Windows 11 22H2

Different system versions have different packaging methods. The packaging process includes: "Language pack: add, associate, delete", "Driver: add, delete", "Cumulative update: add, delete", "InBox Appx: 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?

1. [**TABLE OF CONTENTS**](#_top)
2. [Deployment image](#_Custom_deployment_image)
   * 1. [Prerequisites](#_PREREQUISITES)
     2. [Language Pack: Extraction](#_Language_Pack:_Extraction)
     3. [Customize the deployment image](#_INSTALL_LANGUAGE_PACK)
     4. [ISO](#_4、生成_ISO)
3. Common problem
4. [Known issues](#_Known_issues)
5. [Components: All packages installed in the image, Comparison](#_Components:_Comparison)

## **[deployment image](#_top)**

### **[Prerequisites](#_Custom_deployment_image)**

#### [**ISO tools**](#_PREREQUISITES)

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

#### **[Requirements](#_PREREQUISITES)**

Learn "Attachment: OS" and view the "Windows 11" item.

##### [**System installation package**](#_Get_related_packages)

* 1. Prepare: en-us\_windows\_11\_business\_editions\_version\_22h2\_x64\_dvd\_17a08ce3.iso
  2. Unzip to: D:\en-us\_windows\_11\_business\_editions\_version\_22h2\_x64\_dvd\_17a08c
  3. After decompression is complete, change the directory en-us\_windows\_11\_business\_editions\_version\_22h2\_x64\_dvd\_17a08c change to D:\OS\_11
  4. All scripts and all paths are set to D:\OS\_11 as the image source.

##### [**Language pack**](#_Get_related_packages)

###### **Learn**

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

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

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

In "[Language Pack: Extraction](#_Language_Pack:_Extraction_1)", the automatic recognition function has been added, and you can understand the functions: Function Match\_Required\_Fonts

* + - 1. **Regional association**

What are regional connections?

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

**Known regional associations:**

* + - * 1. Region: zh-TW , Optional associated areas: zh-HK
      1. **Other region-specific requirements**

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

* + - * 1. Region: zh-TW, Package: Microsoft-Windows-InternationalFeatures-Taiwan-Package~31bf3856ad364e35~amd64~~.cab

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

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

**THERE IS CONTROVERSY:**

* During the test, it was found that this package was not installed in the original image of Microsoft's official original Windows 10 and Windows 11. However, there are known issues in the recommended items. In the end, it is consistent with the official Microsoft original version, and the packager can freely choose to install it. or not.

###### **Language pack: Download**

[22621.1.220506-1250.ni\_release\_amd64fre\_CLIENT\_LOF\_PACKAGES\_OEM.iso](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)

###### **Language pack: Fixed**

Select any website and open:

* <https://uupdump.net>
* <https://uup.ee>
  + 1. After opening, search for keywords: 22621.382, select from the search results: Windows 11, version 22H2 (22621.382) amd64
    2. After opening, select "All files";
    3. Search the green part in the "All Files" page and download:
       1. Applies to: Install.wim（ 1 item ）
          1. Media
       2. Applies to: WinRE.wim, none yet
       3. Applies to: Boot.wim, none yet
    4. After downloading all the files, scroll to the bottom of the page, download and run "Generate Rename Script (Windows)"
    5. 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;

##### [**InBox Apps**](#_Get_related_packages)

* 1. Download: [22621.1778.230511-2102.ni\_release\_svc\_prod3\_amd64fre\_InboxApps.iso](https://software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/22621.1778.230511-2102.ni_release_svc_prod3_amd64fre_InboxApps.iso)
  2. Download: [22621.1.220506-1250.ni\_release\_amd64fre\_InboxApps.iso](https://software-static.download.prss.microsoft.com/dbazure/988969d5-f34g-4e03-ac9d-1f9786c66749/22621.1.220506-1250.ni_release_amd64fre_InboxApps.iso) finally, extract the following files and save them to the desktop:
     1. Microsoft.HEVCVideoExtension\_8wekyb3d8bbwe.x64.appx
     2. Microsoft.HEVCVideoExtension\_8wekyb3d8bbwe.x64.xml
  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;

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

Set-ExecutionPolicy -ExecutionPolicy Bypass -Force

1. In this article, PS command line, green part, please copy it, paste it into the "Terminal" dialog box, press Enter and start running;
2. When there is .ps1, right-click the file and select Run with PowerShell, or copy the path and paste it into Terminal to run.

### **[Language Pack: Extraction](#_Custom_deployment_image)**

1. [**Prepare the language pack**](#_2、提取所需语言包)

Mount 22621.1.220506-1250.ni\_release\_amd64fre\_CLIENT\_LOF\_PACKAGES\_OEM.iso or unzipped to any location;

1. **Extract language pack scheme**

##### **Add**

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

##### **Delete**

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

1. [**Execute the extract command**](#_2、提取所需语言包)

* Auto = automatically search all local disks, default;
* Customize the path, for example, specify the E drive: $ISO = "E:\"
* Extract.ps1
  + [\Expand\Extract.ps1](Expand/Extract.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/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 }

### **[Customize the deployment image](#_Custom_deployment_image)**

#### **[Custom deployment image: Install.wim](#_INSTALL_LANGUAGE_PACK)**

##### [**View Install.wim details**](#_3.1__)

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

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

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

**LOOP OPERATING AREA, START,**

##### **[Specify the path to mount Install.wim](#_3.1__)**

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

##### **[Start mounting Install.wim](#_3.1__)**

Default index number: 1

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

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

###### **[Custom deployment image: WinRE.wim](#_3.1__)**

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](#_如何批量替换_Install.wim_里的所有索引号里的)".
  + 1. [**View WinRE.wim details**](#_3.1.2.1__)

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

$ViewFile = "D:\OS\_11\_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

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

* + 1. [**Specify the path to mount WinRE.wim**](#_3.1.2.1__)

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

* + 1. [**Start mounting WinRE.wim**](#_3.1.2.1__)

Default index number: 1

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

* + 1. **[Language pack](#_3.1.2.1__)**

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.

