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?

TABLE OF CONTENTS

ONE.   DEPLOYMENT IMAGE

    A.   Prerequisites

        II   **ISO tools**

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

        III   **Get related packages**

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

        1.   **System installation package**

        1.1.   Prepare: en-us_windows_11_business_editions_version_22h2_x64_dvd_17a08ce3.iso

        1.2.   Unzip to: D:\en-us_windows_11_business_editions_version_22h2_x64_dvd_17a08c

        1.3.   After decompression is complete, change the directory en-us_windows_11_business_editions_version_22h2_x64_dvd_17a08c change to D:\OS11

        1.4.   All scripts and all paths are set to D:\OS11 as the image source.

        2.   **Language pack**

        2.1.   **Learn**

        1.1.1.   Add languages to a Windows 11 image

        1.1.2.   Language and region Features on Demand (FOD)

        1.1.2.1.   **Language tag association**

What is the language tag association?

- When the image language is only in English, after adding the zh-HK language pack, the image language will not add a new language tag. 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 language tag associations:

SN	language tag	Optional language affinity tag	Remark
1	zh-TW	Zh-HK	

1.1.2.2. Fonts

When adding a language pack, when the corresponding language tag is triggered, you need to add the required font function, download "List of all available language FODs" learn more.

In "Language Pack: Extraction", the automatic recognition function has been added, and you can understand the functions: Function Match_Required_Fonts

1.1.2.3. Other region-specific requirements

Region	Package	Description	Recommendation
zh-TW	Microsoft-Windows-InternationalFeatures-Taiwan-Package~31bf3856ad364e35~amd64~~.cab	Supplemental support for Taiwan date formatting requirements. Package will be provided to customers located in Taiwan.	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.

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

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. Is it consistent with the official Microsoft original version, or is it added during third-party production? , it is up to the encapsulator to freely choose whether to install it or not.

2.2. Language pack: Download

22621.1.220506-1250.ni_release_amd64fre_CLIENT_LOF_PACKAGES_OEM.iso

2.3. Language pack: Fixed

Select any website and open:

- https://uupdump.net

- https://uup.ee

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

amd64

2.3.2. After opening, select "All files";

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

2.3.3.1. Applies to: Install.wim （1 item）

2.3.3.1.1. Media

2.3.3.2. Applies to: WinRE.wim, none yet

2.3.3.3. Applies to: Boot.wim, none yet

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

2.4. 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: 22621.1778.230511-2102.ni_release_svc_prod3_amd64fre_InboxApps.iso

3.2. Download: 22621.1.220506-1250.ni_release_amd64fre_InboxApps.iso finally, extract the following files and save them to the desktop:

3.2.1. Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.x64.appx

3.2.2. 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 When running the PS 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.

B. **Language Pack: Extraction**

II **Prepare the language pack**

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

III   **Extract language pack scheme**

    1.   **Add**

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

    2.   **Delete**

        2.1.   Language name: English - United States, language tag: 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:\"

Copy the code or view the source file: Extract.ps1 (Local, Github)

```
$ISO = "Auto"
$SaveTo = "D:\OS11_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"; }
```

```powershell
      @{ 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 )

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

    ForEach ($item in $Fonts) {
      if (($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 {
```

```powershell
        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 = @(
```

```powershell
          "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.   **Customize the deployment image**

II   **Custom deployment image: Install.wim**

   1.   **View Install.wim details**

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

```
$ViewFile = "D:\OS11\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:\OS11_Custom\Install\Install\Mount" -ItemType directory -ea SilentlyContinue
```

   3.   **Start mounting Install.wim**

Default index number: 1

```
Mount-WindowsImage -ImagePath "D:\OS11\sources\install.wim" -Index "1" -Path "D:\OS11_Custom\Install\Install\Mount"
```

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

   3.1.   **Custom deployment image: 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:\OS11_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:\OS11_Custom\Install\WinRE\Mount" -ItemType directory -ea SilentlyContinue

3.1.3. Start mounting WinRE.wim

Default index number: 1

Mount-WindowsImage -ImagePath "D:\OS11_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim" -Index "1" -Path "D:\OS11_Custom\Install\WinRE\Mount"

3.1.4. 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.

3.1.4.1. Language pack: add

Copy the code or view the source file: WinRE.Instl.lang.ps1 (Local, Github)

```
$Mount = "D:\OS11_Custom\Install\WinRE\Mount"
$Sources = "D:\OS11_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) {
```

```powershell
      Write-host "`n  Rule name: $($Rule.Match)" -ForegroundColor Yellow; Write-host "  $('-' * 80)"
      ForEach ($Component in $Initl_install_Language_Component) {
        if ($Component -like "*$($Rule.Match)*") {
          Write-host "  Component name: " -NoNewline
          Write-host $Component -ForegroundColor Green
          Write-host "  Language pack file: " -NoNewline
          Write-host "$($Sources)\$($Rule.File)" -ForegroundColor Green

          Write-Host "  Installing ".PadRight(22) -NoNewline
          try {
            Add-WindowsPackage -Path $Mount -PackagePath "$($Sources)\$($Rule.File)" | Out-Null
            Write-host "Finish" -ForegroundColor Green
          } catch {
            Write-host "Failed" -ForegroundColor Red
          }
          break
        }
      }
}
```

3.1.4.2. Offline image language: change

3.1.4.2.1. Change default language, regional settings, and other international settings

Language Tag: zh-CN

