Yi' Solutions

Microsoft Windows 11 22H2

PACKAGING TUTORIAL: Different system versions have different packaging methods. The packaging process includes: "Language pack: add, associate, delete", "Drive: add, delete", "Cumulative update: add, delete", "InBox Apps: add, Update, mark" etc.

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

Summary

Table of contents

Chapter 1 Encapsulation

A. Prerequisites

II ISO tools

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

III Requirements

1. System installation package

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

1.1. illustrate

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

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

1.1.3. Please use "Initial Version" and "Developer Edition" for production. There was a brief appearance in the official Microsoft documentation that the initial version must be used in production, but this sentence was later deleted in the official documentation;

1.1.4. After installing the language pack, you must re-add the cumulative update (the same version number), and if you do not add the cumulative update, problems such as "garbled characters" and "interface flashback" will occur.

1.1.5. Evolutionary process: Windows 11 22H2, Build 22621.382 + KB5027303 = OS Build 22621.1928

1.2. Prepare to download the initial or developer version

1.2.1. x64

1.2.1.1. en-us_windows_11_business_editions_version_22h2_x64_dvd_17a08ce3.iso

1.2.1.2. en-us_windows_11_consumer_editions_version_22h2_x64_dvd_e630fafd.iso

1.3. After the sample download en-us_windows_11_business_editions_version_22h2_x64_dvd_17a08ce3.iso,Unzip to: D:\en-us_windows_11_business_editions_version_22h2_x64_dvd_17a08c

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

1.4. After decompression is complete, change the directory en-us_windows_11_business_editions_version_22h2_x64_dvd_17a08c change to D:\OS_11

1.5. All scripts and all paths are set to D:\OS_11 as the image source.

1.6. Installation configuration

1.6.1. Learn: Windows Setup Edition Configuration and Product ID Files (EI.cfg and PID.txt)

1.6.2. Known issues

1.6.2.1. When there is no Ei.cfg, ISO boot installation will report an error when selecting certain versions, prompting: Windows cannot find the Microsoft Software License terms. Make sure the insttallation sources are valid and restart the installation.

1.6.2.2. How to solve it? Add ei.cfg to D:\OS_11\Sources and create ei.cfg:

@"

[Channel]

volume

[VL]

1

"@ | Out-File -FilePath "D:\OS_11\sources\EI.CFG" -Encoding Ascii

2. Language pack

2.1. Learn

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

2.1.1. Languages overview

2.1.2. Add languages to a Windows 11 image

2.1.3. Language and region Features on Demand (FOD)

2.2. Language pack: Download

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

2.3. Language pack: Fixed

2.3.1. Select any website and open:

2.3.1.1. https://uupdump.net

2.3.1.2. https://uup.ee

2.3.1.3. https://osdump.com

2.3.2. After opening, search for keywords: 22621.382, select from the search results: Windows 11, version 22H2 (22621.382) amd64

2.3.3. After opening, select "All files";

2.3.4. Search the green part in the "All Files" page and download

2.3.4.1. Applies to: Install.wim:

2.3.4.1.1. MediaPlayer

2.3.4.2. Applies to: WinRE.wim, none yet

2.3.4.3. Applies to: Boot.wim, none yet

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

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

3. InBox Apps

3.1. Download: https://software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/22621.1778.230511-2102.ni_release_svc_prod3_amd64fre_InboxApps.iso

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

3.2.1. Microsoft.HEVCVideoExtension

3.2.1.1. Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.x64.appx

3.2.1.2. Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.x64.xml

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

IV Windows Security

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

How to close:

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

1. Open Windows Security or run: windowsdefender:
2. Select "Virus & Threat Protection" or Run: windowsdefender://threat
3. Find "Virus & Threat Protection Settings", click "Manage Settings" or Run: windowsdefender://threatsettings, we recommend that you turn off some features:

3.1. Real-time protection

3.2. Cloud=delivered protection

3.3. Automatic sample submission

3.4. Tamper Protection

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

V Command line

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

2. Open "Terminal" or "PowerShell ISE" as administrator, it is recommended to set the PowerShell execution policy: bypass, PS command line:

   Set-ExecutionPolicy -ExecutionPolicy Bypass -Force

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

4. When there is .ps1, right-click the file and select Run with PowerShell, or copy the path and paste it into Terminal to run, the path with a colon, add the & character in the command line, example: & "D:\YiSolutions_Encapsulation_SIP.ps1"

B. Language Pack: Extract

II Language pack: Ready

Mount 22621.1.220506-1250.ni_release_amd64fre_CLIENT_LOF_PACKAGES_OEM.iso or unzipped to any location;

III Language pack: Scheme

1. Add

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

2. Delete

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

IV Execute the extract command

- Auto = automatically search all local disks, default;
- Customize the path, for example, specify the E drive: $ISO = "E:\"
- Extract.ps1
  - \Expand\Extract.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/22H2/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 }
```

C. Custom encapsulation

II Custom encapsulation Install.wim

1. View Install.wim details

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

```
$ViewFile = "D:\OS_11\Sources\Install.wim"

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

LOOP OPERATING AREA, START,

2. Specify the path to mount Install.wim

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

3. Start mounting Install.wim

Default index number: 1

```
Mount-WindowsImage -ImagePath "D:\OS_11\sources\install.wim" -Index "1" -Path "D:\OS_11_Custom\Install\Install\Mount"
```

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

3.1. Custom encapsulation WinRE.wim

WARNING:

- WinRE.wim is a file within the Install.wim image;
- When Install.wim has multiple index numbers, only process any WinRE.wim;
- Synchronize to all index numbers to reduce the size of Install.wim, learn "How to batch replace WinRE.wim in all index numbers in Install.wim".