* + - 1. **[Language pack: add](#_Language_pack)**
* WinRE.Instl.lang.ps1
  + [\Expand\Install\WinRE\WinRE.Instl.lang.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\WinRE\WinRE.Instl.lang.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/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

}

}

}

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

Region: zh-CN

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

* + - * 1. **View available language settings**

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

* + - 1. [**Language packs: Removed, optional**](#_执行安装命令后安装语言包到：WinRE)
* After you add languages, if you want to deploy to a non-English locale, you can save space by removing the English language component. When you remove a language, uninstall the language components in the reverse order in which you added them.
* After adding the Chinese, delete "English - United States" in reverse, the Region tag: en-US, you need to extract the language pack in advance
* WinRE.Del.Specified.lang.Tag.ps1
  + [\Expand\Install\WinRE\WinRE.Del.Specified.lang.Tag.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\WinRE\WinRE.Del.Specified.lang.Tag.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Expand/Install/WinRE/WinRE.Del.Specified.lang.Tag.ps1>
* Copy the code

$Lang = "en-US"

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

$Sources = "D:\OS\_11\_Custom\Install\WinRE\Language\Del\en-US"

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

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

$Language = @(

@{ Match = "\*windowsupdate\*"; File = "winpe-windowsupdate\_$($Lang).cab"; }

@{ Match = "\*appxdeployment\*"; File = "winpe-appxdeployment\_$($Lang).cab"; }

@{ Match = "\*hta\*"; File = "winpe-hta\_$($Lang).cab"; }

@{ Match = "\*opcservices\*"; File = "winpe-opcservices\_$($Lang).cab"; }

@{ Match = "\*rejuv\*"; File = "winpe-rejuv\_$($Lang).cab"; }

@{ Match = "\*WiFi\*"; File = "winpe-wifi\_$($Lang).cab"; }

@{ Match = "\*StorageWMI\*"; File = "winpe-storagewmi\_$($Lang).cab"; }

@{ Match = "\*WinPE-AppxPackaging\*"; File = "winpe-appxpackaging\_$($Lang).cab"; }

@{ Match = "\*-WMI-Package\*"; File = "winpe-wmi\_$($Lang).cab"; }

@{ Match = "\*wds-tools\*"; File = "winpe-wds-tools\_$($Lang).cab"; }

@{ Match = "\*srt\*"; File = "winpe-srt\_$($Lang).cab"; }

@{ Match = "\*srh\*"; File = "winpe-srh\_$($Lang).cab"; }

@{ Match = "\*Speech-TTS\*"; File = "winpe-speech-tts\_$($Lang).cab"; }

@{ Match = "\*scripting\*"; File = "winpe-scripting\_$($Lang).cab"; }

@{ Match = "\*Narrator\*"; File = "winpe-narrator\_$($Lang).cab"; }

@{ Match = "\*EnhancedStorage\*"; File = "winpe-enhancedstorage\_$($Lang).cab"; }

@{ Match = "\*AudioDrivers\*"; File = "winpe-audiodrivers\_$($Lang).cab"; }

@{ Match = "\*AudioCore\*"; File = "winpe-audiocore\_$($Lang).cab"; }

@{ Match = "\*ATBroker\*"; File = "winpe-atbroker\_$($Lang).cab"; }

@{ Match = "\*SecureStartup\*"; File = "winpe-securestartup\_$($Lang).cab"; }

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

)

ForEach ($Rule in $Language) {

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

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

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

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 " Deleting ".PadRight(22) -NoNewline

try {

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

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

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

}

break

}

}

}

$InitlClearLanguagePackage = @()

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

if ($\_.PackageName -like "\*$($Lang)\*") {

$InitlClearLanguagePackage += $\_.PackageName

}

}

if ($InitlClearLanguagePackage.count -gt 0) {

ForEach ($item in $InitlClearLanguagePackage) {

Write-Host "`n $($item)" -ForegroundColor Green

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

try {

Remove-AppxProvisionedPackage -Path $Mount -PackageName $item -ErrorAction SilentlyContinue | Out-Null

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

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

}

}

}

* + - 1. **[Components: All packages installed in the image](#_Language_pack)**
         1. [**View**](#_Components_2)

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

* + - * 1. [**Export to csv**](#_Components_2)

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

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

Write-host $SaveTo -ForegroundColor Green

* + 1. **[Cumulative updates, optional](#_3.1.2.1__)**

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

* + - 1. **Add**

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

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

* + - 1. **Delete**

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

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

* + - 1. **Solid update, optional**

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

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

* + - * 1. [**Clean components after curing and updating, optional**](#_固化更新，可选项)

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

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

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

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

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

}

}

* + 1. [**Driver, optional**](#_3.1.2.1__)
    2. [**Save image**](#_3.1.2.1__)

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

* + 1. [**Unmount image**](#_3.1.2.1__)

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

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

* + 1. [**After rebuilding WinRE.wim, the file size can be reduced**](#_3.1.2.1__)
* WinRE.Rebuild.ps1
  + [\Install\WinRE\WinRE.Rebuild.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\WinRE\WinRE.Rebuild.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/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 Rebuild".PadRight(28) -NoNewline

Export-WindowsImage -SourceImagePath $Filename -SourceIndex $\_.ImageIndex -DestinationImagePath "$($FileName).New" -CompressionType max

Write-Host "Finish`n" -ForegroundColor Green

}

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

Remove-Item -Path $Filename

Move-Item -Path "$($FileName).New" -Destination $Filename

Write-Host "Finish" -ForegroundColor Green

} else {

Write-host "Failed" -ForegroundColor Red

}

* + 1. [**Backup WinRE.wim**](#_3.1.2.1__)
* WinRE.Backup.ps1
  + [\Expand\Install\WinRE\WinRE.Backup.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\WinRE\WinRE.Backup.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/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 -ea SilentlyContinue

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

* + 1. [**Replace WinRE.wim within the Install.wim image**](#_3.1.2.1__)
* After each mount Install.wim "[Replace WinRE.wim](#_替换_WinRE.wim)";
* Learn "[Get all index numbers of Install.wim and replace the old WinRE.wim](#_获取_Install.wim_所有索引号后并替换旧的)".

Process the files in the Install.wim image and end.

##### **[Language pack](#_3.1__)**

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.

###### **[Language pack: add](#_After_executing_the_1)**

* Install.Instl.lang.ps1
  + [\Expand\Install\Install.Instl.lang.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\Install.Instl.lang.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/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"

###### **Offline image language: change**

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

Region: zh-CN

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

* + 1. **View available language settings**

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

###### **[Language packs: Removed, optional](#_执行安装命令后安装语言包到：WinRE)**

* After you add languages, if you want to deploy to a non-English locale, you can save space by removing the English language component. When you remove a language, uninstall the language components in the reverse order in which you added them.
* After adding the Chinese, delete "English - United States" in reverse, the Region tag: en-US, you need to extract the language pack in advance
* Install.Del.Specified.lang.Tag.ps1
  + [\Expand\Install\Install.Del.Specified.lang.Tag.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\Install.Del.Specified.lang.Tag.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Expand/Install/Install/Install.Del.Specified.lang.Tag.ps1>
* Copy the code

$Lang = "en-US"

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

$Sources = "D:\OS\_11\_Custom\Install\Install\Language\Del\en-US"

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

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

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

}

$Language = @(

@{ Match = "\*WordPad\*wow64\*"; File = "Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

@{ Match = "\*WordPad\*amd64\*"; File = "Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~$($Lang)~.cab"; }

@{ Match = "\*WMIC\*FoD\*Package\*wow64\*"; File = "Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

@{ Match = "\*WMIC\*FoD\*Package\*amd64\*"; File = "Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