```
Dism /Image:"D:\OS11_Custom\Install\WinRE\Mount" /Set-AllIntl:zh-CN
```

3.1.4.2.2. View available language settings

```
Dism /Image:"D:\OS11_Custom\Install\WinRE\Mount" /Get-Intl
```

3.1.4.3. Language packs: Removed, optional

- 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 language tag: en-US, you need to extract the language pack in advance

Copy the code or view the source files: WinRE.Del.Specified.lang.Tag.ps1 (Local, Github)

```powershell
$Lang = "en-US"
$Mount = "D:\OS11_Custom\Install\WinRE\Mount"
$Sources = "D:\OS11_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"; }
```

```powershell
    @{ 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
    }
  }
}
```

3.1.4.4. Components: All packages installed in the image

3.1.4.4.1. View

```
Get-WindowsPackage -Path "D:\OS11_Custom\Install\WinRE\Mount" | Out-GridView
```

3.1.4.4.2. Export to csv

```
$SaveTo = "D:\OS11_Custom\Install\WinRE\Report.$(Get-Date -Format
"yyyyMMddHHmmss").csv"
Get-WindowsPackage -Path "D:\OS11_Custom\Install\WinRE\Mount" | Export-CSV -
NoType -Path $SaveTo
Write-host $SaveTo -ForegroundColor Green
```

3.1.5. Cumulative updates, optional

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

3.1.5.1. Add

```
$KBPath = "D:\OS11_Custom\Install\WinRE\Update\KB_WinRE.cab"
Add-WindowsPackage -Path "D:\OS11_Custom\Install\WinRE\Mount" -PackagePath $KBPath
```

3.1.5.2. Delete

```
$KBPath = "D:\OS11_Custom\Install\WinRE\Update\KB_WinRE.cab"
Remove-WindowsPackage -Path "D:\OS11_Custom\Install\WinRE\Mount" -PackagePath $KBPath
```

3.1.5.3. 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:\OS11_Custom\Install\WinRE\Mount" /cleanup-image /StartComponentCleanup
/ResetBase
```

3.1.5.3.1. Clean components after curing and updating, optional

```
$Mount = "D:\OS11_Custom\Install\WinRE\Mount"
Get-WindowsPackage -Path $Mount -ErrorAction SilentlyContinue | ForEach-Object {
  if ($_.PackageState -eq "Superseded") {
     Write-Host "  $($_.PackageName)" -ForegroundColor Green
      Remove-WindowsPackage -Path $Mount -PackageName $_.PackageName | Out-Null
   }
}
```

3.1.6. Driver, optional

3.1.7. Save image

Save-WindowsImage -Path "D:\OS11_Custom\Install\WinRE\Mount"

3.1.8. Unmount image

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

Dismount-WindowsImage -Path "D:\OS11_Custom\Install\WinRE\Mount" -Discard

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

Copy the code or view the source file: WinRE.Rebuild.ps1 (Local, Github)

```
$FileName = "D:\OS11_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
}
```

3.1.10. Backup WinRE.wim

Copy the code or view the source file: WinRE.Backup.ps1 (Local, Github)

```
$WimLibPath = "D:\OS11_Custom\Install\Install\Update\Winlib"
$FileName = "D:\OS11_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"
New-Item -Path $WimLibPath -ItemType Directory -ea SilentlyContinue
Copy-Item -Path $FileName -Destination $WimLibPath -Force
```

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

- After each mount Install.wim "Replace WinRE.wim";

- Learn "Get all index numbers of Install.wim and replace the old WinRE.wim".

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

4. **Language pack**

Automatically install language packs: Get "Component: All installed packages in the image" and match them. After matching the corresponding names, install the local corresponding language pack files.