3.1.1. View WinRE.wim details

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

```
$ViewFile = "D:\OS_11_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

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

3.1.2. Specify the path to mount WinRE.wim

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

3.1.3. Start mounting WinRE.wim

Default index number: 1

```
Mount-WindowsImage -ImagePath  "D:\OS_11_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim" -Index "1" -Path "D:\OS_11_Custom\Install\WinRE\Mount"
```

3.1.4. Language pack: WinRE

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

3.1.4.1. Language pack: add

- WinRE.Instl.lang.ps1
  - \Expand\Install\WinRE\WinRE.Instl.lang.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/22H2/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
        }
```

```
    }
}
```

3.1.4.2. Components: All packages installed in the image

3.1.4.2.1. View

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

3.1.4.2.2. Export to csv

```
$SaveTo = "D:\OS_11_Custom\Install\WinRE\Report.$(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
```

3.1.5. Save image: WinRE.wim

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

3.1.6. Unmount image: WinRE.wim

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

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

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

- WinRE.Rebuild.ps1
    - \Install\WinRE\WinRE.Rebuild.ps1
    - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/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
}
```

3.1.8. Backup WinRE.wim

- WinRE.Backup.ps1
  - \Expand\Install\WinRE\WinRE.Backup.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/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
```

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

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

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

4. Language pack

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

4.1. Language pack: add

- Install.Instl.lang.ps1
  - \Expand\Install\Install.Instl.lang.ps1

    - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/22H2/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"
```

4.2. Components: All packages installed in the image

4.2.1. View

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

4.2.2. Export to csv

```
$SaveTo = "D:\OS_11_Custom\Install\Install\Report.$(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
```

5. InBox Apps

5.1. InBox Apps: Installed

5.1.1. View

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

5.1.2. 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
```

5.2. Remove all installed pre-applications

- Install.InBox.Appx.Clear.all.ps1
  - \Expand\Install\Install.InBox.Appx.Clear.all.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.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
  }
}
```

5.3. Region tag: adding method

5.3.1. Execute "Language Pack: Add"

5.3.2. 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. Want to get:

5.3.2.1. 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

5.3.2.2. Manual download

5.3.2.2.1. Region

Download Region: zh-CN, application ID: 9NRMNT6GMZ70, Store link:

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

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

5.3.2.2.2.1. Search keywords:

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

5.3.2.2.2.2. Search 22621 content in the web page, search results: Microsoft.LanguageExperiencePackzh-CN_22621.*.neutral__8wekyb3d8bbwe.appx

5.3.2.2.2.3. After downloading, save it to the

D:\OS_11_Custom\Install\Install\InBox.Appx directory and rename it: LanguageExperiencePack.zh-cn.Neutral.Appx

5.3.2.3. Execute the installation command to install the local language experience package (LXPs)

After understanding how to add zone tags, obtain LanguageExperiencePack.zh-cn.Neutral, 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
```

5.3.2.4. InBox Apps: An installed application package

5.3.2.4.1. View

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

5.3.2.4.2. 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
```

5.4. InBox Apps: Install

5.4.1. Mount or decompress the InBox Apps installation file

Mount 22621.1.220506-1250.ni_release_amd64fre_InboxApps.iso or extract to any location;

5.4.2. After executing the installation command, install InBox Apps to: Install.wim

- Auto = Automatically search all local disks, default;
- Custom path, e.g. specify F drive: $ISO = "F:\packages"
- Architecture: x64
- Install.Inst.InBox.Appx.ps1

- \Expand\Install\Install.Inst.InBox.Appx.ps1
- https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/22H2/Expand/Install/Install.Inst.InBox.Appx.ps1

- Copy the code

```
$ISO = "Auto"

$Mount = "D:\OS_11_Custom\Install\Install\Mount"

$Arch = "x64"

try {

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

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

  Write-Host $Current_Edition_Version -ForegroundColor Green

} catch {

  Write-Host "Error" -ForegroundColor Red

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

  return

}

$Pre_Config_Rules = @{

  Edition = @(

    @{

      Name = @( "CloudEdition"; )

      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.Services.Store.Engagement"; "Microsoft.VP9VideoExtensions";
"Clipchamp.Clipchamp"; "Microsoft.BingNews"; "Microsoft.BingWeather"; "Microsoft.DesktopAppInstaller";
"Microsoft.GetHelp"; "Microsoft.Getstarted"; "Microsoft.HEIFImageExtension"; "Microsoft.HEVCVideoExtension";
"Microsoft.MicrosoftOfficeHub"; "Microsoft.MicrosoftStickyNotes"; "Microsoft.MinecraftEducationEdition";
"Microsoft.Paint"; "Microsoft.RawImageExtension"; "Microsoft.ScreenSketch"; "Microsoft.SecHealthUI";
"Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.WebMediaExtensions";
"Microsoft.WebpImageExtension"; "Microsoft.Whiteboard"; "Microsoft.Windows.Photos"; "Microsoft.WindowsAlarms";
"Microsoft.WindowsCalculator"; "Microsoft.WindowsCamera"; "Microsoft.WindowsFeedbackHub";
"Microsoft.WindowsMaps"; "Microsoft.WindowsNotepad"; "Microsoft.WindowsSoundRecorder";
"Microsoft.Xbox.TCUI"; "Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay";
"Microsoft.ZuneMusic"; "Microsoft.ZuneVideo"; "MicrosoftCorporationII.QuickAssist";