@{ Match = "\*StepsRecorder\*wow64\*"; File = "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

@{ Match = "\*StepsRecorder\*amd64\*"; File = "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~$($Lang)~.cab"; }

@{ Match = "\*Printing\*PMCPPC\*amd64\*"; File = "Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

@{ Match = "\*PowerShell\*wow64\*"; File = "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

@{ Match = "\*PowerShell\*amd64\*"; File = "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~$($Lang)~.cab"; }

@{ Match = "\*MediaPlayer\*wow64\*"; File = "Microsoft-Windows-MediaPlayer-Package-wow64-$($Lang).cab"; }

@{ Match = "\*MediaPlayer\*amd64\*"; File = "Microsoft-Windows-MediaPlayer-Package-AMD64-$($Lang).cab"; }

@{ Match = "\*Notepad\*wow64\*"; File = "Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

@{ Match = "\*Notepad\*amd64\*"; File = "Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

@{ Match = "\*InternetExplorer\*"; File = "Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

@{ Match = "\*TextToSpeech\*"; File = "Microsoft-Windows-LanguageFeatures-TextToSpeech-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

@{ Match = "\*LanguageFeatures-Speech\*"; File = "Microsoft-Windows-LanguageFeatures-Speech-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

@{ Match = "\*OCR\*"; File = "Microsoft-Windows-LanguageFeatures-OCR-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

@{ Match = "\*Handwriting\*"; File = "Microsoft-Windows-LanguageFeatures-Handwriting-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

@{ Match = "\*LanguageFeatures-Basic\*"; File = "Microsoft-Windows-LanguageFeatures-Basic-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

@{ Match = "\*Client-LanguagePack-Package\*"; File = "Microsoft-Windows-Client-Language-Pack\_x64\_$($Lang).cab"; }

)

ForEach ($Rule in $Language) {

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

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

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

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 " Deleting ".PadRight(22) -NoNewline

try {

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

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

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

}

break

}

}

}

$InitlClearLanguagePackage = @()

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

if ($\_.PackageName -like "\*$($Lang)\*") {

$InitlClearLanguagePackage += $\_.PackageName

}

}

if ($InitlClearLanguagePackage.count -gt 0) {

ForEach ($item in $InitlClearLanguagePackage) {

Write-Host "`n $($item)" -ForegroundColor Green

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

try {

Remove-AppxProvisionedPackage -Path $Mount -PackageName $item -ErrorAction SilentlyContinue | Out-Null

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

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

}

}

}

###### **Components: [All packages installed in the image](#_After_executing_the_1)**

* + 1. [**View**](#_Components_1)

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

* + 1. [**Export to csv**](#_Components_1)

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

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

Write-host $SaveTo -ForegroundColor Green

##### **[InBox Apps](#_3.1__)**

###### [**Learn "Attachment: inBox Apps"**](#_InBox_Apps)

Learn about pre-installed apps for different operating systems, app store connections.

###### **[InBox Apps: Installed](#_InBox_Apps)**

* + 1. [**View**](#_InBox_Apps:_Installed)

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

* + 1. [**Export to Csv**](#_InBox_Apps:_Installed)

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

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

Write-host $SaveTo -ForegroundColor Green

###### [**Remove all installed pre-applications**](#_InBox_Apps)

* Install.InBox.Appx.Clear.all.ps1
  + [\Expand\Install\Install.InBox.Appx.Clear.all.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\Install.InBox.Appx.Clear.all.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Expand/Install/Install.InBox.Appx.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

}

}

###### [**Region tag: Understanding**](#_InBox_Apps)

Before understanding zone 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 for 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 area tag.

###### [**Region tag: adding method**](#_InBox_Apps)

* + 1. Execute "[Language Pack:Add](#_Language_pack:_add)"
    2. Install “Local Language Experience Packages (LXPs)”

Microsoft officially provides the Local Language Experience Package (LXPS) installation file for Windows 10. It will no longer be provided for Windows 11. To obtain the offline installation package:

* + - 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.Appx, file format:LanguageExperiencePack.zh-CN.Neutral.Appx

* + - 1. **Manual download** 
         1. **View "Attachment: LXPs"**

Waiting to download Local Language Experience Packs (LXPs) for "Simplified Chinese - China", Region: zh-CN, application ID: 9NRMNT6GMZ70, Store link: https://www.microsoft.com/store/productId/9NRMNT6GMZ70

* + - * 1. **Open the website:** [**https://store.rg-adguard.net**](https://store.rg-adguard.net)

Search keywords:

https://www.microsoft.com/store/productId/9NRMNT6GMZ70

Search 22621 content in the web page, search results: Microsoft.LanguageExperiencePackzh-CN\_22621.\*. neutral\_\_8wekyb3d8bbwe.appx

After downloading, save it to the D:\OS\_11\_Custom\Install\Install\InBox.Appx directory and rename it: LanguageExperiencePack.zh-cn.Neutral.Appx

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

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

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

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

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

* + - * 1. **Export to Csv**

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

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

Write-host $SaveTo -ForegroundColor Green

###### **[InBox Apps: Install](#_InBox_Apps)**

* + 1. [**Mount or decompress the InBox Apps installation file**](#_InBox_Apps:_Install)

Mount 22621.1.220506-1250.ni\_release\_amd64fre\_InboxApps.iso or extract to any location;

* + 1. [**After executing the installation command, install InBox Apps to: Install.wim**](#_InBox_Apps:_Install)
* Auto = Automatically search all local disks, default;
* Custom path, e.g. specify F drive: $ISO = "F:\packages"
* Install.Inst.InBox.Appx.ps1
  + [\Expand\Install\Install.Inst.InBox.Appx.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\Install.Inst.InBox.Appx.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Expand/Install/Install.Inst.InBox.Appx.ps1>
* Copy the code

$ISO = "Auto"

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

try {

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

$Current\_Edition\_Version = (Get-WindowsEdition -Path $Mount).Edition

Write-Host $Current\_Edition\_Version -ForegroundColor Green

} catch {

Write-Host "Error" -ForegroundColor Red

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

return

}

$Pre\_Config\_Rules = @(

@{

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

Apps = @(

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

)

}

@{

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

Apps = @(

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

)

}

@{

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

Apps = @(

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

)

}

)

$Allow\_Install\_App = @()

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

if ($item.Name -contains $Current\_Edition\_Version) {

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

Write-host $Current\_Edition\_Version -ForegroundColor Green

$Allow\_Install\_App = $item.Apps

break

}

}

Write-host "`n The app to install ( $($Allow\_Install\_App.Count) item )" -ForegroundColor Yellow

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

ForEach ($item in $Allow\_Install\_App) {

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

}

$InBoxApps = @(

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

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

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

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

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

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

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

@{ Name = "Microsoft.WindowsStore"; File = "Microsoft.WindowsStore\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.WindowsStore\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.HEIFImageExtension"; File = "Microsoft.HEIFImageExtension\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.HEIFImageExtension\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.HEVCVideoExtension"; File = "Microsoft.HEVCVideoExtension\_8wekyb3d8bbwe.x64.appx"; License = "Microsoft.HEVCVideoExtension\_8wekyb3d8bbwe.x64.xml"; }