4.1. **Language pack: add**

Copy the code or view the source file: Install.Instl.lang.ps1 (Local, Github)

```
$Mount = "D:\OS11_Custom\Install\Install\Mount"
$Sources = "D:\OS11_Custom\Install\Install\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)\Microsoft-Windows-LanguageFeatures-Fonts-Hans-Package~31bf3856ad364e35~amd64~~.cab"

$Language_List = @(
    @{ Match = "*Client-LanguagePack-Package*";  File = "Microsoft-Windows-Client-Language-Pack_x64_zh-CN.cab"; }
    @{ Match = "*LanguageFeatures-Basic*"; File = "Microsoft-Windows-LanguageFeatures-Basic-zh-CN-Package~31bf3856ad364e35~amd64~~.cab"; }
    @{ Match = "*Handwriting*"; File = "Microsoft-Windows-LanguageFeatures-Handwriting-zh-CN-Package~31bf3856ad364e35~amd64~~.cab"; }
    @{ Match = "*OCR*"; File = "Microsoft-Windows-LanguageFeatures-OCR-zh-CN-Package~31bf3856ad364e35~amd64~~.cab"; }
    @{ Match = "*LanguageFeatures-Speech*"; File = "Microsoft-Windows-LanguageFeatures-Speech-zh-CN-Package~31bf3856ad364e35~amd64~~.cab"; }
    @{ Match = "*TextToSpeech*"; File = "Microsoft-Windows-LanguageFeatures-TextToSpeech-zh-CN-Package~31bf3856ad364e35~amd64~~.cab"; }
    @{ Match = "*InternetExplorer*"; File = "Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~AMD64~zh-CN~.cab"; }
    @{ Match = "*Notepad*amd64*"; File = "Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~AMD64~zh-CN~.cab"; }
    @{ Match = "*Notepad*wow64*"; File = "Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~wow64~zh-CN~.cab"; }
    @{ Match = "*MediaPlayer*amd64*"; File = "Microsoft-Windows-MediaPlayer-Package-AMD64-zh-CN.cab"; }
    @{ Match = "*MediaPlayer*wow64*"; File = "Microsoft-Windows-MediaPlayer-Package-wow64-zh-CN.cab"; }
    @{ Match = "*PowerShell*amd64*"; File = "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~zh-CN~.cab"; }
    @{ Match = "*PowerShell*wow64*"; File = "Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~zh-CN~.cab"; }
    @{ Match = "*Printing*PMCPPC*amd64*"; File = "Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~AMD64~zh-CN~.cab"; }
    @{ Match = "*StepsRecorder*amd64*"; File = "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~zh-CN~.cab"; }
    @{ Match = "*StepsRecorder*wow64*"; File = "Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~zh-CN~.cab"; }
    @{ Match = "*WMIC*FoD*Package*amd64*"; File = "Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~AMD64~zh-CN~.cab"; }
    @{ Match = "*WMIC*FoD*Package*wow64*"; File = "Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~zh-CN~.cab"; }
    @{ Match = "*WordPad*amd64*"; File = "Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~zh-CN~.cab"; }
    @{ Match = "*WordPad*wow64*"; File = "Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~zh-
```

```
CN~.cab"; }
    @{ Match = "*Client*LanguagePack*zh-TW*"; File = "Microsoft-Windows-InternationalFeatures-Taiwan-
Package~31bf3856ad364e35~amd64~~.cab"; }
)

ForEach ($Rule in $Language_List) {
  Write-host "`n  Rule name: $($Rule.Match)" -ForegroundColor Yellow; Write-host "  $('-' * 80)"
  ForEach ($Component in $Initl_install_Language_Component) {
    if ($Component -like "*$($Rule.Match)*") {
      Write-host "   Component name: " -NoNewline
      Write-host $Component -ForegroundColor Green
      Write-host "   Language pack file: " -NoNewline
      Write-host "$($Sources)\$($Rule.File)" -ForegroundColor Green

      Write-Host "   Installing ".PadRight(22) -NoNewline
      try {
        Add-WindowsPackage -Path $Mount -PackagePath "$($Sources)\$($Rule.File)" | Out-Null
        Write-host "Finish" -ForegroundColor Green
      } catch {
        Write-host "Failed" -ForegroundColor Red
      }
      break
    }
  }
}
```

4.2. Offline image language: change

- Starting Windows 11, the default System UI Language set by DISM is left unaltered on all editions except for Home edition. For all commercial editions the language chosen during the Out-of-Box Experience (OOBE) is set as the System Preferred UI language and Windows will be displayed in this language and for Home edition the language chosen at OOBE will continue to be the default System UI Language.

- As of Windows 10, version 2004, if an .appx-based Language Experience Pack (LXP) backed language is passed as an argument then the language will be set as the System Preferred UI language and its parent language will be set as the Default System UI language. In prior versions only .cab based language packs were supported.

4.2.1. Change default language, regional settings, and other international settings

Language Tag: zh-CN

Dism /Image:"D:\OS11_Custom\Install\Install\Mount" /Set-AllIntl:zh-CN

4.2.2. View available language settings

Dism /Image:"D:\OS11_Custom\Install\Install\Mount" /Get-Intl

4.3. Language packs: Removed, optional

- 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 language tag: en-US, you need to extract the

language pack in advance

Copy the code or view the source files: Install.Del.Specified.lang.Tag.ps1 (Local, Github)

```powershell
$Lang = "en-US"
$Mount = "D:\OS11_Custom\Install\Install\Mount"
$Sources = "D:\OS11_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)*") {
```

```powershell
        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
        }
    }
}
```

4.4. Components: All packages installed in the image

4.4.1. View

```powershell
Get-WindowsPackage -Path "D:\OS11_Custom\Install\Install\Mount" | Out-GridView
```

4.4.2. Export to csv

```powershell
$SaveTo = "D:\OS11_Custom\Install\Install\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"
Get-WindowsPackage -Path "D:\OS11_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo
Write-host $SaveTo -ForegroundColor Green
```

5. **InBox Apps**

5.1. **Learn "Attachment: inBox Apps"**

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

5.2. **InBox Apps: Installed**

5.2.1. **View**

```
Get-AppXProvisionedPackage -Path "D:\OS11_Custom\Install\Install\Mount" | Out-GridView
```

5.2.2. **Export to Csv**

```
$SaveTo = "D:\OS11_Custom\Install\Install\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"
Get-AppXProvisionedPackage -Path "D:\OS11_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo
Write-host $SaveTo -ForegroundColor Green
```

5.3. **Remove all installed pre-applications**

Copy the code or view the source file: Install.InBox.Appx.Clear.all.ps1 (Local, Github)

```
Get-AppXProvisionedPackage -path "D:\OS11_Custom\Install\Install\Mount" -ErrorAction SilentlyContinue | ForEach-Object {
    Write-host "`n  $($_.DisplayName)"
    Write-Host "   Deleting ".PadRight(22) -NoNewline
    try {
        Remove-AppxProvisionedPackage -Path "D:\OS11_Custom\Install\Install\Mount" -PackageName $_.PackageName -ErrorAction SilentlyContinue | Out-Null
        Write-host "Finish" -ForegroundColor Green
    } catch {
        Write-host "Failed" -ForegroundColor Red
    }
}
```