      )

    }

    @{

      Name = @( "CloudEditionN"; )

      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.Services.Store.Engagement";
"Microsoft.XboxSpeechToTextOverlay"; "Clipchamp.Clipchamp"; "Microsoft.BingNews"; "Microsoft.BingWeather";
"Microsoft.DesktopAppInstaller"; "Microsoft.GetHelp"; "Microsoft.Getstarted"; "Microsoft.MicrosoftOfficeHub";
"Microsoft.MicrosoftStickyNotes"; "Microsoft.MinecraftEducationEdition"; "Microsoft.Paint"; "Microsoft.ScreenSketch";
"Microsoft.SecHealthUI"; "Microsoft.StorePurchaseApp"; "Microsoft.Whiteboard"; "Microsoft.Windows.Photos";
"Microsoft.WindowsAlarms"; "Microsoft.WindowsCalculator"; "Microsoft.WindowsCamera";
"Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsMaps"; "Microsoft.WindowsNotepad";
"Microsoft.XboxIdentityProvider"; "MicrosoftCorporationII.QuickAssist";

        )

    }

    @{

        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.MicrosoftStickyNotes"; "Microsoft.Paint";
"Microsoft.PowerAutomateDesktop"; "Microsoft.ScreenSketch"; "Microsoft.WindowsNotepad";
"Microsoft.WindowsTerminal"; "Clipchamp.Clipchamp"; "Microsoft.MicrosoftSolitaireCollection";
"Microsoft.WindowsAlarms"; "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsMaps"; "Microsoft.ZuneMusic";
"Microsoft.BingNews"; "Microsoft.BingWeather"; "Microsoft.DesktopAppInstaller"; "Microsoft.WindowsCamera";
"Microsoft.Getstarted"; "Microsoft.Cortana"; "Microsoft.GetHelp"; "Microsoft.MicrosoftOfficeHub"; "Microsoft.People";
"Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.WebMediaExtensions"; "Microsoft.Windows.Photos";
"Microsoft.WindowsCalculator"; "Microsoft.windowscommunicationsapps"; "Microsoft.WindowsSoundRecorder";
"Microsoft.Xbox.TCUI"; "Microsoft.XboxGameOverlay"; "Microsoft.XboxGamingOverlay";
"Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay"; "Microsoft.YourPhone";
"Microsoft.ZuneVideo"; "MicrosoftCorporationII.QuickAssist"; "MicrosoftWindows.Client.WebExperience";
"Microsoft.RawImageExtension"; "MicrosoftCorporationII.MicrosoftFamily";

        )

    }

    @{

        Name = @(

            "Education"; "Professional"; "ProfessionalEducation"; "ProfessionalWorkstation"; "Enterprise"; "IoTEnterprise";
"ServerRdsh";

        )

        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.MicrosoftStickyNotes"; "Microsoft.Paint";
```

```
"Microsoft.PowerAutomateDesktop"; "Microsoft.ScreenSketch"; "Microsoft.WindowsNotepad";
"Microsoft.WindowsTerminal"; "Clipchamp.Clipchamp"; "Microsoft.MicrosoftSolitaireCollection";
"Microsoft.WindowsAlarms"; "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsMaps"; "Microsoft.ZuneMusic";
"Microsoft.BingNews"; "Microsoft.BingWeather"; "Microsoft.DesktopAppInstaller"; "Microsoft.WindowsCamera";
"Microsoft.Getstarted"; "Microsoft.Cortana"; "Microsoft.GetHelp"; "Microsoft.MicrosoftOfficeHub"; "Microsoft.People";
"Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.WebMediaExtensions"; "Microsoft.Windows.Photos";
"Microsoft.WindowsCalculator"; "Microsoft.windowscommunicationsapps"; "Microsoft.WindowsSoundRecorder";
"Microsoft.Xbox.TCUI"; "Microsoft.XboxGameOverlay"; "Microsoft.XboxGamingOverlay";
"Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay"; "Microsoft.YourPhone";
"Microsoft.ZuneVideo"; "MicrosoftCorporationII.QuickAssist"; "MicrosoftWindows.Client.WebExperience";
"Microsoft.RawImageExtension";
        )
    }
    @{
      Name = @(
        "EnterpriseN"; "EnterpriseGN"; "EnterpriseSN"; "ProfessionalN"; "EducationN"; "ProfessionalWorkstationN";
"ProfessionalEducationN"; "CloudN"; "CloudEN"; "CloudEditionLN"; "StarterN";
        )
        Apps = @(
          "Microsoft.UI.Xaml.2.3"; "Microsoft.UI.Xaml.2.4"; "Microsoft.UI.Xaml.2.7";
"Microsoft.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.MicrosoftStickyNotes"; "Microsoft.Paint"; "Microsoft.PowerAutomateDesktop"; "Microsoft.ScreenSketch";
"Microsoft.WindowsNotepad"; "Microsoft.WindowsTerminal"; "Clipchamp.Clipchamp";
"Microsoft.MicrosoftSolitaireCollection"; "Microsoft.WindowsAlarms"; "Microsoft.WindowsFeedbackHub";
"Microsoft.WindowsMaps"; "Microsoft.BingNews"; "Microsoft.BingWeather"; "Microsoft.DesktopAppInstaller";
"Microsoft.WindowsCamera"; "Microsoft.Getstarted"; "Microsoft.Cortana"; "Microsoft.GetHelp";
"Microsoft.MicrosoftOfficeHub"; "Microsoft.People"; "Microsoft.StorePurchaseApp"; "Microsoft.Todos";
"Microsoft.Windows.Photos"; "Microsoft.WindowsCalculator"; "Microsoft.windowscommunicationsapps";
"Microsoft.XboxGameOverlay"; "Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay";
"Microsoft.YourPhone"; "MicrosoftCorporationII.QuickAssist"; "MicrosoftWindows.Client.WebExperience";
        )
    }
  )
  Rule = @(
    @{ Name="Microsoft.UI.Xaml.2.3"; Match="UI.Xaml*{ARCHTag}*2.3";License="UI.Xaml*{ARCHTag}*2.3";
Dependencies=@(); }
    @{ Name="Microsoft.UI.Xaml.2.4"; Match="UI.Xaml*{ARCHTag}*2.4";License="UI.Xaml*{ARCHTag}*2.4";
Dependencies=@(); }
    @{ Name="Microsoft.UI.Xaml.2.7"; Match="UI.Xaml*{ARCHTag}*2.7";License="UI.Xaml*{ARCHTag}*2.7";
Dependencies=@(); }
    @{ Name="Microsoft.NET.Native.Framework.2.2";
Match="Native.Framework*{ARCHTag}*2.2";License="Native.Framework*{ARCHTag}*2.2"; Dependencies=@(); }
    @{ Name="Microsoft.NET.Native.Runtime.2.2";
Match="Native.Runtime*{ARCHTag}*2.2";License="Native.Runtime*{ARCHTag}*2.2"; Dependencies=@(); }
```

