



MICROSOFT WINDOWS 11 23H2

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

Chapter 1	Encapsulation
Chapter 2	Common problem
Chapter 3	Known issues

Table of contents

Chapter 1	Encapsulation	Page 6
A.	Prerequisites	Page 6
II	Running system	Page 6
1.	When using the DISM command to create a higher version image	Page 6
2.	Disk partition	Page 6
3.	Windows Security	Page 6
4.	ISO tools	Page 7
5.	Command line	Page 7
III	Requirements	Page 7
1.	System installation package	Page 7
2.	Language pack	Page 9
2.1.	Learn	Page 9
2.2.	Language pack: Download	Page 11
2.3.	Language pack: Fixed	Page 11
3.	InBox Apps	Page 12
B.	Language Pack: Extract	Page 13
II	Language pack: Ready	Page 13
III	Language pack: Scheme	Page 13
IV	Execute the extract command	Page 13
C.	Custom encapsulation	Page 19
II	Custom encapsulation Install.wim	Page 19
1.	View Install.wim details	Page 19
2.	Specify the path to mount Install.wim	Page 19
3.	Start mounting Install.wim	Page 19
3.1.	Custom encapsulation WinRE.wim	Page 19
3.1.1.	View WinRE.wim details	Page 20
3.1.2.	Specify the path to mount WinRE.wim	Page 20
3.1.3.	Start mounting WinRE.wim	Page 20
3.1.4.	Language pack: WinRE	Page 20
3.1.4.1.	Language pack: add	Page 20

3.1.4.2.	Offline image language: change	Page 22
3.1.4.2.1.	Change default language, regional settings, and other international settings	Page 22
3.1.4.2.2.	View available language settings	Page 22
3.1.4.3.	Components: All packages installed in the image	Page 22
3.1.5.	Cumulative updates	Page 22
3.1.5.1.	Add	Page 22
3.1.5.2.	Delete	Page 22
3.1.5.3.	Solid update	Page 22
3.1.5.3.1.	Clean components after curing and updating	Page 23
3.1.6.	Drive	Page 23
3.1.7.	Save image: WinRE.wim	Page 23
3.1.8.	Unmount image: WinRE.wim	Page 23
3.1.9.	After rebuilding WinRE.wim, the file size can be reduced	Page 23
3.1.10.	Backup WinRE.wim	Page 24
3.1.11.	Replace WinRE.wim within the Install.wim image	Page 24
4.	Language pack	Page 24
4.1.	Language pack: add	Page 25
4.2.	Offline image language: change	Page 30
4.2.1.	Change default language, regional settings, and other international settings	Page 31
4.2.2.	View available language settings	Page 31
4.3.	Components: All packages installed in the image	Page 31
5.	InBox Apps	Page 31
5.1.	Learn	Page 31
5.2.	InBox Apps: Pre-install the application	Page 31
5.3.	InBox Apps: Installed	Page 33
5.4.	Remove all installed pre-applications	Page 33
5.5.	Installing "Local Language Experience Packs (LXPs)" to offline images marks: installed languages	Page 34
5.6.	Offline image has language tag installed: How to add it	Page 34
5.7.	InBox Apps: Install	Page 37
5.8.	Offline image has language tag installed: Remove	Page 47

5.9.	InBox Apps: optimization	Page 47
6.	Cumulative updates	Page 47
6.1.	Exclusive feature enablement package	Page 48
6.2.	Initial version	Page 48
6.3.	Other versions	Page 49
6.4.	Solidify Updated	Page 50
6.4.1.	Clean components after curing and updating	Page 50
7.	Drive	Page 50
8.	Deployment engine: Add	Page 50
9.	Health	Page 50
10.	Replace WinRE.wim	Page 50
11.	Save image: Install.wim	Page 50
12.	Unmount image: Install.wim	Page 50
13.	How to batch replace WinRE.wim in all index numbers in Install.wim	Page 51
13.1.	Obtain WimLib	Page 51
13.2.	How to extract and update WinRE.wim in Install.wim	Page 51
14.	Rebuilding Install.wim reduces file size	Page 52
15.	Split, merge, compress, and convert	Page 52
15.1.	Splitting and merging	Page 53
15.1.1.	Splitting	Page 53
15.1.2.	Merge	Page 53
15.2.	Solid compressed ESD format and WIM format conversion	Page 54
15.2.1.	Solid compression	Page 54
15.2.2.	Convert compressed files to WIM file format	Page 55
III	Custom encapsulation boot.wim	Page 56
1.	View Boot.wim details	Page 56
2.	Specify the path to mount Boot.wim	Page 56
3.	Start mounting Boot.wim	Page 56
4.	Language pack: Boot	Page 56
4.1.	Language pack: Add	Page 56

4.2.	Offline image language: change	Page 57
4.2.1.	Change default language, regional settings, and other international settings	Page 58
4.2.2.	View available language settings	Page 58
4.3.	Components: All packages installed in the image	Page 58
4.4.	Language: Repair	Page 58
4.5.	Language packs: sync to ISO installer	Page 58
4.6.	Regenerate Lang.ini	Page 58
4.6.1.	Regenerate the mounted directory lang.ini	Page 58
4.6.2.	After regenerating lang.ini, sync to the installer	Page 59
5.	Cumulative updates	Page 59
5.1.	Add	Page 59
5.2.	Delete	Page 59
5.3.	Solid update	Page 59
5.3.1.	Clean components after curing and updating	Page 59
6.	Drive	Page 59
7.	Other	Page 59
7.1.	Bypass TPM check during installation	Page 59
8.	Save image: Boot.wim	Page 60
9.	Unmount image: Boot.wim	Page 60
IV	Deployment engine	Page 60
1.	Add method	Page 61
2.	Deployment Engine: Advanced	Page 64
D.	ISO	Page 66
II	Generate ISO	Page 66
III	Bypass TPM installation check	Page 66
Chapter 2	Common problem	Page 67
II	Clean all mounts to	Page 67
III	Fix the problem of abnormal mounting	Page 67
IV	Clean up	Page 67

1. When using the DISM command to create a higher version image

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

1.1. Upgrade the running operating system or reinstall to a higher version (recommended);

1.2. Upgrade or install a new version of ADK or PowerShell (not recommended)

1.2.1. You can try to upgrade to the latest PowerShell 7 or higher version;

1.2.2. 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.

2. Disk partition

2.1. 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.

2.2. After the ISO is decompressed, its location is not affected by the REFS partition.

3. Windows Security

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

How to close:

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

3.1. Open Windows Security or run: `windowsdefender:`

3.2. Select "Virus & Threat Protection" or Run: `windowsdefender://threat`

3.3. Find "Virus & Threat Protection Settings", click "Manage Settings" or Run: `windowsdefender://threatsettings`, we recommend that you turn off some features:

3.3.1. Real-time protection

3.3.2. Cloud=delivered protection

3.3.3. Automatic sample submission

3.3.4. Tamper Protection

3.4. When you're not encapsulated, we recommend that you turn on Windows Security.

4. ISO tools

Use a software that can edit ISO files, such as: [PowerISO](#), [DAEMON Tools](#), [ISO Workshop](#);

5. Command line

5.1. Optional "Terminal" or "PowerShell ISE", if "Terminal" is not installed, please go to: <https://github.com/microsoft/terminal/releases>
After downloading;