@{ Name = "Microsoft.SecHealthUI"; File = "Microsoft.SecHealthUI\_8wekyb3d8bbwe.x64.appx"; License = "Microsoft.SecHealthUI\_8wekyb3d8bbwe.x64.xml"; }

@{ Name = "Microsoft.VP9VideoExtensions"; File = "Microsoft.VP9VideoExtensions\_8wekyb3d8bbwe.x64.appx"; License = "Microsoft.VP9VideoExtensions\_8wekyb3d8bbwe.x64.xml"; }

@{ Name = "Microsoft.WebpImageExtension"; File = "Microsoft.WebpImageExtension\_8wekyb3d8bbwe.x64.appx"; License = "Microsoft.WebpImageExtension\_8wekyb3d8bbwe.x64.xml"; }

@{ Name = "Microsoft.GamingApp"; File = "Microsoft.GamingApp\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.GamingApp\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.Sticky.Notes"; File = "Microsoft.MicrosoftStickyNotes\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.MicrosoftStickyNotes\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.Paint"; File = "Microsoft.Paint\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.Paint\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.PowerAutomateDesktop"; File = "Microsoft.PowerAutomateDesktop\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.PowerAutomateDesktop\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.ScreenSketch"; File = "Microsoft.ScreenSketch\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.ScreenSketch\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.WindowsNotepad"; File = "Microsoft.WindowsNotepad\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.WindowsNotepad\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.WindowsTerminal"; File = "Microsoft.WindowsTerminal\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.WindowsTerminal\_8wekyb3d8bbwe.xml"; }

@{ Name = "Clipchamp.Clipchamp"; File = "Clipchamp.Clipchamp\_yxz26nhyzhsrt.msixbundle"; License = "Clipchamp.Clipchamp\_yxz26nhyzhsrt.xml"; }

@{ Name = "Microsoft.Solitaire.Collection"; File = "Microsoft.MicrosoftSolitaireCollection\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.MicrosoftSolitaireCollection\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.WindowsAlarms"; File = "Microsoft.WindowsAlarms\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.WindowsAlarms\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.WindowsFeedbackHub"; File = "Microsoft.WindowsFeedbackHub\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.WindowsFeedbackHub\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.WindowsMaps"; File = "Microsoft.WindowsMaps\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.WindowsMaps\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.ZuneMusic"; File = "Microsoft.ZuneMusic\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.ZuneMusic\_8wekyb3d8bbwe.xml"; }

@{ Name = "MicrosoftCorporationII.MicrosoftFamily"; File = "MicrosoftCorporationII.MicrosoftFamily\_8wekyb3d8bbwe.msixbundle"; License = "MicrosoftCorporationII.MicrosoftFamily\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.BingNews"; File = "Microsoft.BingNews\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.BingNews\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.DesktopAppInstaller"; File = "Microsoft.DesktopAppInstaller\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.DesktopAppInstaller\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.WindowsCamera"; File = "Microsoft.WindowsCamera\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.WindowsCamera\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.Getstarted"; File = "Microsoft.Getstarted\_8wekyb3d8bbwe.msixbundle"; License = "Microsoft.Getstarted\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.Cortana"; File = "Microsoft.CortanaApp\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.CortanaApp\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.BingWeather"; File = "Microsoft.BingWeather\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.BingWeather\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.GetHelp"; File = "Microsoft.GetHelp\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.GetHelp\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.MicrosoftOfficeHub"; File = "Microsoft.MicrosoftOfficeHub\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.MicrosoftOfficeHub\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.People"; File = "Microsoft.People\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.People\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.StorePurchaseApp"; File = "Microsoft.StorePurchaseApp\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.StorePurchaseApp\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.Todos"; File = "Microsoft.Todos\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.Todos\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.WebMediaExtensions"; File = "Microsoft.WebMediaExtensions\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.WebMediaExtensions\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.Windows.Photos"; File = "Microsoft.Windows.Photos\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.Windows.Photos\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.WindowsCalculator"; File = "Microsoft.WindowsCalculator\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.WindowsCalculator\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.Windows.CommunicationsApps"; File = "Microsoft.WindowsCommunicationsApps\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.WindowsCommunicationsApps\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.WindowsSoundRecorder"; File = "Microsoft.WindowsSoundRecorder\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.WindowsSoundRecorder\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.Xbox.TCUI"; File = "Microsoft.Xbox.TCUI\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.Xbox.TCUI\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.XboxGameOverlay"; File = "Microsoft.XboxGameOverlay\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.XboxGameOverlay\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.XboxGamingOverlay"; File = "Microsoft.XboxGamingOverlay\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.XboxGamingOverlay\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.XboxIdentityProvider"; File = "Microsoft.XboxIdentityProvider\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.XboxIdentityProvider\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.XboxSpeechToTextOverlay"; File = "Microsoft.XboxSpeechToTextOverlay\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.XboxSpeechToTextOverlay\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.YourPhone"; File = "Microsoft.YourPhone\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.YourPhone\_8wekyb3d8bbwe.xml"; }

@{ Name = "Microsoft.ZuneVideo"; File = "Microsoft.ZuneVideo\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.ZuneVideo\_8wekyb3d8bbwe.xml"; }

@{ Name = "MicrosoftCorporationII.QuickAssist"; File = "MicrosoftCorporationII.QuickAssist\_8wekyb3d8bbwe.appxbundle"; License = "MicrosoftCorporationII.QuickAssist\_8wekyb3d8bbwe.xml"; }

@{ Name = "MicrosoftWindows.Client.WebExperience"; File = "MicrosoftWindows.Client.WebExperience\_cw5n1h2txyewy.appxbundle"; License = "MicrosoftWindows.Client.WebExperience\_cw5n1h2txyewy.xml"; }

@{ Name = "Microsoft.RawImageExtension"; File = "Microsoft.RawImageExtension\_8wekyb3d8bbwe.appxbundle"; License = "Microsoft.RawImageExtension\_8wekyb3d8bbwe.xml"; }

)

Function Install\_Appx

{

param ( $File, $License )

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

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

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

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

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

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

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

try {

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

Write-Host "Done" -ForegroundColor Green

} catch {

Write-Host "Failed" -ForegroundColor Red

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

}

} else {

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

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

try {

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

Write-Host "Done" -ForegroundColor Green

} catch {

Write-Host "Failed" -ForegroundColor Red

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

}

}

} else {

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

}

}

ForEach ($Rule in $InBoxApps) {

Write-host "`n Name: " -NoNewline

Write-host $Rule.Name -ForegroundColor Yellow

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

if($Allow\_Install\_App -contains $Rule.Name) {

Write-host " Search for apps: " -NoNewline

Write-host $Rule.File -ForegroundColor Yellow

Write-host " Search for License: " -NoNewline

Write-host $Rule.File -ForegroundColor Yellow

if ($ISO -eq "Auto") {

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

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

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

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

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

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

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

Write-host " Discover License: " -NoNewLine; Write-host $LicensePath -ForegroundColor Green