5.4. **Language Markers: Understanding**

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

View installed languages for offline images

```
Dism /Image:"D:\OS11_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, which is the language tag.

5.5. Language tag: adding method

5.5.1. Execute "Language Pack:Add"

5.5.2. 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:

5.5.2.1. Download using the Windows Local Language Experience Packs (LXPs) Downloader

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

After downloading, save to: D:\OS11_Custom\Install\Install\InBox.Appx, file format:LanguageExperiencePack.zh-CN.Neutral.Appx

5.5.2.2. Manual download

5.5.2.2.1. View "Attachment: LXPs"

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

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

5.5.2.2.2.1.   Search keywords:

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

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

5.5.2.2.2.3.   After downloading, save it to the D:\OS11_Custom\Install\Install\InBox.Appx directory and rename it: LanguageExperiencePack.zh-cn.Neutral.Appx

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

After understanding how to add language tags and obtaining LanguageExperiencePack.zh-cn.Neutral, execute the installation command:

Add-AppxProvisionedPackage -Path "D:\OS11_Custom\Install\Install\Mount" -PackagePath "D:\OS11_Custom\Install\Install\InBox.appx\LanguageExperiencePack.zh-cn.Neutral.appx" -SkipLicense

5.5.2.4. InBox Apps: An installed application package

5.5.2.4.1. View

Get-AppXProvisionedPackage -Path "D:\OS11_Custom\Install\Install\Mount" | Out-GridView

5.5.2.4.2. Export to Csv

$SaveTo = "D:\OS11_Custom\Install\Install\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"
Get-AppXProvisionedPackage -Path "D:\OS11_Custom\Install\Install\Mount" | Export-CSV -

```
                                    NoType -Path $SaveTo
                                    Write-host $SaveTo -ForegroundColor Green
```

5.6.   InBox Apps: Install

5.6.1.   Mount or decompress the InBox Apps installation file

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

5.6.2.   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"

Copy the code or view the source file: Install.Inst.InBox.Appx.ps1 (Local, Github)

```powershell
$ISO = "Auto"

$Mount = "D:\OS11_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";
    )
  }
  @{
```

```powershell
      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)"
```

```powershell
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"; }
```

```powershell
  @{ 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"; }
```

```powershell
)

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
              }
            }
```

5.7. **Language tag: delete**

After completing the installation of the InBox Apps application, the language 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 language tag: zh-CN, can be changed to other language tag.

Copy the code or view the source file: Install.Clear.Flag.ps1 (Local, Github)

```
$Lang = "zh-CN"

Get-AppXProvisionedPackage -Path "D:\OS11_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:\OS11_Custom\Install\Install\Mount" -PackageName $_.PackageName -ErrorAction SilentlyContinue | Out-Null
      Write-host "Finish" -ForegroundColor Green
    } catch {
      Write-host "Failed" -ForegroundColor Red
    }
  }
}
```

5.8. **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:\OS11_Custom\Install\Install\Mount" /Optimize-ProvisionedAppxPackages

6. **Cumulative updates**

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.

6.1. **Initial version**

6.1.1. **How to download**

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

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

6.1.1.2. After opening, select "All files";

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

- Windows11.0-KB5016632-x64.cab

- Windows11.0-KB5016632-x64.psf

6.1.1.4. **Merging cumulative updates**

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

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

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

6.1.1.4.4. After running: psfx2cab_GUI.cmd, you will get a new file:

Windows11.0-KB5016632-x64-full_psfx.cab

6.1.1.4.5. Save Windows11.0-KB5016632-x64-full_psfx.cab to:

D:\OS11_Custom\Install\Install\Update\Windows11.0-KB5016632-x64-full_psfx.cab

6.1.2. **Add**

$KBPath = "D:\OS11_Custom\Install\Install\Update\Windows11.0-KB5016632-x64-full_psfx.cab"

Add-WindowsPackage -Path "D:\OS11_Custom\Install\Install\Mount" -PackagePath $KBPath

6.2. **Other Version**

6.2.1. **How to download**

Check out "Windows 11 version 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", Save to:

D:\OS11_Custom\Install\Install\Update\windoOS11.0-kb5030219-x64_659d407821da654d564a3108bf493594bd143ab1.msu

6.2.2. **Add**

$KBPath = "D:\OS11_Custom\Install\Install\Update\windoOS11.0-kb5030219-x64_659d407821da654d564a3108bf493594bd143ab1.msu"

```
Add-WindowsPackage -Path "D:\OS11_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.3. Solidify Updated, optional

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

```
Dism /Image:"D:\OS11_Custom\Install\Install\Mount" /cleanup-image /StartComponentCleanup /ResetBase
```

6.3.1. Clean components after curing and updating, optional

```
$Mount = "D:\OS11_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. Driver, optional

8. 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.

9. Health

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

```
Repair-WindowsImage -Path "D:\OS11_Custom\Install\Install\Mount" -ScanHealth
```

10. Replace WinRE.wim

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

```
$WinRE = "D:\OS11_Custom\Install\Install\Update\Winlib\WinRE.wim"
$CopyTo = "D:\OS11_Custom\Install\Install\Mount\Windows\System32\Recovery"
Copy-Item -Path $WinRE -Destination $CopyTo -Force
```

11. Save image

