRFJBBG A, B, C, D
52. Microsoft.WindowsFeedbackHub 9NBLGGH4R32N A, B, C, D
53. Microsoft.WindowsNotepad 9MSMLRH6LZF3 A, B, C, D
54. Microsoft.WindowsTerminal 9N0DX20HK701 A, B, C, D
55. Microsoft.XboxIdentityProvider 9WZDNCRD1HKW A, B, C, D
56. Microsoft.XboxSpeechToTextOverlay 9P086NHDNB9W A, B, C, D
57. Microsoft.YourPhone 9NMPJ99VJBWV A, B, C, D
58. MicrosoftCorporationII.QuickAssist 9P7BP5VNWKX5 A, B, C, D
59. MicrosoftWindows.Client.WebExperience 9MSSGKG348SP A, B, C

###### InBox Apps: Installed

$custom\_array = @(); Get-AppXProvisionedPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" | ForEach-Object { $custom\_array += [PSCustomObject]@{ DisplayName = $\_.DisplayName; PackageName = $\_.PackageName; Version = $\_.Version; Architecture = $\_.Architecture; ResourceId = $\_.ResourceId; Regions = $\_.Regions; } };

* After executing the above PS code first, you can select the following "View" or "Export to CSV".
* Check the installed InBox Apps and understand the region binding. Unbound regions will not appear in the application after executing OOBE.
  + 1. View

$custom\_array | Out-GridView

* + 1. Export to Csv

$SaveTo = "D:\OS\_11\_Custom\Install\Install\Report.InBox.Apps.$(Get-Date -Format "yyyyMMddHHmmss").csv"

$custom\_array | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green

###### Remove all installed pre-applications

* Install.Inbox.apps.Clear.all.ps1
  + [\Expand\Install\Install.Inbox.apps.Clear.all.ps1](Expand/Install/Install.InBox.Apps.Clear.all.ps1)
  + [https://github.com/ilikeyi/solutions/blob/main/\_Learn/Packaging.tutorial/OS.11/25H2/Expand/Install/Install.InBox.Apps.Clear.all.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/Install.InBox.Apps.Clear.all.ps1)
* Copy the code

Get-AppXProvisionedPackage -path "D:\OS\_11\_Custom\Install\Install\Mount" -ErrorAction SilentlyContinue | ForEach-Object {

Write-host "`n $($\_.DisplayName)"; Write-Host " Deleting ".PadRight(22) -NoNewline

try {

Remove-AppxProvisionedPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" -PackageName $\_.PackageName -ErrorAction SilentlyContinue | Out-Null

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

}

}

###### Installing "Local Language Experience Packs (LXPs)" to offline images marks: installed languages

Before you learn about installed language tags, first understand the list of languages available in the ZuneMusic app: en-us, en-gb, as-in, az-latn-az, bs-latn-ba, bn-in, cs-cz, cy-gb, ar-sa, de-de, af-za, am-et, da-dk, el-gr, es-es, es-mx, ca-es, ca-es-valencia, eu-es, fi-fi, bg-bg, ga-ie, et-ee, gl-es, gu-in, gd-gb, fil-ph, fr-ca, fr-fr, hy-am, fa-ir, hu-hu, id-id, it-it, ja-jp, hr-hr, ka-ge, is-is, he-il, hi-in, kk-kz, lo-la, km-kh, lt-lt, kn-in, kok-in, ko-kr, mk-mk, lv-lv, ms-my, mi-nz, mr-in, mt-mt, ne-np, nl-nl, nb-no, ml-in, nn-no, lb-lu, or-in, ro-ro, sk-sk, ru-ru, pl-pl, quz-pe, pt-br, pt-pt, sq-al, ta-in, th-th, sv-se, pa-in, sr-cyrl-ba, sr-cyrl-rs, sr-latn-rs, ur-pk, ug-cn, tr-tr, uk-ua, tt-ru, zh-cn, vi-vn, uz-latn-uz, sl-si, zh-tw, te-in

View installed languages for offline images

Dism /Image:"D:\OS\_11\_Custom\Install\Install\Mount" /Get-Intl

View results Installed languages: en-US, zh-CN. When installing the ZuneMusic app, the installer will match "Installed Languages" and automatically filter the list of uninstalled languages. This is the language tag.

###### Offline image has language tag installed: How to add it

To mark the offline image: Language has been installed, there are two methods. The first is to install Microsoft-Windows-Client-Language-Pack\_x64\_zh-cn.cab, and the second is to install "Local Language Experience Pack (LXPs)". Microsoft official Windows 10 provides Local Language Experience Pack (LXPS) installation files, but Windows 11 no longer provides them. If you want to get them:

* + 1. Download using the Windows Local Language Experience Packs (LXPs) Downloader