} else {

Write-host " License: " -NoNewLine; Write-host "Not found" -ForegroundColor Red

}

Install\_Appx -File $AppPath -License $LicensePath

return

}

}

} else {

Install\_Appx -File "$($ISO)\$($Rule.File)" -License "$($ISO)\$($Rule.License)"

}

} else {

Write-host " Skip the installation" -ForegroundColor Red

}

}

* 1. [**Region tag: delete**](#_InBox_Apps)
* 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](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\Install.Clear.Flag.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Expand/Install/Install.Clear.Flag.ps1>
* Copy the code

$Lang = "zh-CN"

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

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

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

try {

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

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

}

}

}

###### [**InBox Apps: optimization**](#_InBox_Apps)

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

##### [**Cumulative updates**](#_3.1__)

It is recommended that you download the latest version, which provides cumulative updates of the initial version to solve the problem of refreshing the "Component: All packages installed in the image" status after reinstalling the same version number.

###### **[Initial version](#_更新)**

* + 1. [**How to download**](#_Initial_version)

Cumulative update KB5016632 can no longer be searched from <https://www.catalog.update.microsoft.com/Search.aspx?q=KB5016632>. For other download methods, select any website and open it:

* <https://uupdump.net>
* <https://uup.ee>
  + - 1. After opening, search for keywords: 22621.382, select from the search results: Windows 11, version 22H2 (22621.382) amd64
      2. After opening, select "All files";
      3. Search for: KB5016632 in the "All Files" page, download:
* Windows11.0-KB5016632-x64.cab
* Windows11.0-KB5016632-x64.psf
  + - 1. **Merging cumulative updates**
         1. Go to <https://github.com/abbodi1406/WHD/tree/master/scripts> After downloading [PSFX\_Repack\_6.zip](https://github.com/abbodi1406/WHD/raw/master/scripts/PSFX_Repack_6.zip)
         2. After downloading, unzip it to: D:\PSFX\_Repack\_6
         3. Copy Windows11.0-KB5016632-x64.cab, Windows11.0-KB5016632-x64.psf to: D:\PSFX\_Repack\_6
         4. After running: psfx2cab\_GUI.cmd, you will get a new file:

Windows11.0-KB5016632-x64-full\_psfx.cab

* + - * 1. Save Windows11.0-KB5016632-x64-full\_psfx.cab to:

D:\OS\_11\_Custom\Install\Install\Update\Windows11.0-KB5016632-x64-full\_psfx.cab

* + 1. [**Add**](#_Initial_version)

$KBPath = "D:\OS\_11\_Custom\Install\Install\Update\Windows11.0-KB5016632-x64-full\_psfx.cab"

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

###### **[Other Version](#_更新)**

* + 1. [**How to download**](#_Other_Version)

Check out "[Windows 11 version information](https://learn.microsoft.com/en-us/windows/release-health/windows11-release-information)”, for example, to download the cumulative update: KB5030219, go to the download page: <https://www.catalog.update.microsoft.com/Search.aspx?q=KB5030219> or "[Direct download](https://catalog.sf.dl.delivery.mp.microsoft.com/filestreamingservice/files/b82b3f6d-17e2-46d5-998a-893deb622e0a/public/windows11.0-kb5030219-x64_659d407821da654d564a3108bf493594bd143ab1.msu)", Save to:

D:\OS\_11\_Custom\Install\Install\Update\windoOS\_11.0-kb5030219-x64\_659d407821da654d564a3108bf493594bd143ab1.msu

* + 1. [**Add**](#_Other_Version)

$KBPath = "D:\OS\_11\_Custom\Install\Install\Update\windoOS\_11.0-kb5030219-x64\_659d407821da654d564a3108bf493594bd143ab1.msu"

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

###### **[Solidify Updated, optiona](#_更新)**[l](#_更新)

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

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

* + 1. [**Clean components after curing and updating, optional**](#_固化更新，可选项)

$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

}

}

##### [**Driver, optional**](#_3.1__)

##### **Add deployment engine, optional**

* Learn "Deployment Engine", if added to ISO installation media, can skip adding to mounted.
* After adding the deployment engine, continue at the current location.

##### [**Health**](#_3.1__)

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

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

##### **[Replace WinRE.wim](#_3.1__)**

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

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

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

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

##### [**Save image**](#_3.1__)

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

##### [**Unmount image**](#_3.1__)

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

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

**LOOP OPERATING AREA, END.**

##### **[How to batch replace WinRE.wim in all index numbers in Install.wim](#_3.1__)**

###### [**Obtain WimLib**](#_How_to_batch)

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

###### **[How to extract and update WinRE.wim in Install.wim](#_如何批量替换_Install.wim_里的所有索引号里的)**

* + 1. [**Extract the WinRE.wim file Install.wim from Install.wim**](#_How_to_extract)
* Install.WinRE.Extract.ps1
  + [\Expand\Install\Install.WinRE.Extract.ps1](Expand/Install/Install.WinRE.Extract.ps1)
  + [https://github.com/ilikeyi/solutions/blob/main/\_Documents/Attachment/OS.11/22H2/Expand/Install/Install.WinRE.Extract.ps1](https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Code/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 -ea SilentlyContinue

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

* + 1. **[Get all index numbers of Install.wim and replace the old WinRE.wim](#_How_to_extract)**
* Install.WinRE.Replace.wim.ps1
  + [\Expand\Install\Install.WinRE.Replace.wim.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\Install.WinRE.Replace.wim.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Expand/Install/Install.WinRE.Replace.wim.ps1>
* Copy the code

Get-WindowsImage -ImagePath "D:\OS\_11\sources\install.wim" -ErrorAction SilentlyContinue | ForEach-Object {

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

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

Write-Host "`n Replacement "

$Arguments = @(

"update",

"D:\OS\_11\sources\install.wim", $\_.ImageIndex,

"--command=""add 'D:\OS\_11\_Custom\Install\Install\Update\Winlib\WinRE.wim' '\Windows\System32\Recovery\WinRe.wim'"""

)

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

Write-Host " Finish`n" -ForegroundColor Green

}

##### **[Rebuilding Install.wim reduces file size](#_3.1__)**

* Install.Rebuild.wim.ps1
  + [\Expand\Install\Install.Rebuild.wim.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Install\Install.Rebuild.wim.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/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

}

#### **[Custom deployment image: boot.wim](#_INSTALL_LANGUAGE_PACK)**

##### [**View Boot.wim details**](#_Install_the_language)

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

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

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

##### [**Specify the path to mount Boot.wim**](#_Install_the_language)

New-Item -Path "D:\OS\_11\_Custom\Boot\Boot\Mount" -ItemType directory -ea SilentlyContinue

1. [**Start mounting Boot.wim**](#_Install_the_language)

Default index number: 2

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

##### **[Language pack](#_Install_the_language)**

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.