```
    @{ Name="Microsoft.VCLibs.140.00"; Match="VCLibs*{ARCHTag}";License="VCLibs*{ARCHTag}";
Dependencies=@(); }

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

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

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

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

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

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

    @{ Name="Microsoft.WindowsStore"; Match="WindowsStore";License="WindowsStore";
Dependencies=@("Microsoft.UI.Xaml.2.3","Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2"
,"Microsoft.VCLibs.140.00"); }

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

    @{ Name="Microsoft.MicrosoftStickyNotes"; Match="Microsoft.Sticky.Notes";License="MicrosoftStickyNotes";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.140.
00"); }

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

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

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

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

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

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

    @{ Name="Microsoft.MicrosoftSolitaireCollection";
Match="MicrosoftSolitaireCollection";License="MicrosoftSolitaireCollection";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.140.
00"); }

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

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

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

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

    @{ Name="MicrosoftCorporationII.MicrosoftFamily"; Match="MicrosoftFamily";License="MicrosoftFamily";
Dependencies=@(); }

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

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

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

    @{ Name="Microsoft.WindowsCamera"; Match="WindowsCamera";License="WindowsCamera";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.140.
00"); }

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

    @{ Name="Microsoft.Cortana"; Match="Cortana";License="Cortana";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.140.
00","Microsoft.VCLibs.140.00.UWPDesktop"); }

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

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

    @{ Name="Microsoft.People"; Match="People";License="People";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.140.
00"); }

    @{ Name="Microsoft.StorePurchaseApp"; Match="StorePurchaseApp";License="StorePurchaseApp";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.140.
00"); }

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

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

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

```
    @{ Name="Microsoft.WindowsCalculator"; Match="WindowsCalculator";License="WindowsCalculator";
Dependencies=@("Microsoft.UI.Xaml.2.4","Microsoft.VCLibs.140.00"); }

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

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

    @{ Name="Microsoft.Xbox.TCUI"; Match="Xbox.TCUI";License="Xbox.TCUI";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.140.
00"); }

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

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

    @{ Name="Microsoft.XboxIdentityProvider"; Match="XboxIdentityProvider";License="XboxIdentityProvider";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.140.
00"); }

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

    @{ Name="Microsoft.YourPhone"; Match="YourPhone";License="YourPhone";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.140.
00","Microsoft.VCLibs.140.00.UWPDesktop"); }

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

    @{ Name="MicrosoftCorporationII.QuickAssist"; Match="QuickAssist";License="QuickAssist";
Dependencies=@(); }

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

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

    @{ Name="Microsoft.Whiteboard"; Match="Whiteboard";License="Whiteboard";
Dependencies=@("Microsoft.NET.Native.Framework.2.2","Microsoft.NET.Native.Runtime.2.2","Microsoft.VCLibs.140.
00"); }

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

  )

}

$Allow_Install_App = @()

ForEach ($item in $Pre_Config_Rules.Edition) {

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