```
Save-WindowsImage -Path "D:\OS11_Custom\Install\Install\Mount"
```

12. Unmount image

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

```
Dismount-WindowsImage -Path "D:\OS11_Custom\Install\Install\Mount" -Discard
```

LOOP OPERATING AREA, END.

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

13.1. **Obtain WimLib**

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

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

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

Copy the code or view the source file: Install.WinRE.Extract (Local, Github)

```
$Arguments = @(
  "extract",
  "D:\OS11\sources\install.wim", "1",
  "\Windows\System32\Recovery\Winre.wim",
  "--dest-dir=""D:\OS11_Custom\Install\Install\Update\Winlib"""
)

New-Item -Path "D:\OS11_Custom\Install\Install\Update\Winlib" -ItemType Directory -ea SilentlyContinue
Start-Process -FilePath "d:\wimlib\wimlib-imagex.exe" -ArgumentList $Arguments -wait -nonewwindow
```

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

Copy the code or view the source file: Install.WinRE.Replace.wim.ps1 (Local, Github)

```
Get-WindowsImage -ImagePath "D:\OS11\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:\OS11\sources\install.wim", $_.ImageIndex,
    "--command=""add 'D:\OS11_Custom\Install\Install\Update\Winlib\WinRE.wim'
'\Windows\System32\Recovery\WinRe.wim'"""
  )

  Start-Process -FilePath "d:\wimlib\wimlib-imagex.exe" -ArgumentList $Arguments -wait -nonewwindow
  Write-Host "  Finish`n" -ForegroundColor Green
}
```

14. **Rebuilding Install.wim reduces file size**

Copy the code or view the source file: Install.Rebuild.wim.ps1 (Local, Github)

```
$InstallWim = "D:\OS11\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 deployment image: boot.wim**

1.   **View Boot.wim details**

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

```
$ViewFile = "D:\OS11\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:\OS11_Custom\Boot\Boot\Mount" -ItemType directory -ea SilentlyContinue
```

3.   **Start mounting Boot.wim**

Default index number: 2

```
Mount-WindowsImage -ImagePath "D:\OS11\sources\boot.wim" -Index "2"  -Path "D:\OS11_Custom\Boot\Boot\Mount"
```

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.

4.1.   **Language pack: add**

Copy the code or view the source file: Boot.Instl.lang.ps1 (Local, Github)

```
$Mount = "D:\OS11_Custom\Boot\Boot\Mount"
$Sources = "D:\OS11_Custom\Boot\Boot\Language\Add\zh-CN"

$Initl_install_Language_Component = @()
Get-WindowsPackage -Path $Mount | ForEach-Object {
    $Initl_install_Language_Component += $_.PackageName
}
Add-WindowsPackage -Path $Mount -PackagePath "$($Sources)\WinPE-FontSupport-zh-CN.cab"

$Language_List = @(
    @{ Match = "*WinPE*Setup*Client*Package*"; File = "WINPE-SETUP-CLIENT_zh-CN.CAB"; }
    @{ Match = "*WinPE*Setup*Package*"; File = "WinPE-Setup_zh-CN.cab"; }
    @{ Match = "*WinPE-LanguagePack-Package*"; File = "lp.cab"; }
    @{ Match = "*SecureStartup*"; File = "winpe-securestartup_zh-CN.cab"; }
```

```
        @{ Match = "*ATBroker*"; File = "winpe-atbroker_zh-CN.cab"; }
        @{ Match = "*AudioCore*"; File = "winpe-audiocore_zh-CN.cab"; }
        @{ Match = "*AudioDrivers*"; File = "winpe-audiodrivers_zh-CN.cab"; }
        @{ Match = "*EnhancedStorage*"; File = "winpe-enhancedstorage_zh-CN.cab"; }
        @{ Match = "*Narrator*"; File = "winpe-narrator_zh-CN.cab"; }
        @{ Match = "*scripting*"; File = "winpe-scripting_zh-CN.cab"; }
        @{ Match = "*Speech-TTS*"; File = "winpe-speech-tts_zh-CN.cab"; }
        @{ Match = "*srh*"; File = "winpe-srh_zh-CN.cab"; }
        @{ Match = "*srt*"; File = "winpe-srt_zh-CN.cab"; }
        @{ Match = "*wds-tools*"; File = "winpe-wds-tools_zh-CN.cab"; }
        @{ Match = "*-WMI-Package*"; File = "winpe-wmi_zh-CN.cab"; }
)

ForEach ($Rule in $Language_List) {
    Write-host "`n  Rule name: $($Rule.Match)" -ForegroundColor Yellow; Write-host "   $('-' * 80)"
    ForEach ($Component in $Initl_install_Language_Component) {
        if ($Component -like "*$($Rule.Match)*") {
            Write-host "  Component name: " -NoNewline
            Write-host $Component -ForegroundColor Green
            Write-host "  Language pack file: " -NoNewline
            Write-host "$($Sources)\$($Rule.File)" -ForegroundColor Green
            Write-Host "  Installing ".PadRight(22) -NoNewline
            try {
                Add-WindowsPackage -Path $Mount -PackagePath "$($Sources)\$($Rule.File)" | Out-Null
                Write-host "Finish" -ForegroundColor Green
            } catch {
                Write-host "Failed" -ForegroundColor Red
            }
            break
        }
    }
}
```