learn: <https://github.com/ilikeyi/LXPs>

After downloading, save to: D:\OS\_11\_Custom\Install\Install\Inbox.apps

File format: LanguageExperiencePack.zh-CN.Neutral.Appx

* + 1. Manual download
       1. Region

Region Products ID Region Products ID

en-US C9PDSCC711RVF af-za 9PDW16B5HMXR

am-et 9NGL4R61W3PL ar-sa 9N4S78P86PKX

as-in 9NTJLXMXX35J az-latn-az 9P5TFKZHQ5K8

be-by 9MXPBGNNDW3L bg-bg 9MX54588434F

bn-bd 9PH7TKVXGGM8 bn-in 9P1M44L7W84T

bs-latn-ba 9MVFKLJ10MFL ca-es 9P6JMKJQZ9S7

ca-es-valencia 9P9K3WMFSW90 chr-cher-us 9MX15485N3RK

cs-cz 9P3WXZ1KTM7C cy-gb 9NKJ9TBML4HB

da-dk 9NDMT2VKSNL1 de-de 9P6CT0SLW589

el-gr 9N586B13PBLD en-gb 9NT52VQ39BVN

es-es 9NWVGWLHPB1Z es-mx 9N8MCM1X3928

et-ee 9NFBHFMCR30L eu-es 9NMCHQHZ37HZ

fa-ir 9NGS7DD4QS21 fi-fi 9MW3PQ7SD3QK

fil-ph 9NWM2KGTDSSS fr-ca 9MTP2VP0VL92

fr-fr 9NHMG4BJKMDG ga-ie 9P0L5Q848KXT

gd-gb 9P1DBPF36BF3 gl-es 9NXRNBRNJN9B

gu-in 9P2HMSWDJDQ1 ha-latn-ng 9N1L95DBGRG3

he-il 9NB6ZFND5HCQ hi-in 9NZC3GRX8LD3

hr-hr 9NW01VND4LTW hu-hu 9MWN3C58HL87

hy-am 9NKM28TM6P67 id-id 9P4X3N4SDK8P

ig-ng 9PG4ZFJ48JSX is-is 9NTHJR7TQXX1

it-it 9P8PQWNS6VJX ja-jp 9N1W692FV4S1

ka-ge 9P60JZL05WGH kk-kz 9PHV179R97LV

km-kh 9PGKTS4JS531 kn-in 9NC6DB7N95F9

kok-in 9MV3P55CMZ6P ko-kr 9N4TXPCVRNGF

ku-arab-iq 9P1C18QL3D7H ky-kg 9P7D3JJGZM48

lb-lu 9N0ST1WBZ9D9 lo-la 9N8X352G5NZV

lt-lt 9NWWD891H6HN lv-lv 9N5CQDPH6SQT

mi-nz 9P2GDFB3JPSX mk-mk 9P1X6XB1K3RN

ml-in 9NWDTV8FFV7L mn-mn 9PG1DHC4VTZW

mr-in 9MWXCKHJVR1J ms-my 9NPXL8ZSDDQ7

mt-mt 9PDG96SQ6BN8 nb-no 9N6J0M5DHCK0

ne-np 9P7CHPLWDQVN nl-nl 9PF1C9NB5PRV

nn-no 9PK7KM3Z06KH nso-za 9NS49QLX5CDV

or-in 9NTHCXCXSJDH pa-arab-pk 9NJRL03WH6FM

pa-in 9NSNC0ZJX69B pl-pl 9NC5HW94R0LD

prs-af 9P3NGC6X5ZQC pt-br 9P8LBDM4FW35

pt-pt 9P7X8QJ7FL0X quc-latn-gt 9P2V6MNNQZ0B

quz-pe 9NHTX8NVQ04K ro-ro 9MWXGPJ5PJ3H

ru-ru 9NMJCX77QKPX rw-rw 9NFW0M20H9WG

sd-arab-pk 9NB9JSCXW9X5 si-lk 9NVF9QSLGTL0

sk-sk 9N7LSNN099WB sl-si 9NV27L34J4ST

sq-al 9MWLRGNMDGK7 sr-cyrl-ba 9MXGN7V65C7B

sr-cyrl-rs 9PPD6CCK9K5H sr-latn-rs 9NBZ0SJDPPVT

sv-se 9P0HSNX08177 sw-ke 9NFF2M19DQ55

ta-in 9PDZB1WT1B34 te-in 9PMQJJGF63FW

tg-cyrl-tj 9MZHLBPPT2HC th-th 9MSTWFRL0LR4

ti-et 9NC8C9RDNK2S tk-tm 9NKHQ4GL6VLT

tn-za 9NFSXM123DHT tr-tr 9NL1D3T5HG9R

tt-ru 9NV90Q1X1ZR2 ug-cn 9P52C5D7VL5S

uk-ua 9PPPMZRSGHR8 ur-pk 9NDWFTFW12BQ

uz-latn-uz 9P5P2T5P5L9S vi-vn 9P0W68X0XZPT

wo-sn 9NH3SW1CR90F xh-za 9NW3QWSLQD17

yo-ng 9NGM3VPPZS5V zh-cn 9NRMNT6GMZ7

zh-tw 9PCJ4DHCQ1JQ zu-za 9NNRM7KT5NB0

Download Region: zh-CN, application ID: 9NRMNT6GMZ70, Store link: https://www.microsoft.com/store/productId/9NRMNT6GMZ70

* + - 1. Open the website: <https://store.rg-adguard.net>
         1. a). Select a URL (link). b). Select a channel (RP). c). Enter keywords: https://www.microsoft.com/store/productId/9NRMNT6GMZ70
         2. Search 26200 content in the web page, search results: Microsoft.LanguageExperiencePackzh-CN\_26200.\*. neutral\_\_8wekyb3d8bbwe.appx
         3. After downloading, save it to the

D:\OS\_11\_Custom\Install\Install\Inbox.apps directory and rename it: LanguageExperiencePack.zh-cn.Neutral.Appx

* + 1. Execute the installation command to install the local language experience package (LXPs)

After understanding how to add zone tags, obtain LanguageExperiencePack.zh-cn.Neutral.appx, execute the installation command:

Add-AppxProvisionedPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" -PackagePath "D:\OS\_11\_Custom\Install\Install\Inbox.apps\LanguageExperiencePack.zh-cn.Neutral.appx" -SkipLicense

* + 1. Regional tag: New changes

The installed language of the offline image will become the default. For example, if the installed languages ​​are en-US and zh-CN, when you reinstall or update the InBox Apps application, the installed language will be referenced for matching and the corresponding language pack will be automatically added. You can view the changes in the following ways.

* + - 1. InBox Apps: An installed application package

$custom\_array = @(); Get-AppXProvisionedPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" | ForEach-Object { $custom\_array += [PSCustomObject]@{ DisplayName = $\_.DisplayName; PackageName = $\_.PackageName; Version = $\_.Version; Architecture = $\_.Architecture; ResourceId = $\_.ResourceId; Regions = $\_.Regions; } };

* After executing the above PS code first, you can select the following "View" or "Export to CSV".
  + - * 1. View

$custom\_array | Out-GridView

* + - * 1. Export to Csv

$SaveTo = "D:\OS\_10\_Custom\Install\Install\Report.InBox.Apps.$(Get-Date -Format "yyyyMMddHHmmss").csv"

$custom\_array | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green

* + - 1. View available language settings

Dism /Image:"D:\OS\_10\_Custom\Install\Install\Mount" /Get-Intl

###### InBox Apps: Install

* + 1. Mount or decompress the InBox Apps installation file

Mount 26100.6584.250904-1728.ge\_release\_svc\_prod1\_amd64fre\_InboxApps.iso or extract to any location;

* + 1. After executing the installation command, install InBox Apps to: Install.wim
* Auto = Automatically search all local disks, default;
* Custom path, e.g. specify F drive: $ISO = "F:\packages"
* Architecture: x64
* Install.Inst.Inbox.apps.ps1
  + [\Expand\Install\Install.Inst.Inbox.apps.ps1](Expand/Install/Install.Inst.InBox.Apps.ps1)
  + [https://github.com/ilikeyi/solutions/blob/main/\_Learn/Packaging.tutorial/OS.11/25H2/Expand/Install/Install.Inst.InBox.Apps.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/Install.Inst.InBox.Apps.ps1)
* Copy the code

$ISO = "Auto"

$Mount = "D:\OS\_11\_Custom\Install\Install\Mount"

$Arch = "x64"

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 = @{

Edition = @(

@{

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

Apps = @(

"Microsoft.UI.Xaml.2.8"; "Microsoft.WindowsAppRuntime.1.5"; "Microsoft.WindowsAppRuntime.1.6"; "Microsoft.WindowsAppRuntime.1.7"; "Microsoft.NET.Native.Framework.2.2"; "Microsoft.NET.Native.Runtime.2.2"; "Microsoft.VCLibs.140.00"; "Microsoft.VCLibs.140.00.UWPDesktop"; "Microsoft.Services.Store.Engagement";

"Clipchamp.Clipchamp"; "Microsoft.ApplicationCompatibilityEnhancements"; "Microsoft.AV1VideoExtension"; "Microsoft.AVCEncoderVideoExtension"; "Microsoft.BingNews"; "Microsoft.BingSearch"; "Microsoft.BingWeather"; "Microsoft.Copilot"; "Microsoft.DesktopAppInstaller";

"Microsoft.GamingApp"; "Microsoft.GetHelp"; "Microsoft.HEIFImageExtension"; "Microsoft.HEVCVideoExtension"; "Microsoft.MicrosoftOfficeHub"; "Microsoft.MicrosoftSolitaireCollection"; "Microsoft.MicrosoftStickyNotes"; "Microsoft.MPEG2VideoExtension";

"Microsoft.OutlookForWindows"; "Microsoft.Paint"; "Microsoft.PowerAutomateDesktop"; "Microsoft.RawImageExtension"; "Microsoft.ScreenSketch"; "Microsoft.SecHealthUI"; "Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.VP9VideoExtensions"; "Microsoft.WebMediaExtensions";