###### [**Language pack: add**](#_After_executing_the)

* Boot.Instl.lang.ps1
  + [\Expand\Boot\Boot.Instl.lang.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Boot\Boot.Instl.lang.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/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

}

}

}

###### **Offline image language: change**

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

Region: zh-CN

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

* + 1. **View available language settings**

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

###### **[Language packs: Removed, optional](#_执行安装命令后安装语言包到：WinRE)**

* After you add languages, if you want to deploy to a non-English locale, you can save space by removing the English language component. When you remove a language, uninstall the language components in the reverse order in which you added them.
* After adding the Chinese, delete "English - United States" in reverse, the Region tag: en-US, you need to extract the language pack in advance
* Boot.Del.Specified.lang.Tag.ps1
  + [\Expand\Boot\Boot.Del.Specified.lang.Tag.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Boot\Boot.Del.Specified.lang.Tag.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Expand/Boot/Boot/Boot.Del.Specified.lang.Tag.ps1>
* Copy the code

$Lang = "en-US"

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

$Sources = "D:\OS\_11\_Custom\Boot\Boot\Language\Del\en-US"

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

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

$Language = @(

@{ Match = "\*WordPad\*wow64\*"; File = "Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

@{ Match = "\*WordPad\*amd64\*"; File = "Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~$($Lang)~.cab"; }

@{ Match = "\*WMIC\*FoD\*Package\*wow64\*"; File = "Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

@{ Match = "\*WMIC\*FoD\*Package\*amd64\*"; File = "Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

@{ Match = "\*StepsRecorder\*wow64\*"; File = "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

@{ Match = "\*StepsRecorder\*amd64\*"; File = "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~$($Lang)~.cab"; }

@{ Match = "\*Printing\*PMCPPC\*amd64\*"; File = "Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

@{ Match = "\*PowerShell\*wow64\*"; File = "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

@{ Match = "\*PowerShell\*amd64\*"; File = "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~$($Lang)~.cab"; }

@{ Match = "\*MediaPlayer\*wow64\*"; File = "Microsoft-Windows-MediaPlayer-Package-wow64-$($Lang).cab"; }

@{ Match = "\*MediaPlayer\*amd64\*"; File = "Microsoft-Windows-MediaPlayer-Package-AMD64-$($Lang).cab"; }

@{ Match = "\*Notepad\*wow64\*"; File = "Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~wow64~$($Lang)~.cab"; }

@{ Match = "\*Notepad\*amd64\*"; File = "Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

@{ Match = "\*InternetExplorer\*"; File = "Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~AMD64~$($Lang)~.cab"; }

@{ Match = "\*TextToSpeech\*"; File = "Microsoft-Windows-LanguageFeatures-TextToSpeech-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

@{ Match = "\*LanguageFeatures-Speech\*"; File = "Microsoft-Windows-LanguageFeatures-Speech-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

@{ Match = "\*OCR\*"; File = "Microsoft-Windows-LanguageFeatures-OCR-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

@{ Match = "\*Handwriting\*"; File = "Microsoft-Windows-LanguageFeatures-Handwriting-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

@{ Match = "\*LanguageFeatures-Basic\*"; File = "Microsoft-Windows-LanguageFeatures-Basic-$($Lang)-Package~31bf3856ad364e35~amd64~~.cab"; }

@{ Match = "\*Client-LanguagePack-Package\*"; File = "Microsoft-Windows-Client-Language-Pack\_x64\_$($Lang).cab"; }

)

ForEach ($Rule in $Language) {

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

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

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

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 " Deleting ".PadRight(22) -NoNewline

try {

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

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

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

}

break

}

}

}

$InitlClearLanguagePackage = @()

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

if ($\_.PackageName -like "\*$($Lang)\*") {

$InitlClearLanguagePackage += $\_.PackageName

}

}

if ($InitlClearLanguagePackage.count -gt 0) {

ForEach ($item in $InitlClearLanguagePackage) {

Write-Host "`n $($item)" -ForegroundColor Green

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

try {

Remove-AppxProvisionedPackage -Path $Mount -PackageName $item -ErrorAction SilentlyContinue | Out-Null

Write-host "Finish" -ForegroundColor Green

} catch {

Write-host "Failed" -ForegroundColor Red

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

}

}

}

###### **Components: [All packages installed in the image](#_After_executing_the)**

* + 1. [**View**](#_Components)

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

* + 1. [**Export to csv**](#_Components)

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

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

Write-host $SaveTo -ForegroundColor Green

###### **[Language: Repair](#_After_executing_the)**

* + 1. [**Extract**](#_Language:_Repair)

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:

* + - 1. arunres.dll.mui
      2. spwizres.dll.mui
      3. w32uires.dll.mui
    1. [**Copy**](#_Language:_Repair)

Copy the extracted files to: D:\OS\_11\_Custom\Boot\Boot\Mount\sources\zh-CN

###### **[Language packs: sync to ISO installer](#_After_executing_the)**

Copy-Item -Path "D:\OS\_11\_Custom\Boot\Boot\Mount\sources\zh-CN" -Destination "D:\OS\_11\sources\zh-CN" -Recurse -Force

###### **[Regenerate Lang.ini](#_After_executing_the)**

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

* + 1. [**Regenerate the mounted directory lang.ini**](#_Regenerate_Lang.ini)

Regenerated Lang.ini file location: D:\en-us\_windows\_server\_2022\_x64\_dvd\_620d7eac\_Custom\Boot\Boot\Mount\Sources\lang.ini

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

* + 1. [**After regenerating lang.ini, sync to the installer**](#_Regenerate_Lang.ini)

Regenerated Lang.ini file location: D:\en-us\_windows\_server\_2022\_x64\_dvd\_620d7eac\Sources\lang.ini

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

##### [**Cumulative updates, optional**](#_安装语言包到：boot.wim)

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

* 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

* 1. **Delete**

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

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

* 1. **Solid update, optional**

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

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

* + 1. [**Clean components after curing and updating, optional**](#_固化更新，可选项)

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

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

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

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

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

}

}

##### [**Drive, optional**](#_Install_the_language)

##### **[Other](#_Install_the_language)**

###### [**Bypass TPM**](#_Other)

* Boot.Bypass.TPM.ps1
  + [\Expand\Boot\Boot.Bypass.TPM.ps1](file:///D:\Yi.Solutions\_Documents\Attachment\OS.11\22H2\Expand\Boot\Boot.Bypass.TPM.ps1)
  + <https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Expand/Boot/Boot.Bypass.TPM.ps1>
* Copy the code