5.2. 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
```

5.3. In this article, PS command line, green part, please copy it, paste it into the "Terminal" dialog box, press Enter and start running;

5.4. 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"](#)

II Requirements

1. System installation package

Keywords: [iteration](#), [cross-version](#), [major version](#), [cumulative update](#), [initial release](#)

1.1. illustrate

1.1.1. Please remake the image when each version is updated, for example, when crossing from 21H1 to 22H2, avoid other compatibility problems, and do not make the image based on the old image;

1.1.2. The regulation has been clearly communicated to packagers in various forms by some OEMs, and direct upgrades from iterative versions are not allowed;

1.1.3. 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;

1.1.4. 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.

1.1.5. Evolutionary process: Windows 11 23H2, Build 22631.2428 + (KB5027397 + kb5035853) = OS Build 22631.3296

1.2. Prepare to download the initial or developer version

1.2.1. x64

1.2.1.1. Filename: [en-us_windows_11_business_editions_version_23h2_x64_dvd_a9092734.iso](#)

List of files: <https://files.rg-adguard.net/file/4844bd9b-9676-c00e-76d5-31f9be8d968e>

1.2.1.2. Filename: [en-us_windows_11_consumer_editions_version_23h2_x64_dvd_8ea907fb.iso](#)

List of files: <https://files.rg-adguard.net/file/fa33f04b-e9f3-0b7a-99e5-cd76085cb069>

1.2.1.3. Filename: [en-us_windows_11_iot_enterprise_version_23h2_x64_dvd_fb37549c.iso](#)

List of files: <https://files.rg-adguard.net/file/97a5761d-6736-b102-af2b-488197780556>

1.2.2. arm64

1.2.2.1. Filename: [en-us_windows_11_iot_enterprise_version_23h2_arm64_dvd_6cc52d75.iso](#)

List of files: <https://files.rg-adguard.net/file/f2f8c73e-3e4a-de14-e71e-ddae3bc0ef99>

1.3. After the sample download [en-us_windows_11_business_editions_version_23h2_x64_dvd_a9092734.iso](#), Unzip to: [D:\en-us_windows_11_business_editions_version_23h2_x64_dvd_a9092734](#)

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.

1.4. After decompression is complete, change the directory [D:\en-us_windows_11_business_editions_version_23h2_x64_dvd_a9092734](#) change to [D:\OS_11](#)

1.5. All scripts and all paths are set to [D:\OS_11](#) as the image source.

1.6. Installation configuration

1.6.1. Learn: [Windows Setup Edition Configuration and Product ID Files \(Ei.cfg and PID.txt \)](#)

1.6.2. Known issues

1.6.2.1. 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 installation sources are valid and restart the installation.

1.6.2.2. 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
```

1.7. Report

Check the installed image package: language packs, associated language pack components, pre-installed InBox Apps applications and other detailed information. After downloading the report, use Excel software to open the .xlsx.

1.7.1. Report: Overview

Contains: ISO system installation package, image version, region, region association, associated language pack components, etc.

[Report.Overview.xlsx](#)

○ [Expand\Install\Report\Report.Overview.xlsx](#)

- https://raw.githubusercontent.com/ilikeyi/solutions/main/_Learn/Packaging/tutorial/OS.11/23H2/Expand/Install/Report/Report.Overview.xlsx

1.7.2. Report: Complete

Inclusion: ISO system installation package, image version, region, region association, associated language pack components, installed InBox Apps pre-installed applications, etc.

[en-us](#), [ar-sa](#), [bg-bg](#), [cs-cz](#), [da-dk](#), [de-de](#), [el-gr](#), [en-gb](#), [es-es](#), [es-mx](#), [et-ee](#), [fi-fi](#), [fr-ca](#), [fr-fr](#), [he-il](#), [hr-hr](#), [hu-hu](#), [it-it](#), [ja-jp](#), [ko-kr](#), [lt-lt](#), [lv-lv](#), [nb-no](#), [nl-nl](#), [pl-pl](#), [pt-br](#), [pt-pt](#), [ro-ro](#), [ru-ru](#), [sk-sk](#), [sl-si](#), [sr-latn-rs](#), [sv-se](#), [th-th](#), [tr-tr](#), [uk-ua](#), [zh-cn](#), [zh-tw](#)

2. Language pack

2.1. Learn

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

2.1.1. [Languages overview](#)

2.1.2. [Add languages to a Windows 11 image](#)

2.1.3. [Language and region Features on Demand \(FOD\)](#)

2.1.3.1. 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](#)" learn more.
- In "Language Pack: Extract", the automatic recognition function has been added, and you can understand the functions: [Function Match_Required_Fonts](#)

2.1.3.2. 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:

2.1.3.2.1. Region: [ar-sa](#), Optional associated areas: [en-US](#), associated installation package:

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

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

2.1.3.2.2. Region: [bg-bg](#), Optional associated areas: [en-US](#), associated installation package:

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

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

- 2.1.3.2.3. Region: **da-dk**, Optional associated areas: **en-US**, associated installation package:
- 2.1.3.2.3.1. Microsoft-Windows-LanguageFeatures-Basic-es-us-
Package~31bf3856ad364e35~amd64~~.cab
 - 2.1.3.2.3.2. Microsoft-Windows-LanguageFeatures-OCR-en-us-
Package~31bf3856ad364e35~amd64~~.cab
- 2.1.3.2.4. Region: **el-gr**, Optional associated areas: **en-US**, associated installation package:
- 2.1.3.2.4.1. Microsoft-Windows-LanguageFeatures-Basic-es-us-
Package~31bf3856ad364e35~amd64~~.cab
 - 2.1.3.2.4.2. Microsoft-Windows-LanguageFeatures-OCR-en-us-
Package~31bf3856ad364e35~amd64~~.cab
- 2.1.3.2.5. Region: **fr-ca**, Optional associated areas: **en-US**, **fr-fr**, associated installation package:
- 2.1.3.2.5.1. Microsoft-Windows-LanguageFeatures-Basic-es-us-
Package~31bf3856ad364e35~amd64~~.cab
 - 2.1.3.2.5.2. Microsoft-Windows-LanguageFeatures-OCR-en-us-
Package~31bf3856ad364e35~amd64~~.cab
 - 2.1.3.2.5.3. Microsoft-Windows-LanguageFeatures-Basic-fr-fr-
Package~31bf3856ad364e35~amd64~~.cab
 - 2.1.3.2.5.4. Microsoft-Windows-LanguageFeatures-Handwriting-fi-fi-
Package~31bf3856ad364e35~amd64~~.cab
- 2.1.3.2.6. Region: **he-il**, Optional associated areas: **en-US**, associated installation package:
- 2.1.3.2.6.1. Microsoft-Windows-LanguageFeatures-Basic-es-us-
Package~31bf3856ad364e35~amd64~~.cab
 - 2.1.3.2.6.2. Microsoft-Windows-LanguageFeatures-OCR-en-us-
Package~31bf3856ad364e35~amd64~~.cab
- 2.1.3.2.7. Region: **ru-ru**, Optional associated areas: **en-US**, associated installation package:
- 2.1.3.2.7.1. Microsoft-Windows-LanguageFeatures-Basic-es-us-
Package~31bf3856ad364e35~amd64~~.cab
 - 2.1.3.2.7.2. Microsoft-Windows-LanguageFeatures-OCR-en-us-
Package~31bf3856ad364e35~amd64~~.cab
- 2.1.3.2.8. Region: **th-th**, Optional associated areas: **en-US**, associated installation package:
- 2.1.3.2.8.1. Microsoft-Windows-LanguageFeatures-Basic-es-us-
Package~31bf3856ad364e35~amd64~~.cab
 - 2.1.3.2.8.2. Microsoft-Windows-LanguageFeatures-OCR-en-us-
Package~31bf3856ad364e35~amd64~~.cab
- 2.1.3.2.9. Region: **uk-ua**, Optional associated areas: **en-US**, associated installation package:

- 2.1.3.2.9.1. Microsoft-Windows-LanguageFeatures-Basic-es-us-
Package~31bf3856ad364e35~amd64~~.cab
- 2.1.3.2.9.2. Microsoft-Windows-LanguageFeatures-OCR-en-us-
Package~31bf3856ad364e35~amd64~~.cab
- 2.1.3.2.10. Region: **zh-TW**, Optional associated areas: **zh-HK**, associated installation package:
 - 2.1.3.2.10.1. Microsoft-Windows-LanguageFeatures-Basic-zh-hk-
Package~31bf3856ad364e35~amd64~~.cab
 - 2.1.3.2.10.2. Microsoft-Windows-LanguageFeatures-Speech-zh-hk-
Package~31bf3856ad364e35~amd64~~.cab
 - 2.1.3.2.10.3. Microsoft-Windows-LanguageFeatures-TextToSpeech-zh-hk-
Package~31bf3856ad364e35~amd64~~.cab
- 2.1.3.3. Other region-specific requirements

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

 - 2.1.3.3.1. 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.

2.2. Language pack: Download

2.2.1. x64

2.2.2. Filename: https://software-static.download.prss.microsoft.com/dbazure/988969d5-f34g-4e03-ac9d-1f9786c66749/22621.1.220506-1250.ni_release_amd64fre_CLIENT_LOF_PACKAGES_OEM.iso

List of files: <https://files.rg-adguard.net/file/4f7bf45f-9970-e758-44cf-876643f49709>

2.3. Language pack: Fixed

2.3.1. Select any website and open:

2.3.1.1. <https://uupdump.net>

2.3.1.2. <https://uup.ee>

2.3.1.3. <https://osdump.com>

2.3.2. After opening, search for keywords: [22631.2428](#), select from the search results: [Windows 11, version 23H2 \(22631.2428\)](#)
[amd64](#)

2.3.3. After opening, select "[All files](#)";

2.3.4. Search the green part in the "[All Files](#)" page and download

2.3.4.1. Applies to: Install.wim:

2.3.4.1.1. [MediaPlayer](#)

2.3.4.2. Applies to: WinRE.wim, none yet

2.3.4.3. Applies to: Boot.wim, none yet

2.3.5. After downloading all the files, scroll to the bottom of the page, download and run "[Generate Rename Script \(Windows\)](#)"

2.3.6. Use ISO editing software, edit [22621.1.220506-1250.ni_release_amd64fre_CLIENT_LOF_PACKAGES_OEM.iso](#), and add the downloaded file to the [\[ISO\]:\LanguagesAndOptionalFeatures](#) directory;

3. InBox Apps

3.1. Download: https://software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/22621.2501.231009-1937.ni_release_svc_prod3_amd64fre_InboxApps.iso

List of files: <https://files.rg-adguard.net/file/5e7beb23-960d-d328-24f3-ef6378cd139d>

3.2. Download: https://software-static.download.prss.microsoft.com/dbazure/988969d5-f34g-4e03-ac9d-1f9786c66749/22621.1.220506-1250.ni_release_amd64fre_InboxApps.iso

List of files: <https://files.rg-adguard.net/file/263fed3d-be1f-9307-0c30-259d1027b2bd>

Download finally, Extract:

3.2.1. Microsoft.HEVCVideoExtension

3.2.1.1. [Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.arm64.appx](#)

3.2.1.2. [Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.arm64.xml](#)

3.2.1.3. [Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.x64.appx](#)

3.2.1.4. [Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.x64.xml](#)

3.2.2. Microsoft.WindowsTerminal

3.2.2.1. [Microsoft.WindowsTerminal_8wekyb3d8bbwe.msixbundle](#)

3.2.2.2. [Microsoft.WindowsTerminal_8wekyb3d8bbwe.xml](#)

Warning

- The installation version number of the [Microsoft.WindowsTerminal_8wekyb3d8bbwe.msixbundle](#) provided in the [22621.1778.230511-2102.ni_release_svc_prod3_amd64fre_InboxApps.iso](#) file is: [1.17.12191](#), and it will not run normally after installing this version number.
- The version number provided in the initial release of Windows 11 23H2 22631.2428 is: [1.12.10983](#), and it will work properly when installed.

3.3. Use the ISO editing tool to edit [22621.1778.230511-2102.ni_release_svc_prod3_amd64fre_InboxApps.iso](#) and add the extracted files to the [\[ISO\]:\packages](#) directory;

B. Language Pack: Extract

II Language pack: Ready

Mount [22621.1.220506-1250.ni_release_amd64fre_CLIENT_LOF_PACKAGES_OEM.iso](#) or unzipped to any location;

III Language pack: Scheme

1. Add

1.1. Language name: [Simplified Chinese - China](#), Region: [zh-CN](#), Scope of application: [Install.Wim](#), [Boot.Wim](#), [WinRE.Wim](#)

2. Delete

2.1. Language name: [English - United States](#), Region: [en-US](#), Scope of application: [Install.Wim](#), [Boot.Wim](#), [WinRE.Wim](#)

IV 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](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging/tutorial/OS.11/23H2/Expand/Extract.ps1

- Copy the code

```
$ISO = "Auto"

$SaveTo = "D:\OS_11_Custom"

$Extract_language_Pack = @(

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

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

)

Function Extract_Language

{

    param( $Act, $NewLang, $Expand )
```

Function Match_Required_Fonts

```
{

    param( $Lang )

    $Fonts = @(

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    )

    ForEach ($item in $Fonts) {

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

```

        return $item.Name

    }

}

return "Not_matched"

}

Function Match_Other_Region_Specific_Requirements

{

    param( $Lang )

    $RegionSpecific = @(

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

    )

    ForEach ($item in $RegionSpecific) {

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

            return $item.Name

        }

    }

    return "Skip_specific_packages"

}

Function Extract_Process

{

    param( $Package, $Name, $NewSaveTo )

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

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

    if ($ISO -eq "Auto") {

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

            ForEach ($item in $Package) {

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

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

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

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

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

                }

            }

        }

    }

}

```



```

} else {

    ForEach ($item in $Package) {

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

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

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

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

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

        } else {

            Write-host " Not found"

        }

    }

}

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

ForEach ($item in $Package) {

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

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

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

    } else {

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

    }

}

$AdvLanguage = @(

    @{

        Path = "Install\Install"

        Rule = @(

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

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

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

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

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

"LanguagesAndOptionalFeatures\Microsoft-Windows-WordPad-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-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

        }

    }

}

}

}

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

```

C. Custom encapsulation

II Custom encapsulation Install.wim

1. 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,

2. Specify the path to mount Install.wim

```
New-Item -Path "D:\OS_11_Custom\Install\Install\Mount" -ItemType directory
```

3. 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,

3.1. 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](#)".

3.1.1. 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 }
```

3.1.2. Specify the path to mount WinRE.wim

```
New-Item -Path "D:\OS_11_Custom\Install\WinRE\Mount" -ItemType directory
```

3.1.3. 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"
```

3.1.4. 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.

3.1.4.1. Language pack: add

- WinRE.Instl.lang.ps1
 - [\Expand\Install\WinRE\WinRE.Instl.lang.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging/tutorial/OS.11/23H2/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_List = @(
```