```
      Write-host "`n   Match to: "-NoNewline; Write-host $Current_Edition_Version -ForegroundColor Green
      $Allow_Install_App = $item.Apps
      break
    }
}
Write-host "`n   The app to install ( $($Allow_Install_App.Count) item )" -ForegroundColor Yellow
Write-host "   $('-' * 80)"
ForEach ($item in $Allow_Install_App) {
   Write-host "   $($item)" -ForegroundColor Green
}
Function Match_InBox_Apps_Install_Pack
{
   param ( $NewPath )
   $NewArch  = $Arch
   $NewArchC = $Arch.Replace("AMD64", "x64")
   $NewArchCTag = $Arch.Replace("AMD64", "x64")
   if ($Arch -eq "arm64") { $NewArchCTag = "arm" }
   if ($Pre_Config_Rules.Rule.Count -gt 0) {
      ForEach ($itemInBoxApps in $Pre_Config_Rules.Rule){
         $InstallPacker = ""
         $InstallPackerCert = ""
         $SearchNewStructure = $itemInBoxApps.Match.Replace("{ARCH}", $NewArch).Replace("{ARCHC}",
$NewArchC).Replace("{ARCHTag}", $NewArchCTag)
         $SearchNewLicense = $itemInBoxApps.License.Replace("{ARCH}", $NewArch).Replace("{ARCHC}",
$NewArchC).Replace("{ARCHTag}", $NewArchCTag)
         Get-ChildItem -Path $NewPath -Filter "*$($SearchNewStructure)*" -Include "*.appx", "*.appxbundle",
"*.msixbundle" -Recurse -Force -ErrorAction SilentlyContinue | ForEach-Object {
            if (Test-Path -Path $_.FullName -PathType Leaf) {
               $InstallPacker = $_.FullName
               Get-ChildItem -Path $NewPath -Filter "*$($SearchNewLicense)*" -Include *.xml -Recurse -Force -ErrorAction
SilentlyContinue | ForEach-Object {
                  $InstallPackerCert = $_.FullName
               }
               $Script:InBoxAppx += @{
                  Name         = $itemInBoxApps.Name;
                  Depend       = $itemInBoxApps.Dependencies;
```

```
                Search        = $SearchNewStructure;
                InstallPacker   = $InstallPacker;
                Certificate    = $InstallPackerCert
                CertificateRule = $SearchNewLicense
            }
            return
        }
    }
  }
}
Write-host "`n   InBox Apps: Installation packages, automatic search for full disk or specified paths" -ForegroundColor Yellow
Write-host "   $('-' * 80)"
$Script:InBoxAppx = @()
if ($ISO -eq "Auto") {
   Get-PSDrive -PSProvider FileSystem -ErrorAction SilentlyContinue | ForEach-Object {
      $AppPath = Join-Path -Path $_.Root -ChildPath "packages" -ErrorAction SilentlyContinue
      Match_InBox_Apps_Install_Pack -NewPath $AppPath
   }
} else {
   Match_InBox_Apps_Install_Pack -NewPath $ISO
}
Write-host "   Search Complete" -ForegroundColor Green
Write-host "`n   InBox Apps: Installer Match Results" -ForegroundColor Yellow
Write-host "   $('-' * 80)"
if ($Script:InBoxAppx.Count -gt 0) {
   Write-host "   Match successful" -ForegroundColor Green
} else {
   Write-host "   Failed match" -ForegroundColor Red
   return
}
Write-host "`n   InBox Apps: Details of the application to be installed ( $($Script:InBoxAppx.Count) item )" -ForegroundColor Yellow
Write-host "   $('-' * 80)"
```

```
ForEach ($Rule in $Script:InBoxAppx) {
   Write-host "   Apps name: " -NoNewline; Write-host $Rule.Name -ForegroundColor Yellow
   Write-host "   Apps installer: " -NoNewline; Write-host $Rule.InstallPacker -ForegroundColor Yellow
   Write-host "   License: " -NoNewline; Write-host $Rule.Certificate -ForegroundColor Yellow
   Write-host ""
}
Write-host "`n   InBox Apps: Installation" -ForegroundColor Yellow
Write-host "   $('-' * 80)"
ForEach ($Rule in $Script:InBoxAppx) {
   Write-host "   Name: " -NoNewline; Write-host $Rule.Name -ForegroundColor Yellow
   Write-host "   $('-' * 80)"
  if($Allow_Install_App -contains $Rule.Name) {
    Write-host "   Search for apps: " -NoNewline; Write-host $Rule.InstallPacker -ForegroundColor Yellow
    Write-host "   Search for License: " -NoNewline; Write-host $Rule.Certificate -ForegroundColor Yellow
    if (Test-Path -Path $Rule.InstallPacker -PathType Leaf) {
      if (Test-Path -Path $Rule.Certificate -PathType Leaf) {
        Write-host "   License: " -NoNewline
        Write-host $Rule.Certificate -ForegroundColor Yellow
        Write-host "   With License".PadRight(22) -NoNewline -ForegroundColor Green
        Write-host "   Installing".PadRight(22) -NoNewline
        try {
          Add-AppxProvisionedPackage -Path $Mount -PackagePath $Rule.InstallPacker -LicensePath $Rule.Certificate
-ErrorAction SilentlyContinue | Out-Null
          Write-Host "Done`n" -ForegroundColor Green
        } catch {
          Write-Host "Failed" -ForegroundColor Red
          Write-Host "   $($_)`n" -ForegroundColor Red
        }
      } else {
        Write-host "   No License".PadRight(22) -NoNewline -ForegroundColor Red
        Write-host "   Installing".PadRight(22) -NoNewline
        try {
          Add-AppxProvisionedPackage -Path $Mount -PackagePath $Rule.InstallPacker -SkipLicense -ErrorAction
SilentlyContinue | Out-Null
          Write-Host "Done`n" -ForegroundColor Green
```