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

$RandomGuid = [guid]::NewGuid()

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

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

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

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

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

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

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

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

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

[gc]::collect()

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

Remove-PSDrive -Name OtherTasksTPM

##### [**Save image**](#_Install_the_language)

Save-WindowsImage -Path "D:\OS\_11\_Custom\Boot\Boot\Mount"

##### [**Unmount image**](#_Install_the_language)

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

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

#### **Deployment engine**

* + - * Learn about "Automatically Adding Languages Installed in Windows Systems", learn: <https://github.com/ilikeyi/Multilingual>, how to download:
      * After entering the website, click "Code", "Download Compressed Package", and after the download is completed, you will get the main.zip compressed package file.
      * Go to the <https://github.com/ilikeyi/Multilingual/releases> download page, select the available version: 1.1.0.4, select the download source code format: zip, and get the Multilingual-1.1.0.4.zip compressed package file after the download is completed;
      * Unzip the downloaded main.zip or Multilingual-1.1.0.4.zip to: D:\Multilingual-1.1.0.4, and rename: D:\Multilingual
      * Learn "[Unattended Windows Setup Reference](https://learn.microsoft.com/en-us/windows-hardware/customize/desktop/unattend/)", Intervene in the installation process by leaving it unattended.

##### **Add method**

###### **Add to ISO installation media**

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

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

Copy D:\Multilingual\Unattend\Mul.Unattend.xml to D:\OS\_11\Autounattend.xml

Copy-Item "D:\Multilingual\Unattend\Mul.Unattend.xml" -Destination "D:\OS\_11\Autounattend.xml" -Force

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

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

Copy D:\Multilingual\Unattend\Mul.Unattend.xml to D:\OS\_11\Sources\Unattend.xml

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

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

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

* + - * 1. **Create $OEM$ path**

New-Item -Path "D:\OS\_11\sources\`$OEM$\`$$\Panther" -ItemType Directory

* + - * 1. **Copy**

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

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

* + 1. **Deployment engine: add**

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

* + - 1. **Deployment Engine: Copy**

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

Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS\_11\sources\`$OEM$\`$1\Yi\Engine" -Recurse -Force

* + - 1. **Deployment engine: custom deployment tags**

$Flag = @(

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

# Prerequisite deployment

# "Auto\_Update" # Allow automatic updates

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

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

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

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

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

"Prerequisites\_Reboot" # Restart your computer

# Complete first deployment

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

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

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

"Clear\_Solutions" # Delete the entire solution

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

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

)

ForEach ($item in $Flag) {

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

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

Out-File -FilePath "D:\OS\_11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow\$($item)" -Encoding utf8 -ErrorAction SilentlyContinue

}

###### **Add to mounted**

Through "[Customized deployment image: Install.wim](#_INSTALL_LANGUAGE_PACK)", execute "[Start mounting Install.wim](#_Start_mounting_Install.wim)" and mount to: D:\OS\_11\_Custom\Install\Install\Mount

* + 1. **Unattended**

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

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

* + 1. **Deployment engine: add**

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

* + - 1. **Deployment Engine: Copy**

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

Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS\_11\_Custom\Install\Install\Mount\Yi\Engine" -Recurse -Force

* + - 1. **Deployment engine: custom deployment tags**

$Flag = @(

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

# Prerequisite deployment

# "Auto\_Update" # Allow automatic updates

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

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

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

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

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

"Prerequisites\_Reboot" # Restart your computer

# Complete first deployment

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

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

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

"Clear\_Solutions" # Delete the entire solution

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

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

)

ForEach ($item in $Flag) {

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

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

Out-File -FilePath "D:\OS\_11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow\$($item)" -Encoding utf8 -ErrorAction SilentlyContinue

}

##### **Deployment Engine: Advanced**

###### **Deployment engine: adding process**

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

###### **Unattended solution**

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

* D:\OS\_11\Autounattend.xml
* D:\OS\_11\Sources\Unattend.xml
* D:\OS\_11\sources\$OEM$\$$\Panther\unattend.xml
* D:\OS\_11\_Custom\Install\Install\Mount\Panther\Unattend.xml
  + 1. **Multilingual or monolingual**

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

* + - 1. **Multi-language**

<UILanguage>%OSDUILanguage%</UILanguage>

<InputLocale>%OSDInputLocale%</InputLocale>

<SystemLocale>%OSDSystemLocale%</SystemLocale>

<UILanguage>%OSDUILanguage%</UILanguage>

<UILanguageFallback>%OSDUILanguageFallback%</UILanguageFallback>

<UserLocale>%OSDUserLocale%</UserLocale>

* + - 1. **Monolingual**

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

<UILanguage>zh-CN</UILanguage>

<InputLocale>zh-CN</InputLocale>

<SystemLocale>zh-CN</SystemLocale>

<UILanguage>zh-CN</UILanguage>

<UILanguageFallback>zh-CN</UILanguageFallback>

<UserLocale>zh-CN</UserLocale>

* + 1. **User plan**

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

* + - 1. **Self-created user Administrator**

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

<UserAccounts>

<LocalAccounts>

<LocalAccount wcm:action="add">

<Password>

<Value></Value>

<PlainText>true</PlainText>

</Password>

<Description>Administrator</Description>

<DisplayName>Administrator</DisplayName>

<Group>Administrators</Group>

<Name>Administrator</Name>

</LocalAccount>

</LocalAccounts>

</UserAccounts>

<AutoLogon>

<Password>

<Value></Value>

<PlainText>true</PlainText>

</Password>

<Enabled>true</Enabled>

<Username>Administrator</Username>

</AutoLogon>

* + - 1. **Custom user**

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

* + - * 1. **Delete**

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

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

* + - * 1. **Replace**

From the beginning <OOBE> to </OOBE>

<OOBE>

<ProtectYourPC>3</ProtectYourPC>

<HideEULAPage>true</HideEULAPage>

<HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>

</OOBE>

### **[ISO](#_Custom_deployment_image)**

#### [**Generate ISO**](#_4、生成_ISO)

1. After downloading oscdimg, save it to: C:\Windows\System32 directory, save it in another path, or please enter the absolute location of OScdimg.exe;
2. Use the oscdimg command line to generate an ISO file and save it to: D:\OS\_11.iso

Copy the code or view the source file: ISO.ps1 ( [Local](Expand/ISO.ps1), [Github](https://github.com/ilikeyi/solutions/blob/main/_Documents/Attachment/OS.11/22H2/Expand/ISO.ps1) )

$ISO = "D:\OS\_11"; $Volume = "OS\_11"; $SaveTo = "D:\OS\_11.iso"

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

Start-Process -FilePath "OSCdimg.exe" -ArgumentList $Arguments -wait -nonewwindow