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

@{ Match = "*StorageWMI*"; File = "winpe-storagewmi_zh-CN.cab"; }

@{ Match = "*WiFi*"; File = "winpe-wifi_zh-CN.cab"; }

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

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

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

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

@{ Match = "*appxdeployment*"; File = "winpe-appxdeployment_zh-CN.cab"; }

)

ForEach ($Rule in $Language_List) {

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

    ForEach ($Component in $Initl_install_Language_Component) {

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

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

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

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

            try {

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

                Write-host "Finish" -ForegroundColor Green

            } catch {

                Write-host "Failed" -ForegroundColor Red

            }
        }
    }
}

```

```
        break
    }
}
}
```

3.1.4.2. Offline image language: change

3.1.4.2.1. 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"
```

3.1.4.2.2. View available language settings

```
Dism /Image:"D:\OS_11_Custom\Install\WinRE\Mount" /Get-Intl
```

3.1.4.3. Components: All packages installed in the image

3.1.4.3.1. View

```
Get-WindowsPackage -Path "D:\OS_11_Custom\Install\WinRE\Mount" | Out-GridView
```

3.1.4.3.2. 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
```

3.1.5. Cumulative updates

To prepare the cumulative updates file available, change the example file name: KB_WinRE.cab

3.1.5.1. Add

```
$KBPath = "D:\OS_11_Custom\Install\WinRE\Update\KB_WinRE.cab"
```

```
Add-WindowsPackage -Path "D:\OS_11_Custom\Install\WinRE\Mount" -PackagePath $KBPath
```

3.1.5.2. Delete

```
$KBPath = "D:\OS_11_Custom\Install\WinRE\Update\KB_WinRE.cab"
```

```
Remove-WindowsPackage -Path "D:\OS_11_Custom\Install\WinRE\Mount" -PackagePath $KBPath
```

3.1.5.3. 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
```

3.1.5.3.1. 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  
  
    }  
  
}
```

3.1.6. Drive

3.1.7. Save image: WinRE.wim

```
Save-WindowsImage -Path "D:\OS_11_Custom\Install\WinRE\Mount"
```

3.1.8. 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
```

3.1.9. After rebuilding WinRE.wim, the file size can be reduced

- WinRE.Rebuild.ps1
 - [\Expand\Install\WinRE\WinRE.Rebuild.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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

}

```

3.1.10. Backup WinRE.wim

- WinRE.Backup.ps1
 - [\Expand\Install\WinRE\WinRE.Backup.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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

```

3.1.11. Replace WinRE.wim within the Install.wim image

- After each mount Install.wim "[Replace WinRE.wim](#)";
- Learn "[Get all index numbers of Install.wim and replace the old WinRE.wim](#)".

[Process the files in the Install.wim image and end.](#)

4. 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.

4.1. Language pack: add

- Install.Instl.lang.ps1
 - [\Expand\Install\Install.Instl.lang.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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 = @(

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

            )

        }

        @{

            Name = "Basic"

            Description = "Basic"

            Rule = @(

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

                @{ Match_Name = "*Client*Language*Pack*"; IsMatch = "Yes"; Capability = "Language.Basic~~~lb-LU~0.0.1.0"; }

            )

        }

        @{

            Name = "OCR"

            Description = "Optical character recognition"

            Rule = @(

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

            )

        }

        @{

            Name = "Handwriting"

            Description = "Handwriting recognition"

            Rule = @(

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

            )

        }
    )
}

```

```

@{

    Name = "TextToSpeech"

    Description = "Text-to-speech"

    Rule = @(

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

    )

}

@{

    Name = "Speech"

    Description = "Speech recognition"

    Rule = @(

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

    )

}

@{

    Name = "RegionSpecific"

    Description = "Other region-specific requirements"

    Rule = @(

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

    )

}

@{

    Name = "Retail"

    Description = "Retail demo experience"

    Rule = @(

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

    )

}

@{

    Name = "Features_On_Demand"

    Description = "Features on demand"

    Rule = @(

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

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

    )

}

```

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    )

}

)

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

ForEach ($Wildcard in $Script:Init_Folder_All_File) {

    ForEach ($item in $Script:Search_File_Order) {

        ForEach ($NewRule in $item.Rule) {

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

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

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

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

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

                $new = $OSDefaultUser + $TempSave

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

                    ForEach ($Component in $InitL_install_Language_Component) {

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

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

```

```

        New-Variable -Name "Init_File_Type_$(Item.Name)" -Value $new -Force

        $Script:Init_Folder_All_File_Match_Done += $WildCard

        break
    }

}

} else {

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

    New-Variable -Name "Init_File_Type_$(Item.Name)" -Value $new -Force

    $Script:Init_Folder_All_File_Match_Done += $WildCard

}

}

}

}

}

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

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

ForEach ($WildCard in $Script:Search_File_Order) {

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

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

    if ($OSDefaultUser.Count -gt 0) {

        ForEach ($item in $OSDefaultUser) {

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

        }

    } else {

        Write-host " Not available" -ForegroundColor Red

    }

}

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

ForEach ($item in $Script:Init_Folder_All_File) {

    if ($Script:Init_Folder_All_File_Match_Done -notcontains $item) {

        $Script:Init_Folder_All_File_Exclude += $item

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

    }

}

}

```

```

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

ForEach ($WildCard in $Script:Search_File_Order) {

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

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

    if ($OSDefaultUser.Count -gt 0) {

        ForEach ($item in $OSDefaultUser) {

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

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

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

                try {

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

                    Write-host "Finish`n" -ForegroundColor Green

                } catch {

                    Write-host "Failed" -ForegroundColor Red

                    Write-host " $_" -ForegroundColor Red

                }

            } else {

                Write-host "Does not exist`n"

            }

        }

    } else {

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

    }

}

Language_Install -Mount "D:\OS_11_Custom\Install\Install\Mount" -Sources
"D:\OS_11_Custom\Install\Install\Language\Add" -Lang "zh-CN"

```

4.2. Offline image language: change

- Starting Windows 11, the [default System UI Language](#) set by DISM is left unaltered on all editions except for Home edition. For [all commercial editions](#) the language chosen during the Out-of-Box Experience (OOBE) is set as the [System Preferred UI language](#) 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.

4.2.1. 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

4.2.2. View available language settings

Dism /Image:"D:\OS_11_Custom\Install\Install\Mount" /Get-Intl

4.3. Components: All packages installed in the image

4.3.1. View

Get-WindowsPackage -Path "D:\OS_11_Custom\Install\Install\Mount" | Out-GridView

4.3.2. 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

5. InBox Apps

5.1. Learn

5.1.1. [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

5.2. InBox Apps: Pre-install the application

- Group abbreviations
 - Group A: Education, Professional, ProfessionalEducation, ProfessionalWorkstation, Enterprise, IoTEnterprise, ServerRdsh, ServerStandard, ServerDatacenter
 - Group B: EnterpriseN, EnterpriseGN, EnterpriseSN, ProfessionalN, EducationN, ProfessionalWorkstationN, ProfessionalEducationN
 - Group C: Core, CoreN, CoreSingleLanguage

	Name	Products ID	Grouping
1.	Microsoft.HEIFImageExtension	9PMMSR1CGPWG	A, C
2.	Microsoft.HEVCVideoExtension	9NMZLZ57R3T7	A, C
3.	Microsoft.VP9VideoExtensions	9N4D0MSMP0PT	A, C