4.2.   Offline image language: change

4.2.1.   Change default language, regional settings, and other international settings

Language Tag: zh-CN

Dism /Image:"D:\OS11_Custom\Boot\Boot\Mount" /Set-AllIntl:zh-CN

4.2.2.   View available language settings

Dism /Image:"D:\OS11_Custom\Boot\Boot\Mount" /Get-Intl

4.3.   Language packs: Removed, optional

- 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 language tag: en-US, you need to extract the language pack in advance

Copy the code or view the source files: Boot.Del.Specified.lang.Tag.ps1 (Local, Github)

$Lang = "en-US"

```powershell
$Mount = "D:\OS11_Custom\Boot\Boot\Mount"
$Sources = "D:\OS11_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
    }
  }
}
```

4.4. Components: All packages installed in the image

4.4.1. View

```
Get-WindowsPackage -Path "D:\OS11_Custom\Boot\Boot\Mount" | Out-GridView
```

4.4.2. Export to csv

```
$SaveTo = "D:\OS11_Custom\Boot\Boot\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"
Get-WindowsPackage -Path "D:\OS11_Custom\Boot\Boot\Mount" | Export-CSV -NoType -Path $SaveTo
Write-host $SaveTo -ForegroundColor Green
```

4.5. Language: Repair

4.5.1. Extract

Open: D:\OS11_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:

4.5.1.1. arunres.dll.mui

4.5.1.2. spwizres.dll.mui

4.5.1.3. w32uires.dll.mui

4.5.2. Copy

Copy the extracted files to: D:\OS11_Custom\Boot\Boot\Mount\sources\zh-CN

4.6. Language packs: sync to ISO installer

Copy-Item -Path "D:\OS11_Custom\Boot\Boot\Mount\sources\zh-CN" -Destination "D:\OS11\sources\zh-CN" -Recurse -Force

4.7. 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.

4.7.1. Regenerate the mounted directory 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:\OS11_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS11_Custom\Boot\Boot\Mount"

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

Regenerated Lang.ini file location: D:\en-us_windows_server_2022_x64_dvd_620d7eac\Sources\lang.ini

Dism /image:"D:\OS11_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS11"

5. Cumulative updates, optional

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

5.1. Add

$KBPath = "D:\OS11_Custom\Boot\Boot\Update\KB_Boot.cab"
Add-WindowsPackage -Path "D:\OS11_Custom\Boot\Boot\Mount" -PackagePath $KBPath

5.2. Delete

$KBPath = "D:\OS11_Custom\Boot\Boot\Update\KB_Boot.cab"
Remove-WindowsPackage -Path "D:\OS11_Custom\Boot\Boot\Mount" -PackagePath $KBPath

5.3. 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:\OS11_Custom\Boot\Boot\Mount" /cleanup-image /StartComponentCleanup /ResetBase

5.3.1. Clean components after curing and updating, optional

$Mount = "D:\OS11_Custom\Boot\Boot\Mount"
Get-WindowsPackage -Path $Mount -ErrorAction SilentlyContinue | ForEach-Object {

```
            if ($_.PackageState -eq "Superseded") {
              Write-Host "  $($_.PackageName)" -ForegroundColor Green
              Remove-WindowsPackage -Path $Mount -PackageName $_.PackageName | Out-Null
            }
          }
```

6. **Drive, optional**

7. **Other**

   7.1. **Bypass TPM**

   Copy the code or view the source file: Boot.Bypass.TPM.ps1 (Local, Github)

```
$RegSystem = "D:\OS11_Custom\Boot\Boot\Mount\Windows\System32\Config\SYSTEM"
$RandomGuid = [guid]::NewGuid()
Write-Host "  HKLM:\$($RandomGuid)"
New-PSDrive -PSProvider Registry -Name OtherTasksTPM -Root HKLM -ErrorAction SilentlyContinue | Out-Null
Start-Process reg -ArgumentList "Load ""HKLM\$($RandomGuid)"" ""$($RegSystem)""" -Wait -WindowStyle Hidden -ErrorAction SilentlyContinue
New-Item "HKLM:\$($RandomGuid)\Setup\LabConfig" -force -ea SilentlyContinue | Out-Null
New-ItemProperty -LiteralPath "HKLM:\$($RandomGuid)\Setup\LabConfig" -Name "BypassCPUCheck" -Value 1 -PropertyType DWord -Force -ea SilentlyContinue | Out-Null
New-ItemProperty -LiteralPath "HKLM:\$($RandomGuid)\Setup\LabConfig" -Name "BypassStorageCheck" -Value 1 -PropertyType DWord -Force -ea SilentlyContinue | Out-Null
New-ItemProperty -LiteralPath "HKLM:\$($RandomGuid)\Setup\LabConfig" -Name "BypassRAMCheck" -Value 1 -PropertyType DWord -Force -ea SilentlyContinue | Out-Null
New-ItemProperty -LiteralPath "HKLM:\$($RandomGuid)\Setup\LabConfig" -Name "BypassTPMCheck" -Value 1 -PropertyType DWord -Force -ea SilentlyContinue | Out-Null
New-ItemProperty -LiteralPath "HKLM:\$($RandomGuid)\Setup\LabConfig" -Name "BypassSecureBootCheck" -Value 1 -PropertyType DWord -Force -ea SilentlyContinue | Out-Null
[gc]::collect()
Start-Process reg -ArgumentList "unload ""HKLM\$($RandomGuid)""" -Wait -WindowStyle Hidden -ErrorAction SilentlyContinue
Remove-PSDrive -Name OtherTasksTPM
```

