



MICROSOFT WINDOWS 10

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

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

Summary

Chapter 1 Deployment image

Table of contents

Chapter 1	Deployment image	Page 4
A.	Prerequisites	Page 4
II	ISO tools	Page 4
III	Requirements	Page 4
1.	System installation package	Page 4
2.	Language pack	Page 5
2.1.	Learn	Page 5
2.2.	Language pack: Download	Page 7
3.	InBox Apps	Page 7
IV	Command Line	Page 7
B.	Language Pack: Extract	Page 8
II	Language pack: Ready	Page 8
III	Language pack: Scheme	Page 8
IV	Execute the extract command	Page 8
C.	Customize the deployment image	Page 14
II	Custom deployment image Install.wim	Page 14
1.	View Install.wim details	Page 14
2.	Specify the path to mount Install.wim	Page 14
3.	Start mounting Install.wim	Page 14
3.1.	Custom deployment image WinRE.wim	Page 14
3.1.1.	View WinRE.wim details	Page 15
3.1.2.	Specify the path to mount WinRE.wim	Page 15
3.1.3.	Start mounting WinRE.wim	Page 15
3.1.4.	Language pack	Page 15
3.1.4.1.	Language pack: add	Page 15
3.1.4.2.	Components: All packages installed in the image	Page 17
3.1.5.	Save image	Page 17
3.1.6.	Unmount image	Page 17
3.1.7.	After rebuilding WinRE.wim, the file size can be reduced	Page 17

3.1.8.	Backup WinRE.wim	Page 18
3.1.9.	Replace WinRE.wim within the Install.wim image	Page 18
4.	Language pack	Page 18
4.1.	Language pack: add	Page 19
4.2.	Components: All packages installed in the image	Page 27
5.	InBox Apps	Page 28
5.1.	InBox Apps: Installed	Page 28
5.2.	Remove all installed pre-applications	Page 28
5.3.	Region tag: adding method	Page 28
5.4.	InBox Apps: Install	Page 30
5.5.	InBox Apps: optimization	Page 36
6.	Cumulative updates	Page 37
6.1.	Feature enablement package	Page 37
6.2.	Initial version	Page 37
6.3.	Other Version	Page 38
6.4.	Solidify Updated	Page 38
7.	Add deployment engine	Page 38
8.	Health	Page 38
9.	Replace WinRE.wim	Page 39
10.	Save image	Page 39
11.	Unmount image	Page 39
12.	Rebuilding Install.wim reduces file size	Page 39
13.	How to batch replace WinRE.wim in all index numbers in Install.wim	Page 39
13.1.	Obtain WimLib	Page 40
13.2.	How to extract and update WinRE.wim in Install.wim	Page 40
III	Custom deployment image boot.wim	Page 41
1.	View Boot.wim details	Page 41
2.	Specify the path to mount Boot.wim	Page 41
3.	Start mounting Boot.wim	Page 41
4.	Language pack	Page 41

4.1.	Language pack: Add	Page 41
4.2.	Components: All packages installed in the image	Page 43
4.3.	Language: Repair	Page 43
4.4.	Language packs: sync to ISO installer	Page 43
4.5.	Regenerate Lang.ini	Page 43
4.5.1.	Regenerate the mounted directory lang.ini	Page 43
4.5.2.	After regenerating lang.ini, sync to the installer	Page 44
5.	Save image	Page 44
6.	Unmount image	Page 44
IV	Deployment engine	Page 44
1.	Add method	Page 44
2.	Deployment Engine: Advanced	Page 47
D.	ISO	Page 49
II	Generate ISO	Page 49

Chapter 1 Deployment image

A. Prerequisites

II ISO tools

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

III Requirements

1. System installation package

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

1.1. illustrate

- 1.1.1. Please remake the image when each version is updated, for example, when crossing from 21H1 to 22H2, avoid other compatibility problems, and do not make the image based on the old image;
- 1.1.2. The regulation has been clearly communicated to packagers in various forms by some OEMs, and direct upgrades from iterative versions are not allowed;
- 1.1.3. Please use "Initial Version" and "Developer Edition" for production. There was a brief appearance in the official Microsoft documentation that the initial version must be used in production, but this sentence was later deleted in the official documentation;
- 1.1.4. After installing the language pack, you must re-add the cumulative update (the same version number), and if you do not add the cumulative update, problems such as "garbled characters" and "interface flashback" will occur.
- 1.1.5. Evolutionary process: Windows 10 22H2, Build 19045.2006 + KB5027215 = OS Build 19045.3086

1.2. Prepare to download the initial or developer version

- 1.2.1. [en-us_windows_10_business_editions_version_22h2_x64_dvd_8cf17b79.iso](#)
- 1.2.2. [en-us_windows_10_consumer_editions_version_22h2_x64_dvd_8da72ab3.iso](#)