4.	Microsoft.WebplImageExtension	9PG2DK419DRG	A, C
5.	Microsoft.GamingApp	9MV0B5HZVK9Z	A, C
6.	Microsoft.ZuneMusic	9WZDNCRFJ3PT	A, C
7.	MicrosoftCorporationII.MicrosoftFamily	9PDJDS743XF	C
8.	Microsoft.WebMediaExtensions	9N5TDP8VCMHS	A, C
9.	Microsoft.WindowsSoundRecorder	9WZDNCRFHWKN	A, C
10.	Microsoft.Xbox.TCUI	9NKNC0LD5NN6	A, C
11.	Microsoft.XboxGamingOverlay	9NZKPSTSNW4P	A, C
12.	Microsoft.ZuneVideo	9WZDNCRFJ3P2	A, C
13.	Microsoft.RawImageExtension	9NCTDW2W1BH8	A, C
14.	Microsoft.UI.Xaml.2.3		A, B, C
15.	Microsoft.UI.Xaml.2.4		A, B, C
16.	Microsoft.UI.Xaml.2.7		A, B, C
17.	Microsoft.NET.Native.Framework.2.2		A, B, C
18.	Microsoft.NET.Native.Runtime.2.2		A, B, C
19.	Microsoft.VCLibs.140.00		A, B, C
20.	Microsoft.VCLibs.140.00.UWPDesktop		A, B, C
21.	Microsoft.SecHealthUI	9NK0JLZT7JD0	A, B, C
22.	Microsoft.WindowsStore	9WZDNCRFJBMP	A, B, C
23.	Microsoft.Sticky.Notes	9NBLGGH4QGHW	A, B, C
24.	Microsoft.Paint	9PCFS5B6T72H	A, B, C
25.	Microsoft.PowerAutomateDesktop	9NFTCH6J7FHV	A, B, C
26.	Microsoft.ScreenSketch	9MZ95KL8MR0L	A, B, C
27.	Microsoft.WindowsNotepad	9MSMLRH6LZF3	A, B, C
28.	Microsoft.WindowsTerminal	9N0DX20HK701	A, B, C
29.	Clipchamp.Clipchamp	9P1J8S7CCWWT	A, B, C
30.	Microsoft.Solitaire.Collection	9WZDNCRFHWD2	A, B, C
31.	Microsoft.WindowsAlarms	9WZDNCRFJ3PR	A, B, C
32.	Microsoft.WindowsFeedbackHub	9NBLGGH4R32N	A, B, C
33.	Microsoft.WindowsMaps	9WZDNCRDTBVB	A, B, C
34.	Microsoft.BingNews	9WZDNCRFHVFW	A, B, C
35.	Microsoft.DesktopAppInstaller	9NBLGGH4NNS1	A, B, C
36.	Microsoft.WindowsCamera	9WZDNCRFJBBG	A, B, C
37.	Microsoft.Getstarted	9WZDNCRDTBJJ	A, B, C

38.	Microsoft.Cortana	9NFFX4SZZ23L	A, B, C
39.	Microsoft.BingWeather	9WZDNCRFJ3Q2	A, B, C
40.	Microsoft.GetHelp	9PKDZBMV1H3T	A, B, C
41.	Microsoft.MicrosoftOfficeHub	9WZDNCRD29V9	A, B, C
42.	Microsoft.People	9NBLGGH10PG8	A, B, C
43.	Microsoft.StorePurchaseApp	9NBLGGH4LS1F	A, B, C
44.	Microsoft.Todos	9NBLGGH5R558	A, B, C
45.	Microsoft.Windows.Photos	9WZDNCRFJBH4	A, B, C
46.	Microsoft.WindowsCalculator	9WZDNCRFHVN5	A, B, C
47.	Microsoft.Windows.CommunicationsApps	9WZDNCRFHVQM	A, B, C
48.	Microsoft.XboxGameOverlay	9NBLGGH537C2	A, B, C
49.	Microsoft.XboxIdentityProvider	9WZDNCRD1HKW	A, B, C
50.	Microsoft.XboxSpeechToTextOverlay	9P086NHDNB9W	A, B, C
51.	Microsoft.YourPhone	9NMPJ99VJBWV	A, B, C
52.	MicrosoftCorporationII.QuickAssist	9P7BP5VNWKX5	A, B, C
53.	MicrosoftWindows.Client.WebExperience	9MSSGKG348SP	A, B, C

5.3. 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.

5.3.1. View

```
$custom_array | Out-GridView
```

5.3.2. 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
```

5.4. Remove all installed pre-applications

- 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.11/23H2/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

    }

}
```

5.5. 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.

5.6. 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:

5.6.1. 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**

5.6.2. Manual download

5.6.2.1. Region

Region	Products ID	Region	Products ID
en-US	C9PDSCC711RVF	af-za	9PDW16B5HMXR
am-et	9NGL4R61W3PL	ar-sa	9N4S78P86PKX
as-in	9NTJLXMX35J	az-latn-az	9P5TFKZH5K8
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	9NTHCXCSJDH	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>

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

5.6.2.2.1. Search keywords: <https://www.microsoft.com/store/productId/9NRMNT6GMZ70>

5.6.2.2.2. Search 22621 content in the web page, search results:
[Microsoft.LanguageExperiencePackzh-CN_22621.*.neutral__8wekyb3d8bbwe.appx](#)

5.6.2.2.3. After downloading, save it to the
[D:\OS_11_Custom\Install\Install\InBox.Apps](#) directory and rename it:
[LanguageExperiencePack.zh-cn.Neutral.Appx](#)

5.6.3. 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
```

5.6.4. 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.

5.6.4.1. 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".

5.6.4.1.1. View

```
$custom_array | Out-GridView
```

5.6.4.1.2. 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
```

5.6.4.2. View available language settings

```
Dism /Image:"D:\OS_11_Custom\Install\Install\Mount" /Get-Intl
```

5.7. InBox Apps: Install

5.7.1. Mount or decompress the InBox Apps installation file

Mount [22621.1.220506-1250.ni_release_amd64fre_InboxApps.iso](#) or extract to any location;

5.7.2. 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
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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"; "CoreN"; "CoreSingleLanguage"; )

            Apps = @(

                "Microsoft.UI.Xaml.2.3"; "Microsoft.UI.Xaml.2.4"; "Microsoft.UI.Xaml.2.7"; "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";

                "Microsoft.HEIFImageExtension"; "Microsoft.HEVCVideoExtension"; "Microsoft.SecHealthUI";
                "Microsoft.VP9VideoExtensions"; "Microsoft.WebpImageExtension"; "Microsoft.WindowsStore";
                "Microsoft.GamingApp"; "Microsoft.MicrosoftStickyNotes"; "Microsoft.Paint";
                "Microsoft.PowerAutomateDesktop";

                "Microsoft.ScreenSketch"; "Microsoft.WindowsNotepad"; "Microsoft.WindowsTerminal";
                "Clipchamp.Clipchamp"; "Microsoft.MicrosoftSolitaireCollection"; "Microsoft.WindowsAlarms";
                "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsMaps"; "Microsoft.ZuneMusic"; "Microsoft.BingNews";

                "Microsoft.BingWeather"; "Microsoft.DesktopAppInstaller"; "Microsoft.WindowsCamera";
                "Microsoft.Getstarted"; "Microsoft.Cortana"; "Microsoft.GetHelp"; "Microsoft.MicrosoftOfficeHub";
                "Microsoft.People"; "Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.WebMediaExtensions";