8. **Save image**

```
Save-WindowsImage -Path "D:\OS11_Custom\Boot\Boot\Mount"
```

9. **Unmount image**

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

```
Dismount-WindowsImage -Path "D:\OS11_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: 1.1.0.4, 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", 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\Unattend\Mul.Unattend.xml to D:\OS11\Autounattend.xml

            Copy-Item "D:\Multilingual\Unattend\Mul.Unattend.xml" -Destination "D:\OS11\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\Unattend\Mul.Unattend.xml to D:\OS11\Sources\Unattend.xml

            Copy-Item "D:\Multilingual\Unattend\Mul.Unattend.xml" -Destination "D:\OS11\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:\OS11\sources\`$OEM$\`$$\Panther" -ItemType Directory

                1.1.1.3.2. **Copy**

                Copy D:\Multilingual\Unattend\Mul.Unattend.xml to D:\OS11\Sources\OEM\Panther\Unattend.xml

                Copy-Item "D:\Multilingual\Unattend\Mul.Unattend.xml" -Destination "D:\OS11\sources\`$OEM$\`$$\Panther\Unattend.xml" -Force

        1.1.2. **Deployment engine: add**

            Add "Automatically add installed languages for Windows systems" to D:\OS11\sources\$OEM$\$1\Yi\Engine in the directory.

            1.1.2.1. **Deployment Engine: Copy**

                Copy D:\Multilingual\Engine to D:\OS11\Sources\$OEM$\$1\Yi\Engine

                Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS11\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:\OS11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -
ErrorAction SilentlyContinue | Out-Null

    Out-File -FilePath "D:\OS11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow\$($item)" -Encoding utf8 -
ErrorAction SilentlyContinue
}
```

1.2. Add to mounted

Through "Customized deployment image: Install.wim", execute "Start mounting Install.wim" and mount to:
D:\OS11_Custom\Install\Install\Mount

1.2.1. Unattended

Copy D:\Multilingual\Unattend\Mul.Unattend.xml to D:\OS11_Custom\Install\Install\Mount\Panther\Unattend.xml

```
Copy-Item "D:\Multilingual\Unattend\Mul.Unattend.xml" -Destination
"D:\OS11_Custom\Install\Install\Mount\Panther" -Force
```

1.2.2. Deployment engine: add

Add "Automatically add languages installed on Windows systems" to the
D:\OS11_Custom\Install\Install\Mount\Yi\Engine directory.

1.2.2.1. Deployment Engine: Copy

Copy D:\Multilingual\Engine to D:\OS11_Custom\Install\Install\Mount\Yi\Engine

```
Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS11_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:\OS11\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -
ErrorAction SilentlyContinue | Out-Null
   Out-File -FilePath "D:\OS11\sources\`$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:\OS11\Autounattend.xml
- D:\OS11\Sources\Unattend.xml
- D:\OS11\sources\$OEM$\$$\Panther\unattend.xml
- D:\OS11_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 language tag, for example, specify a language tag: zh-CN

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

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. 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:\OS11.iso

   Copy the code or view the source file: ISO.ps1 (Local, Github)

```powershell
$ISO = "D:\OS11"
$Volume = "OS11"
$SaveTo = "D:\OS11.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
```

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:\OS11.iso file to Quick_11_iso_esd_wim_TPM_toggle.bat, and "add" or "delete" the TPM installation check function in reverse order.

TWO. COMMON PROBLEM

II Clean all mounts to

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

```
Dismount-WindowsImage -Path "D:\OS11_Custom\Install\Install\Mount" -Discard
```

```
Dismount-WindowsImage -Path "D:\OS11_Custom\Install\WinRE\Mount" -Discard
```