"Microsoft.WebpImageExtension"; "Microsoft.Windows.DevHome"; "Microsoft.Windows.Photos"; "Microsoft.WindowsAlarms"; "Microsoft.WindowsCalculator"; "Microsoft.WindowsCamera"; "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsNotepad"; "Microsoft.WindowsSoundRecorder";

"Microsoft.WindowsStore"; "Microsoft.WindowsTerminal"; "Microsoft.Xbox.TCUI"; "Microsoft.XboxGamingOverlay"; "Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay"; "Microsoft.YourPhone"; "Microsoft.ZuneMusic"; "MicrosoftCorporationII.MicrosoftFamily";

"MicrosoftCorporationII.QuickAssist"; "MicrosoftWindows.Client.WebExperience"; "Microsoft.CrossDevice"; "MSTeams";

)

}

@{

Name = @( "CoreN" )

Apps = @(

"Microsoft.UI.Xaml.2.8"; "Microsoft.WindowsAppRuntime.1.3"; "Microsoft.WindowsAppRuntime.1.4"; "Microsoft.NET.Native.Framework.2.2"; "Microsoft.NET.Native.Runtime.2.2"; "Microsoft.VCLibs.140.00"; "Microsoft.VCLibs.140.00.UWPDesktop"; "Microsoft.Services.Store.Engagement";

"Clipchamp.Clipchamp"; "Microsoft.ApplicationCompatibilityEnhancements"; "Microsoft.BingNews"; "Microsoft.BingSearch"; "Microsoft.BingWeather"; "Microsoft.Copilot"; "Microsoft.DesktopAppInstaller"; "Microsoft.GetHelp"; "Microsoft.MicrosoftOfficeHub"; "Microsoft.MicrosoftSolitaireCollection";

"Microsoft.MicrosoftStickyNotes"; "Microsoft.OutlookForWindows"; "Microsoft.Paint"; "Microsoft.PowerAutomateDesktop"; "Microsoft.ScreenSketch"; "Microsoft.SecHealthUI"; "Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.Windows.Photos"; "Microsoft.WindowsAlarms";

"Microsoft.WindowsCalculator"; "Microsoft.WindowsCamera"; "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsNotepad"; "Microsoft.WindowsStore"; "Microsoft.WindowsTerminal"; "Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay"; "Microsoft.YourPhone";

"MicrosoftCorporationII.MicrosoftFamily"; "MicrosoftCorporationII.QuickAssist"; "MicrosoftWindows.Client.WebExperience"; "Microsoft.CrossDevice"; "MSTeams";

)

}

@{

Name = @( "Education"; "Professional"; "ProfessionalEducation"; "ProfessionalWorkstation"; "Enterprise"; "IoTEnterprise"; "IoTEnterpriseK"; "ServerRdsh"; "ServerStandard"; "ServerDatacenter"; )

Apps = @(

"Microsoft.UI.Xaml.2.8"; "Microsoft.WindowsAppRuntime.1.3"; "Microsoft.WindowsAppRuntime.1.4"; "Microsoft.NET.Native.Framework.2.2"; "Microsoft.NET.Native.Runtime.2.2"; "Microsoft.VCLibs.140.00"; "Microsoft.VCLibs.140.00.UWPDesktop"; "Microsoft.Services.Store.Engagement";

"Clipchamp.Clipchamp"; "Microsoft.ApplicationCompatibilityEnhancements"; "Microsoft.AV1VideoExtension"; "Microsoft.AVCEncoderVideoExtension"; "Microsoft.BingNews"; "Microsoft.BingSearch"; "Microsoft.BingWeather"; "Microsoft.Copilot"; "Microsoft.DesktopAppInstaller"; "Microsoft.GamingApp";

"Microsoft.GetHelp"; "Microsoft.HEIFImageExtension"; "Microsoft.HEVCVideoExtension"; "Microsoft.MicrosoftOfficeHub"; "Microsoft.MicrosoftSolitaireCollection"; "Microsoft.MicrosoftStickyNotes"; "Microsoft.MPEG2VideoExtension"; "Microsoft.OutlookForWindows"; "Microsoft.Paint";

"Microsoft.PowerAutomateDesktop"; "Microsoft.RawImageExtension"; "Microsoft.ScreenSketch"; "Microsoft.SecHealthUI"; "Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.VP9VideoExtensions"; "Microsoft.WebMediaExtensions"; "Microsoft.WebpImageExtension";

"Microsoft.Windows.DevHome"; "Microsoft.Windows.Photos"; "Microsoft.WindowsAlarms"; "Microsoft.WindowsCalculator"; "Microsoft.WindowsCamera"; "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsNotepad"; "Microsoft.WindowsSoundRecorder"; "Microsoft.WindowsStore";

"Microsoft.WindowsTerminal"; "Microsoft.Xbox.TCUI"; "Microsoft.XboxGamingOverlay"; "Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay"; "Microsoft.YourPhone"; "Microsoft.ZuneMusic"; "MicrosoftCorporationII.QuickAssist"; "MicrosoftWindows.Client.WebExperience";

"Microsoft.CrossDevice"; "MSTeams";

)

}

@{

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

Apps = @(

"Microsoft.UI.Xaml.2.8"; "Microsoft.WindowsAppRuntime.1.3"; "Microsoft.WindowsAppRuntime.1.4"; "Microsoft.NET.Native.Framework.2.2"; "Microsoft.NET.Native.Runtime.2.2"; "Microsoft.VCLibs.140.00"; "Microsoft.VCLibs.140.00.UWPDesktop"; "Microsoft.Services.Store.Engagement";

"Clipchamp.Clipchamp"; "Microsoft.ApplicationCompatibilityEnhancements"; "Microsoft.BingNews"; "Microsoft.BingSearch"; "Microsoft.BingWeather"; "Microsoft.Copilot"; "Microsoft.DesktopAppInstaller"; "Microsoft.GetHelp"; "Microsoft.MicrosoftOfficeHub"; "Microsoft.MicrosoftSolitaireCollection";

"Microsoft.MicrosoftStickyNotes"; "Microsoft.OutlookForWindows"; "Microsoft.Paint"; "Microsoft.PowerAutomateDesktop"; "Microsoft.ScreenSketch"; "Microsoft.SecHealthUI"; "Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.Windows.Photos"; "Microsoft.WindowsAlarms";

"Microsoft.WindowsCalculator"; "Microsoft.WindowsCamera"; "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsNotepad"; "Microsoft.WindowsStore"; "Microsoft.WindowsTerminal"; "Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay"; "Microsoft.YourPhone";

"MicrosoftCorporationII.QuickAssist"; "MicrosoftWindows.Client.WebExperience"; "Microsoft.CrossDevice"; "MSTeams";

            )

        }

@{

Name = @( "EnterpriseS"; "EnterpriseSN"; "IoTEnterpriseS"; "IoTEnterpriseSK"; )

Apps = @( "Microsoft.SecHealthUI"; )

        }

    )