#### [**Bypass TPM installation check**](#_4、生成_ISO)

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

## **COMMON PROBLEM**

1. **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

1. **Fix the problem of abnormal mounting**
2. **View mounted**

Get-WindowsImage -Mounted

1. **Delete the DISM mount record saved in the registry**

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

1. **Delete all resources associated with the corrupted mounted image**

Clear-WindowsCorruptMountPoint

Dism /cleanup-wim

1. **Clean directories**

Remove-Item "D:\OS\_11\_Custom" -Force -Recurse

Remove-Item "D:\OS\_11\Sources\`$OEM$" -Force -Recurse

Remove-Item -Path "D:\OS\_11\Autounattend.xml" -Force

Remove-Item -Path "D:\OS\_11\Sources\Unattend.xml" -Force

## **[Known issues](#_top)**

None yet

## **[Components: All packages installed in the image, Comparison](#_top)**

Changes after installing new language pack, component list:

| SN | Type | Component: Original | After you install the Chinese Simplified - China language pack |
| --- | --- | --- | --- |
|  | OnDemandPack | Microsoft-OneCore-ApplicationModel-Sync-Desktop-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.317 | Microsoft-OneCore-ApplicationModel-Sync-Desktop-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.317 |
|  | OnDemandPack | Microsoft-OneCore-DirectX-Database-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-OneCore-DirectX-Database-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | LanguagePack | Microsoft-Windows-Client-LanguagePack-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.382 | Microsoft-Windows-Client-LanguagePack-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.382 |
|  | LanguagePack |  | Microsoft-Windows-Client-LanguagePack-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.382 |
|  | OnDemandPack | Microsoft-Windows-Ethernet-Client-Intel-E1i68x64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Ethernet-Client-Intel-E1i68x64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Ethernet-Client-Intel-E2f68-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Ethernet-Client-Intel-E2f68-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Ethernet-Client-Realtek-Rtcx21x64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Ethernet-Client-Realtek-Rtcx21x64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Ethernet-Client-Vmware-Vmxnet3-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Ethernet-Client-Vmware-Vmxnet3-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | FeaturePack | Microsoft-Windows-FodMetadata-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-FodMetadata-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | Foundation | Microsoft-Windows-Foundation-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Foundation-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Hello-Face-Package~31bf3856ad364e35~amd64~~10.0.22621.317 | Microsoft-Windows-Hello-Face-Package~31bf3856ad364e35~amd64~~10.0.22621.317 |
|  | OnDemandPack | Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~amd64~en-US~11.0.22621.1 | Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~amd64~en-US~11.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~amd64~zh-CN~11.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~amd64~~11.0.22621.1 | Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~amd64~~11.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Kernel-LA57-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.382 | Microsoft-Windows-Kernel-LA57-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.382 |
|  | OnDemandPack | Microsoft-Windows-LanguageFeatures-Basic-en-us-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-LanguageFeatures-Basic-en-us-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-LanguageFeatures-Basic-zh-cn-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-LanguageFeatures-Fonts-Hans-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-LanguageFeatures-Handwriting-en-us-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-LanguageFeatures-Handwriting-en-us-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-LanguageFeatures-Handwriting-zh-cn-Package~31bf3856ad364e35~amd64~~10.0.22621.317 |
|  | OnDemandPack | Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-LanguageFeatures-OCR-en-us-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-LanguageFeatures-OCR-zh-cn-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-LanguageFeatures-Speech-en-us-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-LanguageFeatures-Speech-en-us-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-LanguageFeatures-Speech-zh-cn-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-LanguageFeatures-TextToSpeech-en-us-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-LanguageFeatures-TextToSpeech-en-us-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-LanguageFeatures-TextToSpeech-zh-cn-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.317 | Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.317 |
|  | OnDemandPack |  | Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.317 |
|  | OnDemandPack | Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~amd64~~10.0.22621.382 | Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~amd64~~10.0.22621.382 |
|  | OnDemandPack | Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 | Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~wow64~~10.0.22621.1 | Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~wow64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 | Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 | Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~wow64~~10.0.22621.1 | Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~wow64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 | Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 | Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~~10.0.22621.1 | Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 | Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 | Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 | Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~~10.0.22621.1 | Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-TabletPCMath-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-TabletPCMath-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wallpaper-Content-Extended-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wallpaper-Content-Extended-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Broadcom-Bcmpciedhd63-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Broadcom-Bcmpciedhd63-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Broadcom-Bcmwl63a-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Broadcom-Bcmwl63a-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Broadcom-Bcmwl63al-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Broadcom-Bcmwl63al-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwbw02-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwbw02-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwew00-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwew00-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwew01-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwew01-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwlv64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwlv64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwns64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwns64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwsw00-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwsw00-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwtw02-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwtw02-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwtw04-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwtw04-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwtw06-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwtw06-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwtw08-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwtw08-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Intel-Netwtw10-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Intel-Netwtw10-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Marvel-Mrvlpcie8897-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Marvel-Mrvlpcie8897-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Qualcomm-Athw8x-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Qualcomm-Athw8x-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Qualcomm-Athwnx-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Qualcomm-Athwnx-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Qualcomm-Qcamain10x64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Qualcomm-Qcamain10x64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Ralink-Netr28x-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Ralink-Netr28x-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Realtek-Rtl8187se-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Realtek-Rtl8187se-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Realtek-Rtl8192se-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Realtek-Rtl8192se-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Realtek-Rtl819xp-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Realtek-Rtl819xp-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Realtek-Rtl85n64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Realtek-Rtl85n64-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Realtek-Rtwlane-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Realtek-Rtwlane-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Realtek-Rtwlane01-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Realtek-Rtwlane01-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-Wifi-Client-Realtek-Rtwlane13-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | Microsoft-Windows-Wifi-Client-Realtek-Rtwlane13-FOD-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 | Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.382 | Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.382 |
|  | OnDemandPack | Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 | Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~~10.0.22621.1 | Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 | Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.317 | Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.317 |
|  | OnDemandPack | Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 | Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~en-US~10.0.22621.1 |
|  | OnDemandPack |  | Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1 |
|  | OnDemandPack | Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~~10.0.22621.1 | Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~~10.0.22621.1 |
|  | OnDemandPack | OpenSSH-Client-Package~31bf3856ad364e35~amd64~~10.0.22621.1 | OpenSSH-Client-Package~31bf3856ad364e35~amd64~~10.0.22621.1 |
|  | Update | Package\_for\_DotNetRollup\_481~31bf3856ad364e35~amd64~~10.0.9065.6 | Package\_for\_DotNetRollup\_481~31bf3856ad364e35~amd64~~10.0.9065.6 |
|  | SecurityUpdate | Package\_for\_RollupFix~31bf3856ad364e35~amd64~~22621.382.1.13 | Package\_for\_RollupFix~31bf3856ad364e35~amd64~~22621.382.1.13 |
|  | SecurityUpdate | Package\_for\_ServicingStack\_378~31bf3856ad364e35~amd64~~22621.378.1.0 | Package\_for\_ServicingStack\_378~31bf3856ad364e35~amd64~~22621.378.1.0 |