                "Microsoft.Windows.Photos"; "Microsoft.WindowsCalculator"; "Microsoft.windowscommunicationsapps";
                "Microsoft.WindowsSoundRecorder"; "Microsoft.Xbox.TCUI"; "Microsoft.XboxGameOverlay";
                "Microsoft.XboxGamingOverlay"; "Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay";
```

```

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

    )

}

@{

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

    Apps = @(

        "Microsoft.UI.Xaml.2.3"; "Microsoft.UI.Xaml.2.4"; "Microsoft.UI.Xaml.2.7"; "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";

        "Microsoft.HEIFImageExtension"; "Microsoft.HEVCVideoExtension"; "Microsoft.SecHealthUI";
"Microsoft.VP9VideoExtensions"; "Microsoft.WebpImageExtension"; "Microsoft.WindowsStore";
"Microsoft.GamingApp"; "Microsoft.MicrosoftStickyNotes"; "Microsoft.Paint";
"Microsoft.PowerAutomateDesktop";

        "Microsoft.ScreenSketch"; "Microsoft.WindowsNotepad"; "Microsoft.WindowsTerminal";
"Clipchamp.Clipchamp"; "Microsoft.MicrosoftSolitaireCollection"; "Microsoft.WindowsAlarms";
"Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsMaps"; "Microsoft.ZuneMusic"; "Microsoft.BingNews";

        "Microsoft.BingWeather"; "Microsoft.DesktopAppInstaller"; "Microsoft.WindowsCamera";
"Microsoft.Getstarted"; "Microsoft.Cortana"; "Microsoft.GetHelp"; "Microsoft.MicrosoftOfficeHub";
"Microsoft.People"; "Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.WebMediaExtensions";

        "Microsoft.Windows.Photos"; "Microsoft.WindowsCalculator"; "Microsoft.windowscommunicationsapps";
"Microsoft.WindowsSoundRecorder"; "Microsoft.Xbox.TCUI"; "Microsoft.XboxGameOverlay";
"Microsoft.XboxGamingOverlay"; "Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay";

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

    )

}

@{

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

    Apps = @(

        "Microsoft.UI.Xaml.2.3"; "Microsoft.UI.Xaml.2.4"; "Microsoft.UI.Xaml.2.7"; "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";

        "Microsoft.SecHealthUI"; "Microsoft.WindowsStore"; "Microsoft.MicrosoftStickyNotes"; "Microsoft.Paint";
"Microsoft.PowerAutomateDesktop"; "Microsoft.ScreenSketch"; "Microsoft.WindowsNotepad";
"Microsoft.WindowsTerminal"; "Clipchamp.Clipchamp"; "Microsoft.MicrosoftSolitaireCollection";

        "Microsoft.WindowsAlarms"; "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsMaps";
"Microsoft.BingNews"; "Microsoft.BingWeather"; "Microsoft.DesktopAppInstaller"; "Microsoft.WindowsCamera";
"Microsoft.Getstarted"; "Microsoft.Cortana"; "Microsoft.GetHelp"; "Microsoft.MicrosoftOfficeHub";

```



```

        "Microsoft.People"; "Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.Windows.Photos";
"Microsoft.WindowsCalculator"; "Microsoft.windowscommunicationsapps"; "Microsoft.XboxGameOverlay";
"Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay"; "Microsoft.YourPhone";

        "MicrosoftCorporationII.QuickAssist"; "MicrosoftWindows.Client.WebExperience";

    )

}

)

Rule = @(

    @{ Name="Microsoft.UI.Xaml.2.3"; Match="UI.Xaml*{ARCHC}*2.3"; License="UI.Xaml*{ARCHC}*2.3";
Region="All"; Dependencies=@(); }

    @{ Name="Microsoft.UI.Xaml.2.4"; Match="UI.Xaml*{ARCHTag}*2.4"; License="UI.Xaml*{ARCHTag}*2.4";
Region="All"; Dependencies=@(); }

    @{ Name="Microsoft.UI.Xaml.2.7"; Match="UI.Xaml*{ARCHTag}*2.7"; License="UI.Xaml*{ARCHTag}*2.7";
Region="All"; Dependencies=@(); }

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

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

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

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

    @{ Name="Microsoft.HEVCVideoExtension"; Match="HEVCVideoExtension*{ARCHC}";
License="HEVCVideoExtension*{ARCHC}*xml"; Region="All"; Dependencies=@("Microsoft.VCLibs.140.00"); }

    @{ Name="Microsoft.SecHealthUI"; Match="SecHealthUI*{ARCHC}"; License="SecHealthUI*{ARCHC}";
Region="All"; Dependencies=@("Microsoft.UI.Xaml.2.4", "Microsoft.VCLibs.140.00"); }

    @{ Name="Microsoft.VP9VideoExtensions"; Match="VP9VideoExtensions*{ARCHC}";
License="VP9VideoExtensions*{ARCHC}"; Region="All"; Dependencies=@("Microsoft.VCLibs.140.00"); }

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

    @{ Name="Microsoft.WindowsStore"; Match="WindowsStore"; License="WindowsStore"; Region="All";
Dependencies=@("Microsoft.UI.Xaml.2.7", "Microsoft.NET.Native.Framework.2.2", "Microsoft.NET.Native.Runtime
.2.2", "Microsoft.VCLibs.140.00"); }

    @{ Name="Microsoft.GamingApp"; Match="GamingApp"; License="GamingApp"; Region="All";
Dependencies=@("Microsoft.UI.Xaml.2.3", "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.Paint"; Match="Paint"; License="Paint"; Region="All";  
Dependencies=@("Microsoft.VCLibs.140.00","Microsoft.VCLibs.140.00.UWPDesktop","Microsoft.UI.Xaml.2.7"); }
```

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

```
@{ Name="Microsoft.ScreenSketch"; Match="ScreenSketch"; License="ScreenSketch"; Region="All";  
Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.VCLibs.140.00"); }
```

```
@{ Name="Microsoft.WindowsNotepad"; Match="WindowsNotepad"; License="WindowsNotepad";  
Region="All";  
Dependencies=@("Microsoft.VCLibs.140.00","Microsoft.VCLibs.140.00.UWPDesktop","Microsoft.UI.Xaml.2.7"); }
```

```
@{ Name="Microsoft.WindowsTerminal"; Match="WindowsTerminal"; License="WindowsTerminal";  
Region="All"; Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.VCLibs.140.00.UWPDesktop"); }
```

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

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

```
@{ 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"); }
```

```
@{ Name="Microsoft.WindowsFeedbackHub"; Match="WindowsFeedbackHub";  
License="WindowsFeedbackHub"; Region="All";  
Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime  
.2.2","Microsoft.VCLibs.140.00"); }
```

```
@{ Name="Microsoft.WindowsMaps"; Match="WindowsMaps"; License="WindowsMaps"; Region="All";  
Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime  
.2.2","Microsoft.VCLibs.140.00"); }
```

```
@{ Name="Microsoft.ZuneMusic"; Match="ZuneMusic"; License="ZuneMusic"; Region="All";  
Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime  
.2.2","Microsoft.VCLibs.140.00"); }
```

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

```
@{ Name="Microsoft.BingNews"; Match="BingNews"; License="BingNews"; Region="All";  
Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime  
.2.2","Microsoft.VCLibs.140.00"); }
```

```
@{ Name="Microsoft.BingWeather"; Match="BingWeather"; License="BingWeather"; Region="All";  
Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime  
.2.2","Microsoft.VCLibs.140.00"); }
```