    Rule = @(

@{ Name = "Microsoft.UI.Xaml.2.8"; Match = "UI.Xaml\*{ARCHTag}\*2.8"; License = "UI.Xaml\*{ARCHTag}\*2.8"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.WindowsAppRuntime.1.5"; Match = "WindowsAppRuntime.{ARCHC}.1.5"; License = "WindowsAppRuntime.{ARCHC}.1.5"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.WindowsAppRuntime.1.6"; Match = "WindowsAppRuntime.{ARCHC}.1.6"; License = "WindowsAppRuntime.{ARCHC}.1.6"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.WindowsAppRuntime.1.7"; Match = "WindowsAppRuntime.{ARCHC}.1.7"; License = "WindowsAppRuntime.{ARCHC}.1.7"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.NET.Native.Framework.2.2"; Match = "Native.Framework\*{ARCHTag}\*2.2"; License = "Native.Framework\*{ARCHTag}\*2.2"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.NET.Native.Runtime.2.2"; Match = "Native.Runtime\*{ARCHTag}\*2.2"; License = "Native.Runtime\*{ARCHTag}\*2.2"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.VCLibs.140.00"; Match = "Microsoft.VCLibs.{ARCHTag}.14.00.appx"; License = "VCLibs.{ARCHTag}.14.00.appx"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.VCLibs.140.00.UWPDesktop"; Match = "Microsoft.VCLibs.{ARCHTag}.14.00.UWPDesktop"; License = "Microsoft.VCLibs.{ARCHTag}.14.00.UWPDesktop"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.Services.Store.Engagement"; Match = "Services.Store.Engagement\*{ARCHC}"; License = "Services.Store.Engagement\*{ARCHC}"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.DesktopAppInstaller"; Match = "DesktopAppInstaller"; License = "DesktopAppInstaller"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.WindowsStore"; Match = "WindowsStore"; License = "WindowsStore"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "MicrosoftWindows.Client.WebExperience"; Match = "WebExperience"; License = "WebExperience"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00"); }

@{ Name = "Clipchamp.Clipchamp"; Match = "Clipchamp.Clipchamp"; License = "Clipchamp.Clipchamp"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.ApplicationCompatibilityEnhancements"; Match = "ApplicationCompatibilityEnhancements"; License = "ApplicationCompatibilityEnhancements"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.AV1VideoExtension"; Match = "AV1VideoExtension"; License = "AV1VideoExtension"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.AVCEncoderVideoExtension"; Match = "AVCEncoderVideoExtension"; License = "AVCEncoderVideoExtension"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.BingNews"; Match = "BingNews"; License = "BingNews"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.BingSearch"; Match = "BingSearch"; License = "BingSearch"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.BingWeather"; Match = "BingWeather"; License = "BingWeather"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.Copilot"; Match = "Copilot"; License = "Copilot"; Region = "AD;AE;AF;AG;AI;AL;AM;AO;AQ;AR;AS;AT;AU;AW;AX;AZ;BA;BB;BD;BE;BF;BG;BH;BI;BJ;BL;BM;BN;BO;BQ;BR;BS;BT;BV;BW;BZ;CA;CC;CD;CF;CG;CH;CI;CK;CL;CM;CO;CR;CV;CW;CX;CY;CZ;DE;DJ;DK;DM;DO;DZ;EC;EE;EG;ER;ES;ET;FI;FJ;FK;FM;FO;FR;GA;GB;GD;GE;GF;GG;GH;GI;GL;GM;GN;GP;GQ;GR;GS;GT;GU;GW;GY;HK;HM;HN;HR;HT;HU;ID;IE;IL;IM;IN;IO;IQ;IS;IT;JE;JM;JO;JP;KE;KG;KH;KI;KM;KN;KR;KW;KY;KZ;LA;LB;LC;LI;LK;LR;LS;LT;LU;LV;LY;MA;MC;MD;ME;MF;MG;MH;MK;ML;MM;MN;MO;MP;MQ;MR;MS;MT;MU;MV;MW;MX;MY;MZ;NA;NC;NE;NF;NG;NI;NL;NO;NP;NR;NU;NZ;OM;PA;PE;PF;PG;PH;PK;PL;PM;PN;PR;PS;PT;PW;PY;QA;RE;RO;RS;RW;SA;SB;SC;SD;SE;SG;SH;SI;SJ;SK;SL;SM;SN;SO;SR;SS;ST;SV;SX;SZ;TC;TD;TF;TG;TH;TJ;TK;TL;TM;TN;TO;TR;TT;TV;TW;TZ;UA;UG;UM;US;UY;UZ;VA;VC;VE;VG;VI;VN;VU;WF;WS;XK;YE;YT;ZA;ZM;ZW"; Dependencies = @("Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.GamingApp"; Match = "GamingApp"; License = "GamingApp"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.GetHelp"; Match = "GetHelp"; License = "GetHelp"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.HEIFImageExtension"; Match = "HEIFImageExtension"; License = "HEIFImageExtension\*"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.HEVCVideoExtension"; Match = "HEVCVideoExtension"; License = "15a6383d188a454ca98e15931ce48dc4\_License1"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.MicrosoftOfficeHub"; Match = "MicrosoftOfficeHub"; License = "MicrosoftOfficeHub"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.MicrosoftSolitaireCollection"; Match = "MicrosoftSolitaireCollection"; License = "MicrosoftSolitaireCollection"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.MicrosoftStickyNotes"; Match = "MicrosoftStickyNotes"; License = "MicrosoftStickyNotes"; Region = "All"; Dependencies = @("Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.MPEG2VideoExtension"; Match = "MPEG2VideoExtension"; License = "MPEG2VideoExtension"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.OutlookForWindows"; Match = "Microsoft.OutlookForWindows\_8wekyb3d8bbwe"; License = "Microsoft.OutlookForWindows\_8wekyb3d8bbwe"; Region = "AD;AE;AF;AG;AI;AL;AM;AO;AQ;AR;AS;AT;AU;AW;AX;AZ;BA;BB;BD;BE;BF;BG;BH;BI;BJ;BL;BM;BN;BO;BQ;BR;BS;BT;BV;BW;BY;BZ;CA;CC;CD;CF;CG;CH;CI;CK;CL;CM;CO;CR;CU;CV;CW;CX;CY;CZ;DE;DJ;DK;DM;DO;DZ;EC;EE;EG;ER;ES;ET;FI;FJ;FK;FM;FO;FR;GA;GB;GD;GE;GF;GG;GH;GI;GL;GM;GN;GP;GQ;GR;GS;GT;GU;GW;GY;HK;HM;HN;HR;HT;HU;ID;IE;IL;IM;IN;IO;IQ;IR;IS;IT;JE;JM;JO;JP;KE;KG;KH;KI;KM;KN;KP;KR;KW;KY;KZ;LA;LB;LC;LI;LK;LR;LS;LT;LU;LV;LY;MA;MC;MD;ME;MF;MG;MH;MK;ML;MM;MN;MO;MP;MQ;MR;MS;MT;MU;MV;MW;MX;MY;MZ;NA;NC;NE;NF;NG;NI;NL;NO;NP;NR;NU;NZ;OM;PA;PE;PF;PG;PH;PK;PL;PM;PN;PR;PS;PT;PW;PY;QA;RE;RO;RS;RU;RW;SA;SB;SC;SD;SE;SG;SH;SI;SJ;SK;SL;SM;SN;SO;SR;SS;ST;SV;SX;SY;SZ;TC;TD;TF;TG;TH;TJ;TK;TL;TM;TN;TO;TR;TT;TV;TW;TZ;UA;UG;UM;US;UY;UZ;VA;VC;VE;VG;VI;VN;VU;WF;WS;XK;YE;YT;ZA;ZM;ZW"; Dependencies = @(); }

@{ Name = "Microsoft.Paint"; Match = "Paint"; License = "Paint"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop", "Microsoft.UI.Xaml.2.8"); }