```
Dismount-WindowsImage -Path "D:\OS11_Custom\Boot\Boot\Mount" -Discard
```

III   Fix the problem of abnormal mounting

1.   View mounted

Get-WindowsImage -Mounted

2.   Delete the DISM mount record saved in the registry

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

3.   Delete all resources associated with the corrupted mounted image

Clear-WindowsCorruptMountPoint

Dism /cleanup-wim

IV   Clean directories

Remove-Item "D:\OS11_Custom" -Force -Recurse

Remove-Item "D:\OS11\Sources\`OEM" -Force -Recurse

Remove-Item -Path "D:\OS11\Autounattend.xml" -Force

Remove-Item -Path "D:\OS11\Sources\Unattend.xml" -Force

THREE.   KNOWN ISSUES

None yet

FOUR. COMPONENTS: ALL PACKAGES INSTALLED IN THE IMAGE, COMPARISON

Changes after installing new language pack, component list:

Sn	Type	Component: Original	Component: Changes after installing "Simplified Chinese - China" language pack
1	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
2	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
3	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
4	LanguagePack		Microsoft-Windows-Client-LanguagePack-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.382
5	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
6	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
7	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
8	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
9	FeaturePack	Microsoft-Windows-FodMetadata-Package~31bf3856ad364e35~amd64~~10.0.22621.1	Microsoft-Windows-FodMetadata-Package~31bf3856ad364e35~amd64~~10.0.22621.1
10	Foundation	Microsoft-Windows-Foundation-Package~31bf3856ad364e35~amd64~~10.0.22621.1	Microsoft-Windows-Foundation-Package~31bf3856ad364e35~amd64~~10.0.22621.1
11	OnDemandPack	Microsoft-Windows-Hello-Face-Package~31bf3856ad364e35~amd64~~10.0.22621.317	Microsoft-Windows-Hello-Face-Package~31bf3856ad364e35~amd64~~10.0.22621.317
12	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
13	OnDemandPack		Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~amd64~zh-CN~11.0.22621.1
14	OnDemandPack	Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~amd64~~11.0.22621.1	Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~amd64~~11.0.22621.1
15	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
16	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
17	OnDemandPack		Microsoft-Windows-LanguageFeatures-Basic-zh-cn-Package~31bf3856ad364e35~amd64~~10.0.22621.1
18	OnDemandPack		Microsoft-Windows-LanguageFeatures-Fonts-Hans-Package~31bf3856ad364e35~amd64~~10.0.22621.1
19	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
20	OnDemandPack		Microsoft-Windows-LanguageFeatures-Handwriting-zh-cn-Package~31bf3856ad364e35~amd64~~10.0.22621.317
21	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
22	OnDemandPack		Microsoft-Windows-LanguageFeatures-OCR-zh-cn-Package~31bf3856ad364e35~amd64~~10.0.22621.1
23	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
24	OnDemandPack		Microsoft-Windows-LanguageFeatures-Speech-zh-cn-Package~31bf3856ad364e35~amd64~~10.0.22621.1
25	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
26	OnDemandPack		Microsoft-Windows-LanguageFeatures-TextToSpeech-zh-cn-Package~31bf3856ad364e35~amd64~~10.0.22621.1
27	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

SN	TYPE	COMPONENT: ORIGINAL	COMPONENT: CHANGES AFTER INSTALLING "SIMPLIFIED CHINESE - CHINA" LANGUAGE PACK
28	OnDemandPack		Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.317
29	OnDemandPack	Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~amd64~~10.0.22621.382	Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~amd64~~10.0.22621.382
30	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
31	OnDemandPack		Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1
32	OnDemandPack	Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~wow64~~10.0.22621.1	Microsoft-Windows-MediaPlayer-Package~31bf3856ad364e35~wow64~~10.0.22621.1
33	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
34	OnDemandPack		Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1
35	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
36	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
37	OnDemandPack		Microsoft-Windows-Notepad-System-FoD-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1
38	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
39	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
40	OnDemandPack		Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1
41	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
42	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
43	OnDemandPack		Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1
44	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
45	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
46	OnDemandPack		Microsoft-Windows-Printing-PMCPPC-FoD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1
47	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
48	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
49	OnDemandPack		Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1
50	OnDemandPack	Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~~10.0.22621.1	Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~amd64~~10.0.22621.1
51	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
52	OnDemandPack		Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1
53	OnDemandPack	Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~~10.0.22621.1	Microsoft-Windows-StepsRecorder-Package~31bf3856ad364e35~wow64~~10.0.22621.1
54	OnDemandPack	Microsoft-Windows-TabletPCMath-Package~31bf3856ad364e35~amd64~~10.0.22621.1	Microsoft-Windows-TabletPCMath-Package~31bf3856ad364e35~amd64~~10.0.22621.1
55	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
56	OnDemandPack	Microsoft-Windows-Wifi-Client-Broadcom-Bcmpciedhd63-FOD-	Microsoft-Windows-Wifi-Client-Broadcom-Bcmpciedhd63-FOD-

Sn	Type	Component: Original	Component: Changes after installing "Simplified Chinese - China" language pack
		Package~31bf3856ad364e35~amd64~~10.0.22621.1	Package~31bf3856ad364e35~amd64~~10.0.22621.1
57	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
58	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
59	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
60	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
61	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
62	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
63	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
64	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
65	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
66	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
67	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
68	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
69	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
70	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
71	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
72	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
73	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
74	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
75	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
76	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
77	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
78	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
79	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
80	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
81	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
82	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
83	OnDemandPack		Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1
84	OnDemandPack	Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.382	Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.382

Sn	Type	Component: Original	Component: Changes after installing "Simplified Chinese - China" language pack
85	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
86	OnDemandPack		Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1
87	OnDemandPack	Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~~10.0.22621.1	Microsoft-Windows-WMIC-FoD-Package~31bf3856ad364e35~wow64~~10.0.22621.1
88	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
89	OnDemandPack		Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~zh-CN~10.0.22621.1
90	OnDemandPack	Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.317	Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~amd64~~10.0.22621.317
91	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
92	OnDemandPack		Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~zh-CN~10.0.22621.1
93	OnDemandPack	Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~~10.0.22621.1	Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~~10.0.22621.1
94	OnDemandPack	OpenSSH-Client-Package~31bf3856ad364e35~amd64~~10.0.22621.1	OpenSSH-Client-Package~31bf3856ad364e35~amd64~~10.0.22621.1
95	Update	Package_for_DotNetRollup_481~31bf3856ad364e35~amd64~~10.0.9065.6	Package_for_DotNetRollup_481~31bf3856ad364e35~amd64~~10.0.9065.6
96	SecurityUpdate	Package_for_RollupFix~31bf3856ad364e35~amd64~~22621.382.1.13	Package_for_RollupFix~31bf3856ad364e35~amd64~~22621.382.1.13
97	SecurityUpdate	Package_for_ServicingStack_378~31bf3856ad364e35~amd64~~22621.378.1.0	Package_for_ServicingStack_378~31bf3856ad364e35~amd64~~22621.378.1.0