```
@{ Name="Microsoft.DesktopAppInstaller"; Match="DesktopAppInstaller"; License="DesktopAppInstaller";  
Region="All"; Dependencies=@("Microsoft.UI.Xaml.2.7","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.Getstarted"; Match="Getstarted"; License="Getstarted"; Region="All";
Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime
.2.2","Microsoft.VCLibs.140.00"); }

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

    @{ Name="Microsoft.GetHelp"; Match="GetHelp"; License="GetHelp"; Region="All";
Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime
.2.2","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.People"; Match="People"; License="People"; Region="All";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.
140.00"); }

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

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

    @{ Name="Microsoft.Windows.Photos"; Match="Windows.Photos"; License="Windows.Photos"; Region="All";
Dependencies=@("Microsoft.UI.Xaml.2.4","Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime
.2.2","Microsoft.VCLibs.140.00"); }

    @{ 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.windowscommunicationsapps"; Match="WindowsCommunicationsApps";
License="WindowsCommunicationsApps"; Region="All";
Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.VCLibs.140.00"); }

    @{ Name="Microsoft.WindowsSoundRecorder"; Match="WindowsSoundRecorder";
License="WindowsSoundRecorder"; Region="All";
Dependencies=@("Microsoft.UI.Xaml.2.3","Microsoft.VCLibs.140.00"); }

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

        @{ Name="Microsoft.ZuneVideo"; Match="ZuneVideo"; License="ZuneVideo"; Region="All";
Dependencies=@("Microsoft.UI.Xaml.2.7","Microsoft.VCLibs.140.00"); }

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

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

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

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

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

    )

}

$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 }

}

```

5.8. Offline 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](#)

- https://github.com/likeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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

        }

    }

}

```

5.9. 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.

```

Dism /Image:"D:\OS_11_Custom\Install\Install\Mount" /Optimize-ProvisionedAppxPackages

```

6. Cumulative updates

- When upgrading different versions or old versions to the latest version, you need to add the "Function Enablement Package" first before adding the latest cumulative update;

- After adding a language pack, you can install the same cumulative update as the initial version to resolve a known issue where the "Components: 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.

6.1. Exclusive feature enablement package

- Learn: [KB5027397: Feature update to Windows 11, version 23H2 by using an enablement package](#)
- When making Windows 11 versions 22H2 and 23H2, you need to install the "Feature Enablement Package" in advance before you can install the cumulative update of 22631.*. After downloading, save it to: [D:\OS_11_Custom\Install\Install\Update](#). Select the download according to the architecture:

6.1.1. x64, default

- Direct download

https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/caa3ff4a-6420-4341-aeae-33b2d7f463be/public/windows11.0-kb5027397-x64_3a9c368e239bb928c32a790cf1663338d2cad472.msu

- Add to

```
$KBPath = "D:\OS_11_Custom\Install\Install\Update\windows11.0-kb5027397-x64_3a9c368e239bb928c32a790cf1663338d2cad472.msu"
```

```
Add-WindowsPackage -Path "D:\OS_11_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.1.2. Arm64

- Direct download

https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/c29dd4ea-7f6a-4636-a991-29ba8ae70658/public/windows11.0-kb5027397-arm64_bacb74fba9077a5b7ae2f74a3ebb0b506f9708f3.msu

- Add to

```
$KBPath = "D:\OS_11_Custom\Install\Install\Update\windows11.0-kb5027397-arm64_bacb74fba9077a5b7ae2f74a3ebb0b506f9708f3.msu"
```

```
Add-WindowsPackage -Path "D:\OS_11_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.2. Initial version

- Microsoft has officially provided an exclusive feature activation package for the initial version of Windows 11 23H2 22631.2428. File name: [X23-59425_Windows11-23H2-FeatureEnablement.zip](#). The compressed package contains the following files: [KB5031354](#), [KB5027397](#).
- First add the "Exclusive Function Enablement Package" and then add the "Initial Version", go to the download page: <https://www.catalog.update.microsoft.com/Search.aspx?q=KB5031354>, and save it to: [D:\OS_11_Custom\Install\Install\Update](#), or download through direct connection, select download according to the architecture:

6.2.1. x64, default

- Direct download

https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/885dcacb-fb29-4536-88f6-4fd4792a4a65/public/windows11.0-kb5031354-x64_3391407e62562e0e48a437c5c7115667305d788c.msu

- Add to

```
$KBPath = "D:\OS_11_Custom\Install\Install\Update\windows11.0-kb5031354-x64_3391407e62562e0e48a437c5c7115667305d788c.msu"
```

```
Add-WindowsPackage -Path "D:\OS_11_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.2.2. Arm64

- Direct download

https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/e3e1d074-cfde-4c2d-ba0c-f6887db3b166/public/windows11.0-kb5031354-arm64_3848ce9f925f15d07ad1dd06fe976e6719634653.msu

- Add to

```
$KBPath = "D:\OS_11_Custom\Install\Install\Update\windows11.0-kb5031354-arm64_3848ce9f925f15d07ad1dd06fe976e6719634653.msu"
```

```
Add-WindowsPackage -Path "D:\OS_11_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.3. Other versions

- Before installation, download the exclusive function activation package [KB5027397](#) and add it first, and then add other versions in sequence.
- Check "[Windows 11 version information](#)", for example, download cumulative update: [KB5035853](#), Version: [22631.3296](#), go to the download page: <https://www.catalog.update.microsoft.com/Search.aspx?q=KB5035853>, download and save to: [D:\OS_11_Custom\Install\Install\Update](#), or download through direct connection, select download according to the architecture:

6.3.1. x64, default

- Direct download

https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/594b22d5-84c3-4665-bdc7-3167c91759b9/public/windows11.0-kb5035853-x64_8ca1a9a646dbe25c071a8057f249633a61929efa.msu

- Add to

```
$KBPath = "D:\OS_11_Custom\Install\Install\Update\windows11.0-kb5035853-x64_8ca1a9a646dbe25c071a8057f249633a61929efa.msu"
```

```
Add-WindowsPackage -Path "D:\OS_11_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.3.2. Arm64

- Direct download

https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/72ba502a-7982-471c-aad2-355efddbff61/public/windows11.0-kb5035853-arm64_076c009bca80cfa85272bd50d65951c2f663fcc5.msu

- Add to

```
$KBPath = "D:\OS_11_Custom\Install\Install\Update\windows11.0-kb5035853-arm64_076c009bca80cfa85272bd50d65951c2f663fcc5.msu"
```

```
Add-WindowsPackage -Path "D:\OS_11_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.4. 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
```

6.4.1. 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

    }

}
```

7. Drive

8. 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.

9. 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
```

10. 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
```

11. Save image: Install.wim

```
Save-WindowsImage -Path "D:\OS_11_Custom\Install\Install\Mount"
```

12. 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.