@{ Name = "Microsoft.PowerAutomateDesktop"; Match = "PowerAutomateDesktop"; License = "PowerAutomateDesktop"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.RawImageExtension"; Match = "RawImageExtension"; License = "RawImageExtension"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.ScreenSketch"; Match = "ScreenSketch"; License = "ScreenSketch"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.WindowsAppRuntime.1.3", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.SecHealthUI"; Match = "SecHealthUI\*{ARCHC}"; License = "SecHealthUI\*{ARCHC}"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.StorePurchaseApp"; Match = "StorePurchaseApp"; License = "StorePurchaseApp"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.Todos"; Match = "Todos"; License = "Todos"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop", "Microsoft.Services.Store.Engagement"); }

@{ Name = "Microsoft.VP9VideoExtensions"; Match = "VP9VideoExtensions"; License = "VP9VideoExtensions"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.WebMediaExtensions"; Match = "WebMediaExtensions"; License = "WebMediaExtensions"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.WebpImageExtension"; Match = "WebpImageExtension"; License = "WebpImageExtension"; Region = "All"; Dependencies = @(); }

@{ Name = "Microsoft.Windows.DevHome"; Match = "DevHome"; License = "DevHome"; Region = "All"; Dependencies = @("Microsoft.WindowsAppRuntime.1.3", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.Windows.Photos"; Match = "Windows.Photos"; License = "Windows.Photos"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.WindowsAlarms"; Match = "WindowsAlarms"; License = "WindowsAlarms"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop", "Microsoft.WindowsAppRuntime.1.3"); }

@{ Name = "Microsoft.WindowsCalculator"; Match = "WindowsCalculator"; License = "WindowsCalculator"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.WindowsCamera"; Match = "WindowsCamera"; License = "WindowsCamera"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.WindowsFeedbackHub"; Match = "WindowsFeedbackHub"; License = "WindowsFeedbackHub"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.WindowsNotepad"; Match = "WindowsNotepad"; License = "WindowsNotepad"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.WindowsSoundRecorder"; Match = "WindowsSoundRecorder"; License = "WindowsSoundRecorder"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.WindowsTerminal"; Match = "WindowsTerminal"; License = "WindowsTerminal"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8"); }

@{ Name = "Microsoft.Xbox.TCUI"; Match = "Xbox.TCUI"; License = "Xbox.TCUI"; Region = "All"; Dependencies = @("Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.XboxGamingOverlay"; Match = "XboxGamingOverlay"; License = "XboxGamingOverlay"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.XboxIdentityProvider"; Match = "XboxIdentityProvider"; License = "XboxIdentityProvider"; Region = "All"; Dependencies = @("Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.XboxSpeechToTextOverlay"; Match = "XboxSpeechToTextOverlay"; License = "XboxSpeechToTextOverlay"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00"); }

@{ Name = "Microsoft.YourPhone"; Match = "YourPhone"; License = "YourPhone"; Region = "All"; Dependencies = @("Microsoft.WindowsAppRuntime.1.4", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.ZuneMusic"; Match = "ZuneMusic"; License = "ZuneMusic"; Region = "All"; Dependencies = @("Microsoft.UI.Xaml.2.8", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime.2.2", "Microsoft.VCLibs.140.00"); }

@{ Name = "MicrosoftCorporationII.QuickAssist"; Match = "QuickAssist"; License = "QuickAssist"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "Microsoft.CrossDevice"; Match = "CrossDevice"; License = "CrossDevice"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "MSTeams"; Match = "MSTeams\*{ARCHC}"; License = "MSTeams\*{ARCHC}"; Region = "AD;AE;AF;AG;AI;AL;AM;AO;AQ;AR;AS;AT;AU;AW;AX;AZ;BA;BB;BD;BE;BF;BG;BH;BI;BJ;BL;BM;BN;BO;BQ;BR;BS;BT;BV;BW;BY;BZ;CA;CC;CD;CF;CG;CH;CI;CK;CL;CM;CO;CR;CU;CV;CW;CX;CY;CZ;DE;DJ;DK;DM;DO;DZ;EC;EE;EG;ER;ES;ET;FI;FJ;FK;FM;FO;FR;GA;GB;GD;GE;GF;GG;GH;GI;GL;GM;GN;GP;GQ;GR;GS;GT;GU;GW;GY;HK;HM;HN;HR;HT;HU;ID;IE;IL;IM;IN;IO;IQ;IR;IS;IT;JE;JM;JO;JP;KE;KG;KH;KI;KM;KN;KP;KR;KW;KY;KZ;LA;LB;LC;LI;LK;LR;LS;LT;LU;LV;LY;MA;MC;MD;ME;MF;MG;MH;MK;ML;MM;MN;MO;MP;MQ;MR;MS;MT;MU;MV;MW;MX;MY;MZ;NA;NC;NE;NF;NG;NI;NL;NO;NP;NR;NU;NZ;OM;PA;PE;PF;PG;PH;PK;PL;PM;PN;PR;PS;PT;PW;PY;QA;RE;RO;RS;RU;RW;SA;SB;SC;SD;SE;SG;SH;SI;SJ;SK;SL;SM;SN;SO;SR;SS;ST;SV;SX;SY;SZ;TC;TD;TF;TG;TH;TJ;TK;TL;TM;TN;TO;TR;TT;TV;TW;TZ;UA;UG;UM;US;UY;UZ;VA;VC;VE;VG;VI;VN;VU;WF;WS;XK;YE;YT;ZA;ZM;ZW"; Dependencies = @("Microsoft.WindowsAppRuntime.1.3", "Microsoft.VCLibs.140.00", "Microsoft.VCLibs.140.00.UWPDesktop"); }

@{ Name = "MicrosoftCorporationII.MicrosoftFamily"; Match = "MicrosoftFamily"; License = "MicrosoftFamily"; Region = "All"; Dependencies = @("Microsoft.VCLibs.140.00.UWPDesktop"); }

)

}

$Allow\_Install\_App = @()

ForEach ($item in $Pre\_Config\_Rules.Edition) {

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 }

Function Match\_InBox\_Apps\_Install\_Pack

{

param ( $NewPath )

$NewArch = $Arch

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

$NewArchCTag = $Arch.Replace("AMD64", "x64")

if ($Arch -eq "arm64") { $NewArchCTag = "arm" }

if ($Pre\_Config\_Rules.Rule.Count -gt 0) {

ForEach ($itemInBoxApps in $Pre\_Config\_Rules.Rule){

$InstallPacker = ""

$InstallPackerCert = ""

$SearchNewStructure = $itemInBoxApps.Match.Replace("{ARCH}", $NewArch).Replace("{ARCHC}", $NewArchC).Replace("{ARCHTag}", $NewArchCTag)

$SearchNewLicense = $itemInBoxApps.License.Replace("{ARCH}", $NewArch).Replace("{ARCHC}", $NewArchC).Replace("{ARCHTag}", $NewArchCTag)

Get-ChildItem -Path $NewPath -Filter "\*$($SearchNewStructure)\*" -Include "\*.appx", "\*.appxbundle", "\*.msixbundle", "\*.msix" -Recurse -Force -ErrorAction SilentlyContinue | ForEach-Object {

if (Test-Path -Path $\_.FullName -PathType Leaf) {

$InstallPacker = $\_.FullName

Get-ChildItem -Path $NewPath -Filter "\*$($SearchNewLicense)\*" -Include \*.xml -Recurse -Force -ErrorAction SilentlyContinue | ForEach-Object {