```
            } catch {
                Write-Host "Failed" -ForegroundColor Red
                Write-Host "   $($_)`n" -ForegroundColor Red
            }
        }
    } else {
        Write-host "   The installation package does not exist" -ForegroundColor Red
    }
} else {
    Write-host "   Skip the installation`n" -ForegroundColor Red
}
```

5.5. InBox Apps: optimization

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

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

6. Cumulative updates

- When upgrading different versions or old versions to the latest version, you need to add the "Function Enablement Package" first before adding the latest cumulative update;
- After adding a language pack, you can install the same cumulative update as the initial version to resolve a known issue where the "Components: All packages installed in the image" status is not refreshed after installation;
- To stay up to date, it is recommended that you download the latest version.

6.1. Initial version

6.1.1. How to download

Cumulative update KB5016632 is no longer searchable from https://www.catalog.update.microsoft.com/Search.aspx?q=KB5016632.

6.1.1.1. Select any website and open:

6.1.1.1.1. https://uupdump.net

6.1.1.1.2. https://uup.ee

6.1.1.1.3. https://osdump.com

6.1.1.2. After opening, search for keywords: 22621.382, select from the search results: Windows 11, version 22H2 (22621.382) amd64 and Windows 11, version 22H2 (22621.382) arm64

6.1.1.3. After opening, select "All files";

6.1.1.4. Search for: KB5016632 in the "All Files" page, download:

- Windows11.0-KB5016632-x64.cab
- Windows11.0-KB5016632-x64.psf

6.1.1.5. Cumulative updates: Merging

6.1.1.5.1. Go to https://github.com/abbodi1406/WHD/tree/master/scripts After downloading PSFX_Repack_6.zip

6.1.1.5.2. After downloading, unzip it to: D:\PSFX_Repack_6

6.1.1.5.3. Copy Windows11.0-KB5016632-x64.cab, Windows11.0-KB5016632-x64.psf to: D:\PSFX_Repack_6

6.1.1.5.4. After running: psfx2cab_GUI.cmd, you will get a new file: Windows11.0-KB5016632-x64-full_psfx.cab

6.1.1.5.5. Save Windows11.0-KB5016632-x64-full_psfx.cab to: D:\OS_11_Custom\Install\Install\Update\Windows11.0-KB5016632-x64-full_psfx.cab

6.1.2. Add

```
$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
```

6.2. Other Version

Check "Windows 11 version information", for example, download cumulative update: KB5035853, Version: 22621.3296, go to the download page: https://www.catalog.update.microsoft.com/Search.aspx?q=KB5035853, download and save to: D:\OS_11_Custom\Install\Install\Update, or download through direct connection, select download according to the architecture:

6.2.1. x64, default

- Direct download

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

- Add to

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

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

6.3. Solidify Updated

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

```
Dism /Image:"D:\OS_11_Custom\Install\Install\Mount" /cleanup-image /StartComponentCleanup /ResetBase
```

6.3.1. Clean components after curing and updating

```
$Mount = "D:\OS_11_Custom\Install\Install\Mount"
Get-WindowsPackage -Path $Mount -ErrorAction SilentlyContinue | ForEach-Object {
  if ($_.PackageState -eq "Superseded") {
    Write-Host "   $($_.PackageName)" -ForegroundColor Green
    Remove-WindowsPackage -Path $Mount -PackageName $_.PackageName | Out-Null
  }
}
```

7. Deployment engine: Add

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

8. Health

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

```
Repair-WindowsImage -Path "D:\OS_11_Custom\Install\Install\Mount" -ScanHealth
```

9. Replace WinRE.wim

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

```
$WinRE = "D:\OS_11_Custom\Install\Install\Update\Winlib\WinRE.wim"
$CopyTo = "D:\OS_11_Custom\Install\Install\Mount\Windows\System32\Recovery"
Copy-Item -Path $WinRE -Destination $CopyTo -Force
```

10. Save image: Install.wim

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

11. Unmount image: Install.wim

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

```
Dismount-WindowsImage -Path "D:\OS_11_Custom\Install\Install\Mount" -Discard
```

LOOP OPERATING AREA, END.

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

12.1. Obtain WimLib

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

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

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

- Install.WinRE.Extract.ps1
  - \Expand\Install\Install.WinRE.Extract.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/22H2/Expand/Install/Install.WinRE.Extract.ps1

- Copy the code

```
$Arguments = @(
  "extract",
  "D:\OS_11\sources\install.wim", "1",
  "\Windows\System32\Recovery\Winre.wim",
  "--dest-dir=""D:\OS_11_Custom\Install\Install\Update\Winlib"""
)
New-Item -Path "D:\OS_11_Custom\Install\Install\Update\Winlib" -ItemType Directory -ea SilentlyContinue
Start-Process -FilePath "d:\wimlib\wimlib-imagex.exe" -ArgumentList $Arguments -wait -nonewwindow
```

12.2.2. Get all index numbers of Install.wim and replace the old WinRE.wim

- Install.WinRE.Replace.wim.ps1
  - \Expand\Install\Install.WinRE.Replace.wim.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/22H2/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
```

```
}
```

13. Rebuilding Install.wim reduces file size

- Install.Rebuild.wim.ps1
  - \Expand\Install\Install.Rebuild.wim.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/22H2/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
}
```

III Custom encapsulation boot.wim

1. View Boot.wim details

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

   ```
   $ViewFile = "D:\OS_11\Sources\Boot.wim"
   Get-WindowsImage -ImagePath $ViewFile | Foreach-Object { Get-WindowsImage -ImagePath $ViewFile -index $_.ImageIndex }
   ```

2. Specify the path to mount Boot.wim

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

3. Start mounting Boot.wim

   Default index number: 2

```
Mount-WindowsImage -ImagePath "D:\OS_11\sources\boot.wim" -Index "2"  -Path "D:\OS_11_Custom\Boot\Boot\Mount"
```

4. Language pack: Boot

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

4.1. Language pack: Add

- Boot.Instl.lang.ps1
  - \Expand\Boot\Boot.Instl.lang.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/22H2/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
        }
    }
}
```