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

13.1. Obtain WimLib

After going to the official website of <https://wimlib.net>, select a different version: [arm64](#), [x64](#), [x86](#), and extract it to: [D:Wimlib](#) after downloading

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

13.2.1. Extract the WinRE.wim file Install.wim from Install.wim

- Install.WinRE.Extract.ps1
 - [\Expand\Install\Install.WinRE.Extract.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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
```

13.2.2. 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](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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

}

```

14. Rebuilding Install.wim reduces file size

- Install.Rebuild.wim.ps1
 - [\Expand\Install\Install.Rebuild.wim.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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

}

```

15. 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.

15.1. Splitting and merging

15.1.1. 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](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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

}
```

15.1.2. Merge

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

- Install.Merging.ps1
 - [\Expand\Install\Install.Merging.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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

}

```

15.2. Solid compressed ESD format and WIM format conversion

15.2.1. 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](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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

}

```

15.2.2. Convert compressed files to WIM file format

- Install.Convert.ps1
 - [\Expand\Install\Install.Convert.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/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

```



```
}
```

III Custom encapsulation boot.wim

1. 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 }
```

2. Specify the path to mount Boot.wim

```
New-Item -Path "D:\OS_11_Custom\Boot\Boot\Mount" -ItemType directory
```

3. 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"
```

4. 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.

4.1. Language pack: Add

- Boot.Instl.lang.ps1
 - [\Expand\Boot\Boot.Instl.lang.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging/tutorial/OS.11/23H2/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_List = @(
```

```
    @{ Match = "*WinPE*Setup*Client*Package*"; File = "WINPE-SETUP-CLIENT_zh-CN.CAB"; }
```

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

)

ForEach ($Rule in $Language_List) {

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

    ForEach ($Component in $Initl_install_Language_Component) {

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

            Write-host "  Component name: " -NoNewline

            Write-host $Component -ForegroundColor Green

            Write-host "  Language pack file: " -NoNewline

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

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

            try {

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

                Write-host "Finish" -ForegroundColor Green

            } catch {

                Write-host "Failed" -ForegroundColor Red

            }

            break

        }

    }

}

```

4.2. Offline image language: change

4.2.1. 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"
```

4.2.2. View available language settings

```
Dism /Image:"D:\OS_11_Custom\Boot\Boot\Mount" /Get-Intl
```

4.3. Components: All packages installed in the image

4.3.1. View

```
Get-WindowsPackage -Path "D:\OS_11_Custom\Boot\Boot\Mount" | Out-GridView
```

4.3.2. 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
```

4.4. Language: Repair

4.4.1. Extract

Open: D:\OS_11_Custom\Install\Install\Language\Add\zh-CN\Microsoft-Windows-Client-Language-Pack_x64_zh-CN.cab, enter the directory: Setup\sources\zh-cn\cli, and copy the following files to the deskto:

4.4.1.1. arunres.dll.mui

4.4.1.2. spwizres.dll.mui

4.4.1.3. w32uires.dll.mui

4.4.2. Copy

Copy the extracted files to: D:\OS_11_Custom\Boot\Boot\Mount\sources\zh-CN

4.5. 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
```

4.6. 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.

4.6.1. 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"
```

4.6.2. 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"
```

5. Cumulative updates

To prepare the cumulative updates file available, change the example file name: `KB_Boot.cab`

5.1. Add

```
$KBPath = "D:\OS_11_Custom\Boot\Boot\Update\KB_Boot.cab"
```

```
Add-WindowsPackage -Path "D:\OS_11_Custom\Boot\Boot\Mount" -PackagePath $KBPath
```

5.2. Delete

```
$KBPath = "D:\OS_11_Custom\Boot\Boot\Update\KB_Boot.cab"
```

```
Remove-WindowsPackage -Path "D:\OS_11_Custom\Boot\Boot\Mount" -PackagePath $KBPath
```

5.3. 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
```

5.3.1. 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
```

```
    }
```

```
}
```

6. Drive

7. Other

7.1. Bypass TPM check during installation

- `Boot.Bypass.TPM.ps1`
 - `\Expand\Boot\Boot.Bypass.TPM.ps1`
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging/tutorial/OS.11/23H2/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
```

8. Save image: Boot.wim

```
Save-WindowsImage -Path "D:\OS_11_Custom\Boot\Boot\Mount"
```

9. 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
```

IV 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**", Intervene in the installation process by leaving it unattended.

1. Add method

1.1. Add to ISO installation media

1.1.1. Unattended

1.1.1.1. 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

1.1.1.2. 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

1.1.1.3. 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

1.1.1.3.1. Create \$OEM\$ path

New-Item -Path "D:\OS_11\sources\`\$OEM\$\`\$\$Panther" -ItemType Directory

1.1.1.3.2. 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

1.1.2. Deployment engine: add

Add "Automatically add installed languages for Windows systems" to D:\OS_11\sources\\$\OEM\$\\$1Yi\Engine in the directory.

1.1.2.1. Deployment Engine: Copy

Copy D:\Multilingual\Engine to D:\OS_11\Sources\\$\OEM\$\\$1Yi\Engine

Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS_11\sources\`\$OEM\$\`\$1Yi\Engine" -Recurse -Force

1.1.2.2. 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

}

```

1.2. Add to mounted

Through "[Custom encapsulation: Install.wim](#)", execute "[Start mounting Install.wim](#)" and mount to:

[D:\OS_11_Custom\Install\Install\Mount](#)

1.2.1. 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
```

1.2.2. Deployment engine: add

Add "Automatically add languages installed on Windows systems" to the D:\OS_11_Custom\Install\Install\Mount\Yi\Engine directory.

1.2.2.1. 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
```

1.2.2.2. 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

}
```


2. Deployment Engine: Advanced

2.1. Deployment engine: adding process

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

2.2. 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`

2.2.1. 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.

2.2.1.1. Multi-language

```
<UILanguage>%OSDUILanguage%</UILanguage>

<InputLocale>%OSDInputLocale%</InputLocale>

<SystemLocale>%OSDSysLocale%</SystemLocale>

<UILanguage>%OSDUILanguage%</UILanguage>

<UILanguageFallback>%OSDUILanguageFallback%</UILanguageFallback>

<UserLocale>%OSDUserLocale%</UserLocale>
```

2.2.1.2. 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>
```

2.2.2. 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.

2.2.2.1. 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>
```

2.2.2.2. Custom user

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

2.2.2.2.1. Delete

Username: Removed from start <UserAccounts> to </UserAccounts>
Autologin: Remove from start <AutoLogon> to </AutoLogon>

2.2.2.2.2. Replace

From the beginning <OOBE> to </OOBE>

```
<OOBE>

<ProtectYourPC>3</ProtectYourPC>

<HideEULAPage>true</HideEULAPage>

<HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>

</OOBE>
```

D. ISO

II Generate ISO

1. Download OScding

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

1.1. x64

https://github.com/ilikeyi/Solutions/raw/refs/heads/main/_Encapsulation/Modules/AIO/Oscding/amd64/oscding.exe

1.2. x86

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

1.3. arm64

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

2. Use the oscding command line to generate an ISO file and save it to: **D:\OS_11.iso**

- ISO.ps1
 - [\Expand\ISO.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/23H2/Expand/ISO.ps1

- Copy the code

```
$Oscding = "D:\Oscding.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 $Oscding -ArgumentList $Arguments -wait -newwindow
```

III 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.

Chapter 2 Common problem

II 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
```

III Fix the problem of abnormal mounting

1. View mounted

```
Get-WindowsImage -Mounted
```

2. Delete the DISM mount record saved in the registry

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

3. Delete all resources associated with the corrupted mounted image

```
Clear-WindowsCorruptMountPoint
```

```
Dism /cleanup-wim
```

IV 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

1.1. 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
```

```
}  
  
}  
  
}  
  
}
```

1.2. Manual deletion

1.2.1. 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

1.2.2. 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

1.2.3. 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.

2. Disk cleanup

Run `cleanmgr`, selecting the disks and types to clean.

Chapter 3	Known issues
	None yet



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, ilikeyi@outlook.com

Document version: 1.7

Translation: Chinese to English version

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

Document last updated: 11 / 2024

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