$InstallPackerCert = $\_.FullName

}

$Script:InBoxAppx += @{

Name = $itemInBoxApps.Name

Depend = $itemInBoxApps.Dependencies

Region = $itemInBoxApps.Region

Search = $SearchNewStructure

InstallPacker = $InstallPacker

Certificate = $InstallPackerCert

CertificateRule = $SearchNewLicense

}

return

}

}

}

}

}

Write-host "`n InBox Apps: Installation packages, automatic search for full disk or specified paths" -ForegroundColor Yellow

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

$Script:InBoxAppx = @()

if ($ISO -eq "Auto") {

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

$AppPath = Join-Path -Path $\_.Root -ChildPath "packages" -ErrorAction SilentlyContinue

Match\_InBox\_Apps\_Install\_Pack -NewPath $AppPath

}

} else { Match\_InBox\_Apps\_Install\_Pack -NewPath $ISO }

Write-host " Search Complete" -ForegroundColor Green

Write-host "`n InBox Apps: Installer Match Results" -ForegroundColor Yellow

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

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

Write-host " Match successful" -ForegroundColor Green

} else { Write-host " Failed match" -ForegroundColor Red; return; }

Write-host "`n InBox Apps: Details of the application to be installed ( $($Script:InBoxAppx.Count) item )" -ForegroundColor Yellow

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

ForEach ($Rule in $Script:InBoxAppx) {

Write-host " Apps name: ".PadRight(22) -NoNewline; Write-host $Rule.Name -ForegroundColor Yellow

Write-host " Region: ".PadRight(22) -NoNewline; Write-host $Rule.Region -ForegroundColor Yellow

Write-host " Apps installer: ".PadRight(22) -NoNewline; Write-host $Rule.InstallPacker -ForegroundColor Yellow

Write-host " License: ".PadRight(22) -NoNewline; Write-host $Rule.Certificate -ForegroundColor Yellow

Write-host ""

}

Write-host "`n InBox Apps: Installation" -ForegroundColor Yellow

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

ForEach ($Rule in $Script:InBoxAppx) {

Write-host " 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.InstallPacker -ForegroundColor Yellow

Write-host " Search for License: " -NoNewline; Write-host $Rule.Certificate -ForegroundColor Yellow

if (Test-Path -Path $Rule.InstallPacker -PathType Leaf) {

Write-host " Region: " -NoNewline; Write-host $Rule.Region -ForegroundColor Yellow

if ([string]::IsNullOrEmpty($Rule.Certificate)) {

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

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

try {

Add-AppxProvisionedPackage -Path $Mount -PackagePath $Rule.InstallPacker -SkipLicense -Regions $Rule.Region -ErrorAction SilentlyContinue | Out-Null

Write-Host "Done`n" -ForegroundColor Green

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

} else {

if (Test-Path -Path $Rule.Certificate -PathType Leaf) {

Write-host " License: " -NoNewline; Write-host $Rule.Certificate -ForegroundColor Yellow;

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

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

try {

Add-AppxProvisionedPackage -Path $Mount -PackagePath $Rule.InstallPacker -LicensePath $Rule.Certificate -Regions $Rule.Region -ErrorAction SilentlyContinue | Out-Null

Write-Host "Done`n" -ForegroundColor Green

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

} else {

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

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

try {

Add-AppxProvisionedPackage -Path $Mount -PackagePath $Rule.InstallPacker -SkipLicense -Regions $Rule.Region -ErrorAction SilentlyContinue | Out-Null

Write-Host "Done`n" -ForegroundColor Green

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

}

}

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

} else { Write-host " Skip the installation`n" -ForegroundColor Red }

}

###### ffline image has language tag installed: Remove

* After completing the installation of the InBox Apps application, the Region tag is no longer important and can be deleted or not deleted. Delete "Local Language Experience Package (LXPs), Simplified Chinese - China)" and uniquely identify the Region tag: zh-CN, can be changed to other Region tag.
* Install.Clear.Flag.ps1
  + [\Expand\Install\Install.Clear.Flag.ps1](Expand/Install/Install.Clear.Flag.ps1)
  + [https://github.com/ilikeyi/solutions/blob/main/\_Learn/Packaging.tutorial/OS.11/25H2/Expand/Install/Install.Clear.Flag.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/Install.Clear.Flag.ps1)
* Copy the code

$Lang = "zh-CN"

Get-AppXProvisionedPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" | Foreach-object {

if ($\_.DisplayName -Like "\*LanguageExperiencePack\*$($Lang)\*") {

Write-host " $($\_.DisplayName)"; Write-Host " Deleting ".PadRight(22) -NoNewline

try {

Remove-AppxProvisionedPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" -PackageName $\_.PackageName -ErrorAction SilentlyContinue | Out-Null

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

}

}

}

###### InBox Apps: optimization

After the app is installed, provisioning the Appx package should be optimized to reduce the app's disk usage by replacing identical files with hard links, only for offline images.

Optimize-AppXProvisionedPackages -Path "D:\OS\_11\_Custom\Install\Install\Mount"

##### Cumulative updates

* + - * When upgrading different versions or older versions to the latest version, you need to add the "checkpoint update" first before adding the latest cumulative update;
      * After adding the language pack, you can install the same cumulative update as the initial version to solve the known issue that the "Component: All packages installed in the image" status is not refreshed after installation;
      * To stay up to date, it is recommended that you download the latest version.

###### Feature Enablement Package

* Before you add the latest cumulative update, you need to add the feature enablement package first, otherwise you cannot add the latest cumulative update. Go to the download page: <https://www.catalog.update.microsoft.com/Search.aspx?q=KB5054156>, and save it to: D:\ OS\_11\_Custom\Install\Install\Update, or download through direct connection, select download according to the architecture:
  + 1. x64 eKB, default
* Direct download

<https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/67731ce0-1988-426a-afa3-044a032eb6c6/public/windows11.0-kb5054156-x64_9fd13360d2c5af23ec4f591b86f1c1db37aada37.msu>

* Add to

$KBPath = "D:\OS\_11\_Custom\Install\Install\Update\windows11.0-kb5054156-x64\_9fd13360d2c5af23ec4f591b86f1c1db37aada37.msu"

Add-WindowsPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" -PackagePath $KBPath

* + 1. Arm64 eKB
* Direct download

<https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/2e42e2b9-993c-4cc5-a8c5-6e20fc64b17e/public/windows11.0-kb5054156-arm64_e57cf6837a8b9877786ec3e7a366503a6072f5e5.msu>

* Add to

$KBPath = "D:\OS\_11\_Custom\Install\Install\Update\windows11.0-kb5054156-arm64\_e57cf6837a8b9877786ec3e7a366503a6072f5e5.msu"

Add-WindowsPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" -PackagePath $KBPath

###### Initial release cumulative update

* After adding the feature enablement package, add the cumulative update with the same initial version. Otherwise, the latest cumulative update cannot be added. Go to the download page: <https://www.catalog.update.microsoft.com/Search.aspx?q=KB5043080>, download and save to: D:\OS\_11\_Custom\Install\Install\Update, or download through direct connection and select the download according to the architecture:
  + 1. x64, default
* Direct download

<https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/d8b7f92b-bd35-4b4c-96e5-46ce984b31e0/public/windows11.0-kb5043080-x64_953449672073f8fb99badb4cc6d5d7849b9c83e8.msu>

* Add to

$KBPath = "D:\OS\_11\_Custom\Install\Install\Update\windows11.0-kb5043080-x64\_953449672073f8fb99badb4cc6d5d7849b9c83e8.msu"

Add-WindowsPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" -PackagePath $KBPath