4.2. Components: All packages installed in the image

4.2.1. View

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

4.2.2. Export to csv

```
$SaveTo = "D:\OS_11_Custom\Boot\Boot\Report.$(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
```

4.3. Language: Repair

4.3.1. Extract

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

4.3.1.1. arunres.dll.mui

4.3.1.2. spwizres.dll.mui

4.3.1.3. w32uires.dll.mui

4.3.2. Copy

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

4.4. Language packs: sync to ISO installer

```
Copy-Item -Path "D:\OS_11_Custom\Boot\Boot\Mount\sources\zh-CN" -Destination "D:\OS_11\sources\zh-CN" -Recurse -Force
```

4.5. Regenerate Lang.ini

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

4.5.1. Regenerate the mounted directory lang.ini

Regenerated Lang.ini file location: D:\OS_11_Custom\Boot\Boot\Mount\Sources\lang.ini

```
Dism /image:"D:\OS_11_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS_11_Custom\Boot\Boot\Mount"
```

4.5.2. After regenerating lang.ini, sync to the installer

Regenerated Lang.ini file location: D:\OS_11\Sources\lang.ini

```
Dism /image:"D:\OS_11_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS_11"
```

5. Other

5.1. Bypass TPM check during installation

- Boot.Bypass.TPM.ps1
  - \Expand\Boot\Boot.Bypass.TPM.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/22H2/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
```

6. Save image: Boot.wim

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

7. Unmount image: Boot.wim

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

```
Dismount-WindowsImage -Path "D:\OS_11_Custom\Boot\Boot\Mount" -Discard
```

IV Deployment engine

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

1. Add method

1.1. Add to ISO installation media

1.1.1. Unattended

1.1.1.1. Add to: [ISO]:\Autounattend.xml

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

Copy D:\Multilingual_Learn\Unattend\Mul.Unattend.xml to D:\OS_11\Autounattend.xml

```
Copy-Item "D:\Multilingual\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS_11\Autounattend.xml" -Force
```

1.1.1.2. Add to: [ISO]:\Sources\Unattend.xml

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

Copy D:\Multilingual_Learn\Unattend\Mul.Unattend.xml to D:\OS_11\Sources\Unattend.xml

```
Copy-Item "D:\Multilingual\_Learn\Unattend\Mul.Unattend.xml" -Destination
"D:\OS_11\Sources\Unattend.xml" -Force
```

1.1.1.3. Add to: [ISO]:\sources\$OEM$\$$\Panther\unattend.xml

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

1.1.1.3.1. Create OEM path

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

1.1.1.3.2. Copy

Copy D:\Multilingual_Learn\Unattend\Mul.Unattend.xml to D:\OS_11\Sources\OEM\Panther\Unattend.xml

```
Copy-Item "D:\Multilingual\_Learn\Unattend\Mul.Unattend.xml" -Destination
"D:\OS_11\sources\`$OEM$\`$$\Panther\Unattend.xml" -Force
```

1.1.2. Deployment engine: add

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

1.1.2.1. 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
```

1.1.2.2. Deployment engine: custom deployment tags

```
$Flag = @(
  "Is_Mark_Sync" # Allow global search and synchronization of deployment tags
  # Prerequisite deployment
#   "Auto_Update" # Allow automatic updates
#   "Use_UTF8" # Beta: Global language support using Unicode UTF-8
  "Disable_Network_Location_Wizard" # Network Location Wizard
  "Disable_Cleanup_Appx_Tasks" # Appx Cleanup and maintenance tasks
  "Disable_Cleanup_On_Demand_Language" # Prevent cleanup of unused on-demand feature language
packs
```

```
    "Disable_Cleanup_Unsed_Language"     # Prevent cleaning of unused language packs
    "Prerequisites_Reboot" # Restart your computer
    # Complete first deployment
#    "Popup_Engine" # Allow the deployment engine main interface to pop up for the first time
#    "Allow_First_Pre_Experience" # Allow first preview, as planned
    "Reset_Execution_Policy" # Restore PowerShell execution policy: Restricted
    "Clear_Solutions" # Delete the entire solution
    "Clear_Engine" # Delete the deployment engine and keep the others
#    "First_Experience_Reboot" # Restart your computer
)
ForEach ($item in $Flag) {
    Write-host "   $($item)" -ForegroundColor Green
    New-Item -Path "D:\OS_11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -ErrorAction SilentlyContinue | Out-Null
    Out-File -FilePath "D:\OS_11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow\$($item)" -Encoding utf8 -ErrorAction SilentlyContinue
}
```

1.2. Add to mounted

Through "Custom encapsulation: Install.wim", execute "Start mounting Install.wim" and mount to: D:\OS_11_Custom\Install\Install\Mount

1.2.1. Unattended

Copy D:\Multilingual_Learn\Unattend\Mul.Unattend.xml to D:\OS_11_Custom\Install\Install\Mount\Panther\Unattend.xml

```
Copy-Item "D:\Multilingual\_Learn\Unattend\Mul.Unattend.xml" -Destination "D:\OS_11_Custom\Install\Install\Mount\Panther" -Force
```

1.2.2. Deployment engine: add

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

1.2.2.1. Deployment Engine: Copy

Copy D:\Multilingual\Engine to D:\OS_11_Custom\Install\Install\Mount\Yi\Engine

```
Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS_11_Custom\Install\Install\Mount\Yi\Engine" -Recurse -Force
```