* + 1. Arm64
* Direct download

<https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/f263aa64-f367-42f0-9cad-328f342b93f7/public/windows11.0-kb5043080-arm64_df540a05f9b118e339c5520f4090bb5d450f090b.msu>

* Add to

$KBPath = "D:\OS\_11\_Custom\Install\Install\Update\windows11.0-kb5043080-arm64\_df540a05f9b118e339c5520f4090bb5d450f090b.msu"

###### Add-WindowsPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" -PackagePath $KBPath

###### Other versions

* Sequence: Add the feature enablement package (KB5054156), add the initial version cumulative update (KB5043080), and then add other cumulative updates, otherwise they cannot be installed.
* Check "[Windows 11 version information](https://learn.microsoft.com/en-us/windows/release-health/windows11-release-information)", for example, download cumulative update: KB5065789, Version: 26200.6725, go to the download page: <https://www.catalog.update.microsoft.com/Search.aspx?q=KB5065789>, download and save to: D:\OS\_11\_Custom \Install\Install\Update, or download through direct connection, select download according to the architecture:
  + 1. x64, default
* Direct download

<https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/72f34af9-6ab3-4bb0-9dfe-103e56a305fb/public/windows11.0-kb5044284-x64_d7eb7ceaa4798b92b988fd7dcc7c6bb39476ccf3.msu>

* Add to

$KBPath = "D:\OS\_11\_Custom\Install\Install\Update\windows11.0-kb5044284-x64\_d7eb7ceaa4798b92b988fd7dcc7c6bb39476ccf3.msu"

Add-WindowsPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" -PackagePath $KBPath

* + 1. Arm64
* Direct download

<https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/02370eb1-b77a-4986-8fa7-5f834b10133f/public/windows11.0-kb5044284-arm64_c2971fdbcc0bc11bf914c23465e88e812614d8c4.msu>

* Add to

$KBPath = "D:\OS\_11\_Custom\Install\Install\Update\windows11.0-kb5044284-arm64\_c2971fdbcc0bc11bf914c23465e88e812614d8c4.msu"

Add-WindowsPackage -Path "D:\OS\_11\_Custom\Install\Install\Mount" -PackagePath $KBPath

###### Solidify Updated

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

Dism /Image:"D:\OS\_11\_Custom\Install\Install\Mount" /cleanup-image /StartComponentCleanup /ResetBase

* + 1. Clean components after curing and updating

$Mount = "D:\OS\_11\_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", if added to ISO installation media, can skip adding to mounted.
* After adding the deployment engine, continue at the current location.

##### Health

Before saving, check whether it is damaged. If the health status is abnormal, stop saving.

Repair-WindowsImage -Path "D:\OS\_11\_Custom\Install\Install\Mount" -ScanHealth

##### Replace WinRE.wim

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

$WinRE = "D:\OS\_11\_Custom\Install\Install\Update\Winlib\WinRE.wim"

$CopyTo = "D:\OS\_11\_Custom\Install\Install\Mount\Windows\System32\Recovery"

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

##### Save image: Install.wim

Save-WindowsImage -Path "D:\OS\_11\_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\_11\_Custom\Install\Install\Mount" -Discard

LOOP OPERATING AREA, END.

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

###### Obtain 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 Install.wim 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.11/25H2/Expand/Install/Install.WinRE.Extract.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/Install.WinRE.Extract.ps1)
* Copy the code

$Arguments = @(

"extract",

"D:\OS\_11\sources\install.wim", "1",

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

"--dest-dir=""D:\OS\_11\_Custom\Install\Install\Update\Winlib"""

)

New-Item -Path "D:\OS\_11\_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.11/25H2/Expand/Install/Install.WinRE.Replace.wim.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/Install.WinRE.Replace.wim.ps1)
* Copy the code

Get-WindowsImage -ImagePath "D:\OS\_11\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\_11\sources\install.wim", $\_.ImageIndex,

"--command=""add 'D:\OS\_11\_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.11/25H2/Expand/Install/Install.Rebuild.wim.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/Install.Rebuild.wim.ps1)
* Copy the code

$InstallWim = "D:\OS\_11\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

}

##### 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.11/25H2/Expand/Install/Install.Split.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/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\_11\sources\install.wim" -SplitImagePath "D:\OS\_11\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\_11\sources\install.swm" -PathType leaf) {

Remove-Item -Path "D:\OS\_11\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.11/25H2/Expand/Install/Install.Merging.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/Install.Merging.ps1)
* Copy the code

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

Get-WindowsImage -ImagePath "D:\OS\_11\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\_11\Sources\install.swm" /swmfile:"D:\OS\_11\sources\install\*.swm" /SourceIndex:"$($\_.ImageIndex)" /DestinationImageFile:"D:\OS\_11\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\_11\Sources\install.wim" -PathType leaf) {

Get-ChildItem -Path "D:\OS\_11\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.11/25H2/Expand/Install/Install.Compress.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/Install.Compress.ps1)
* Copy the code

Write-host "Solid compressed Install.wim";

Get-WindowsImage -ImagePath "D:\OS\_11\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\_11\Sources\install.wim" /SourceIndex:"$($\_.ImageIndex)" /DestinationImageFile:"D:\OS\_11\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\_11\Sources\install.esd" -PathType leaf) {

Remove-Item -Path "D:\OS\_11\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.11/25H2/Expand/Install/Install.Convert.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Install/Install.Convert.ps1)
* Copy the code

Write-host "Convert ESD to WIM";

Get-WindowsImage -ImagePath "D:\OS\_11\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\_11\Sources\install.esd" -SourceIndex $\_.ImageIndex -DestinationImagePath "D:\OS\_11\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\_11\Sources\install.wim" -PathType leaf) {

Remove-Item -Path "D:\OS\_11\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\_11\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\_11\_Custom\Boot\Boot\Mount" -ItemType directory

##### Start mounting Boot.wim

Default index number: 2

Mount-WindowsImage -ImagePath "D:\OS\_11\sources\boot.wim" -Index "2" -Path "D:\OS\_11\_Custom\Boot\Boot\Mount"

##### Language pack: Boot

* 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.11/25H2/Expand/Boot/Boot.Instl.lang.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Boot/Boot.Instl.lang.ps1)
* Copy the code

$Mount = "D:\OS\_11\_Custom\Boot\Boot\Mount"

$Sources = "D:\OS\_11\_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-LanguagePack-Package\*"; File = "lp.cab"; }

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

@{ Match = "\*WinPE\*Setup\*Package\*"; File = "WinPE-Setup\_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 = "\*SecureStartup\*"; File = "winpe-securestartup\_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 = "\*WinPE-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

}

}

}

###### Offline image language: change

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

Region: zh-CN

Dism /Image:"D:\OS\_11\_Custom\Boot\Boot\Mount" /Set-AllIntl:"zh-CN"

* + 1. View available language settings

Dism /Image:"D:\OS\_11\_Custom\Boot\Boot\Mount" /Get-Intl

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

* + 1. View

Get-WindowsPackage -Path "D:\OS\_11\_Custom\Boot\Boot\Mount" | Out-GridView

* + 1. Export to csv

$SaveTo = "D:\OS\_11\_Custom\Boot\Boot\Report.Components.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-WindowsPackage -Path "D:\OS\_11\_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\_11\_Custom\Boot\Boot\Mount\sources\zh-CN" -Destination "D:\OS\_11\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

Regenerated Lang.ini file location: D:\OS\_11\_Custom\Boot\Boot\Mount\Sources\lang.ini

Dism /image:"D:\OS\_11\_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS\_11\_Custom\Boot\Boot\Mount"

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

Regenerated Lang.ini file location: D:\OS\_11\Sources\lang.ini

Dism /image:"D:\OS\_11\_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS\_11"

##### Cumulative updates

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

###### Add

$KBPath = "D:\OS\_11\_Custom\Boot\Boot\Update\KB\_Boot.cab"

Add-WindowsPackage -Path "D:\OS\_11\_Custom\Boot\Boot\Mount" -PackagePath $KBPath

###### Delete

$KBPath = "D:\OS\_11\_Custom\Boot\Boot\Update\KB\_Boot.cab"

Remove-WindowsPackage -Path "D:\OS\_11\_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\_11\_Custom\Boot\Boot\Mount" /cleanup-image /StartComponentCleanup /ResetBase

* + 1. Clean components after curing and updating

$Mount = "D:\OS\_11\_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

##### Other

###### Bypass TPM check during installation

* Boot.Bypass.TPM.ps1
  + [\Expand\Boot\Boot.Bypass.TPM.ps1](Expand/Boot/Boot.Bypass.TPM.ps1)
  + [https://github.com/ilikeyi/solutions/blob/main/\_Learn/Packaging.tutorial/OS.11/25H2/Expand/Boot/Boot.Bypass.TPM.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/Boot/Boot.Bypass.TPM.ps1)
* Copy the code

$RegSystem = "D:\OS\_11\_Custom\Boot\Boot\Mount\Windows\System32\Config\SYSTEM"

$RandomGuid = [guid]::NewGuid()

Write-Host " HKLM:\$($RandomGuid)"

New-PSDrive -PSProvider Registry -Name OtherTasksTPM -Root HKLM -ErrorAction SilentlyContinue | Out-Null

Start-Process reg -ArgumentList "Load ""HKLM\$($RandomGuid)"" ""$($RegSystem)""" -Wait -WindowStyle Hidden -ErrorAction SilentlyContinue

New-Item "HKLM:\$($RandomGuid)\Setup\LabConfig" -force -ea SilentlyContinue | Out-Null

New-ItemProperty -LiteralPath "HKLM:\$($RandomGuid)\Setup\LabConfig" -Name "BypassCPUCheck" -Value 1 -PropertyType DWord -Force -ea SilentlyContinue | Out-Null

New-ItemProperty -LiteralPath "HKLM:\$($RandomGuid)\Setup\LabConfig" -Name "BypassStorageCheck" -Value 1 -PropertyType DWord -Force -ea SilentlyContinue | Out-Null

New-ItemProperty -LiteralPath "HKLM:\$($RandomGuid)\Setup\LabConfig" -Name "BypassRAMCheck" -Value 1 -PropertyType DWord -Force -ea SilentlyContinue | Out-Null

New-ItemProperty -LiteralPath "HKLM:\$($RandomGuid)\Setup\LabConfig" -Name "BypassTPMCheck" -Value 1 -PropertyType DWord -Force -ea SilentlyContinue | Out-Null

New-ItemProperty -LiteralPath "HKLM:\$($RandomGuid)\Setup\LabConfig" -Name "BypassSecureBootCheck" -Value 1 -PropertyType DWord -Force -ea SilentlyContinue | Out-Null

[gc]::collect()

Start-Process reg -ArgumentList "unload ""HKLM\$($RandomGuid)""" -Wait -WindowStyle Hidden -ErrorAction SilentlyContinue

Remove-PSDrive -Name OtherTasksTPM

##### Save image: Boot.wim

Save-WindowsImage -Path "D:\OS\_11\_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\_11\_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\_11\Autounattend.xml

Copy-Item "D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS\_11\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\_11\Sources\Unattend.xml

Copy-Item "D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS\_11\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\_11\sources\`$OEM$\`$$\Panther" -ItemType Directory

* + - * 1. Copy

Copy D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml to D:\OS\_11\Sources\$OEM$\Panther\Unattend.xml

Copy-Item "D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS\_11\sources\`$OEM$\`$$\Panther\Unattend.xml" -Force

* + 1. Deployment engine: add

Add "Automatically add installed languages for Windows systems" to D:\OS\_11\sources\$OEM$\$1\Yi\Engine in the directory.

* + - 1. Deployment Engine: Copy

Copy D:\Multilingual\Engine to D:\OS\_11\Sources\$OEM$\$1\Yi\Engine

Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS\_11\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\_11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -ErrorAction SilentlyContinue | Out-Null

Out-File -FilePath "D:\OS\_11\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\_11\_Custom\Install\Install\Mount

* + 1. Unattended

Copy D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml to D:\OS\_11\_Custom\Install\Install\Mount\Panther\Unattend.xml

Copy-Item "D:\Multilingual\\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS\_11\_Custom\Install\Install\Mount\Panther" -Force

* + 1. Deployment engine: add

Add "Automatically add languages installed on Windows systems" to the D:\OS\_11\_Custom\Install\Install\Mount\Yi\Engine directory.

* + - 1. Deployment Engine: Copy

Copy D:\Multilingual\Engine to D:\OS\_11\_Custom\Install\Install\Mount\Yi\Engine

Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS\_11\_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\_11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -ErrorAction SilentlyContinue | Out-Null

Out-File -FilePath "D:\OS\_11\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\_11\Autounattend.xml
* D:\OS\_11\Sources\Unattend.xml
* D:\OS\_11\sources\$OEM$\$$\Panther\unattend.xml
* D:\OS\_11\_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 Region tag, for example, specify a Region 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>

### ISO

#### Generate ISO

1. 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>

1. Use the oscdimg command line to generate an ISO file and save it to: D:\OS\_11.iso

* ISO.ps1
  + [\Expand\ISO.ps1](Expand/ISO.ps1)
  + [https://github.com/ilikeyi/solutions/blob/main/\_Learn/Packaging.tutorial/OS.11/25H2/Expand/ISO.ps1](https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/24H2/Expand/ISO.ps1)
* Copy the code

$Oscdimg = "D:\Oscdimg.exe"

$ISO = "D:\OS\_11"

$Volume = "OS\_11"

$SaveTo = "D:\OS\_11.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

#### Bypass TPM installation check

1. Learn about: <https://github.com/AveYo/MediaCreationTool.bat/tree/main/bypass11> and download: Quick\_11\_iso\_esd\_wim\_TPM\_toggle.bat
2. Drag the D:\OS\_11.iso file to Quick\_11\_iso\_esd\_wim\_TPM\_toggle.bat, and "add" or "delete" the TPM installation check function in reverse order.

## 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\_11\_Custom\Install\Install\Mount" -Discard

Dismount-WindowsImage -Path "D:\OS\_11\_Custom\Install\WinRE\Mount" -Discard

Dismount-WindowsImage -Path "D:\OS\_11\_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. Windows Security Center

When testing a fresh install of the initial version of Windows 11 25H2 26200.6584 (Chinese version), using the image file: zh-cn\_windows\_11\_consumer\_editions\_version\_25h2\_x64\_dvd\_0549fc93.iso, the Windows Security Center appears in English after installation, with the language pack not translated into Chinese. This issue has been occurring since the May/June images of Windows 11 24H2. Installing the "Simplified Chinese Experience Pack" offline or online does not resolve the issue. Known solutions:

* Open Windows Update and the issue will be resolved after the update is pushed.

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: 1.0

Translation: Chinese to English version

Initial public offering time: 10 / 2025

All scripts included in the document, last tested: 10 / 2025

Document last updated: 10 / 2025

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