1.2.2.2. Deployment engine: custom deployment tags

```
$Flag = @(
   "Is_Mark_Sync" # Allow global search and synchronization of deployment tags
   # Prerequisite deployment
#    "Auto_Update" # Allow automatic updates
#    "Use_UTF8" # Beta: Global language support using Unicode UTF-8
   "Disable_Network_Location_Wizard" # Network Location Wizard
   "Disable_Cleanup_Appx_Tasks" # Appx Cleanup and maintenance tasks
   "Disable_Cleanup_On_Demand_Language" # Prevent cleanup of unused on-demand feature language packs
   "Disable_Cleanup_Unsed_Language"     # Prevent cleaning of unused language packs
   "Prerequisites_Reboot" # Restart your computer
   # Complete first deployment
#    "Popup_Engine" # Allow the deployment engine main interface to pop up for the first time
#    "Allow_First_Pre_Experience" # Allow first preview, as planned
   "Reset_Execution_Policy" # Restore PowerShell execution policy: Restricted
   "Clear_Solutions" # Delete the entire solution
   "Clear_Engine" # Delete the deployment engine and keep the others
#    "First_Experience_Reboot" # Restart your computer
)
ForEach ($item in $Flag) {
   Write-host "   $($item)" -ForegroundColor Green
   New-Item -Path "D:\OS_11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -ErrorAction SilentlyContinue | Out-Null
   Out-File -FilePath "D:\OS_11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow\$($item)" -Encoding utf8 -ErrorAction SilentlyContinue
}
```

2. Deployment Engine: Advanced

2.1. Deployment engine: adding process

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

2.2. Unattended solution

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

- D:\OS_11\Autounattend.xml
- D:\OS_11\Sources\Unattend.xml
- D:\OS_11\sources\$OEM$\$$\Panther\unattend.xml

- D:\OS_11_Custom\Install\Install\Mount\Panther\Unattend.xml

2.2.1. Multilingual or monolingual

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

2.2.1.1. Multi-language

```
<UILanguage>%OSDUILanguage%</UILanguage>
<InputLocale>%OSDInputLocale%</InputLocale>
<SystemLocale>%OSDSystemLocale%</SystemLocale>
<UILanguage>%OSDUILanguage%</UILanguage>
<UILanguageFallback>%OSDUILanguageFallback%</UILanguageFallback>
<UserLocale>%OSDUserLocale%</UserLocale>
```

2.2.1.2. Monolingual

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

```
<UILanguage>zh-CN</UILanguage>
<InputLocale>zh-CN</InputLocale>
<SystemLocale>zh-CN</SystemLocale>
<UILanguage>zh-CN</UILanguage>
<UILanguageFallback>zh-CN</UILanguageFallback>
<UserLocale>zh-CN</UserLocale>
```

2.2.2. User plan

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

2.2.2.1. Self-created user Administrator

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

```
<UserAccounts>
  <LocalAccounts>
    <LocalAccount wcm:action="add">
      <Password>
        <Value></Value>
        <PlainText>true</PlainText>
      </Password>
```

```
        <Description>Administrator</Description>
        <DisplayName>Administrator</DisplayName>
        <Group>Administrators</Group>
        <Name>Administrator</Name>
      </LocalAccount>
    </LocalAccounts>
  </UserAccounts>
  <AutoLogon>
    <Password>
      <Value></Value>
      <PlainText>true</PlainText>
    </Password>
    <Enabled>true</Enabled>
    <Username>Administrator</Username>
  </AutoLogon>
```

2.2.2.2. Custom user

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

2.2.2.2.1. Delete

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

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

2.2.2.2.2. Replace

From the beginning <OOBE> to </OOBE>

```
<OOBE>
  <ProtectYourPC>3</ProtectYourPC>
  <HideEULAPage>true</HideEULAPage>
  <HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>
</OOBE>
```

D. ISO

II Generate ISO

1. Download OScdimg

Select the Oscdimg version according to the architecture, and save it to: D:\ after downloading. To save in other paths, please enter the

absolute path of OScdimg.exe;

1.1. x64

https://github.com/ilikeyi/solutions/raw/main/_Software/Oscdimg/amd64/oscdimg.exe

1.2. x86

https://github.com/ilikeyi/solutions/raw/main/_Software/Oscdimg/x86/oscdimg.exe

1.3. arm64

https://github.com/ilikeyi/solutions/raw/main/_Software/Oscdimg/arm64/oscdimg.exe

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

- ISO.ps1
  - \Expand\ISO.ps1
  - https://github.com/ilikeyi/solutions/blob/main/_Learn/Packaging.tutorial/OS.11/22H2/Expand/ISO.ps1

- Copy the code

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

$ISO = "D:\OS_11"

$Volume = "OS_11"

$SaveTo = "D:\OS_11.iso"

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

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

III Bypass TPM installation check

1. Learn about: https://github.com/AveYo/MediaCreationTool.bat/tree/main/bypass11 and download: Quick_11_iso_esd_wim_TPM_toggle.bat

2. Drag the D:\OS_11.iso file to Quick_11_iso_esd_wim_TPM_toggle.bat, and "add" or "delete" the TPM installation check function in reverse order.

Yi' Solutions

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

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

Author: Yi

EMail: 775159955@qq.com, ilikeyi@outlook.com

Document version: 1.1

Documentation model: Lite version

Translation: Chinese to English version

Updated: 2024 - 6

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