1.3. After the sample download [en-us_windows_10_business_editions_version_22h2_x64_dvd_8cf17b79.iso](#), Unzip to: [D:\en-us_windows_10_business_editions_version_22h2_x64_dvd_8cf17b79](#)

1.4. After decompression is complete, change the directory [en-us_windows_10_business_editions_version_22h2_x64_dvd_8cf17b79](#) change to [D:\OS_10](#)

1.5. All scripts and all paths are set to [D:\OS_10](#) as the image source.

1.6. Installation configuration

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

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

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

[@"](#)

```
[Channel]

volume

[VL]

1

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

2. Language pack

2.1. Learn

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

- 2.1.1. [Languages overview](#)
- 2.1.2. [Add languages to a Windows 10 image](#)
- 2.1.3. [Language and region Features on Demand \(FOD\)](#)

2.1.3.1. Fonts

- When adding a language pack, when the corresponding region is triggered, the required font functions need to be added, download "[List of all available language FODs](#)" learn more.
- In "Language Pack: Extract", the automatic recognition function has been added, and you can understand the functions: [Function Match_Required_Fonts](#)

2.1.3.2. Regional association

What are regional connections?

- When the image language is only in English, after adding the [zh-HK](#) language pack, the image language will not be added. You should install [zh-TW](#) first, and then install [zh-HK](#) to obtain the corresponding association.

Known regional associations:

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

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

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

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

2.1.3.2.3. Region: [da-dk](#), Optional associated areas: [en-US](#), associated installation package:

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

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

2.1.3.2.10. Region: zh-TW , Optional associated areas: zh-HK, associated installation package:

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

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

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

2.1.3.3. Other region-specific requirements

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

2.1.3.3.1. Region: zh-TW

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

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

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

THERE IS CONTROVERSY:

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

2.2. Language pack: Download

https://software-download.microsoft.com/download/pr/19041.1.191206-1406.vb_release_CLIENTLANGPACKDVD_OEM_MULTI.iso

2.3. Feature pack: Download

X64: https://software-download.microsoft.com/download/pr/19041.1.191206-1406.vb_release_amd64fre_FOD-PACKAGES_OEM_PT1_amd64fre_MULTI.iso

3. InBox Apps

https://software-static.download.prss.microsoft.com/dbazure/888969d5-f34g-4e03-ac9d-1f9786c66749/19041.3031.230508-1728.vb_release_svc_prod3_amd64fre_InboxApps.iso

IV Command line

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

downloading;

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

`Set-ExecutionPolicy -ExecutionPolicy Bypass -Force`

3. In this article, PS command line, green part, please copy it, paste it into the "Terminal" dialog box, press Enter and start running;
4. When there is `.ps1`, right-click the file and select Run with PowerShell, or copy the path and paste it into Terminal to run, the path with a colon, add the `&` character in the command line, example: `& "D:\Yi.Solutions\Encapsulation\SIP.ps1"`

B. Language Pack: Extract

II Language pack: Ready

1. Language pack: Mount

Mount `19041.1.191206-1406.vb_release_CLIENTLANGPACKDVD_OEM_MULTI.iso` or unzipped to any location;

2. Feature pack: Mount

Please select the schema version correctly, extract the wrong language pack, and the installation will report an error.

X64: `19041.1.191206-1406.vb_release_amd64fre_FOD-PACKAGES_OEM_PT1_amd64fre_MULTI.iso`

III Language pack: Scheme

1. Add

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

2. Delete

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

3. Architecture: `x64`

IV Execute the extract command

- `Auto` = automatically search all local disks, default;
- Customize the path, for example, specify the E drive: `$ISO = "E:\\"`
- `Extract.ps1`
 - `\Expand\Extract.ps1`
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Extract.ps1
- Copy the code

`$ISO = "Auto"`

`$SaveTo = "D:\OS_10_Custom"`

`$Extract_Language_Pack = @(`

```
@{ Tag = "zh-CN"; Arch = "AMD64"; Act = "Add"; Scope = @"(\"Install\\Install\"; \"Install\\WinRE\"; \"Boot\\Boot\") }

@{ Tag = "en-US"; Arch = "AMD64"; Act = "Del"; Scope = @( \"Install\\Install\"; \"Install\\WinRE\"; \"Boot\\Boot\" ) }

)
```

Function Extract_Language

```
{

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

    Function Match_Required_Fonts

    {

        param($Lang)

        $Fonts = @(

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

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

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

    )

    ForEach ($item in $Fonts) {

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

            return $item.Name

        }

    }

    return "Not_matched"

}

Function Match_Other_Region_Specific_Requirements

{

    param($Lang)

    $RegionSpecific = @(

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

    )

    ForEach ($item in $RegionSpecific) {

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

            return $item.Name

        }

    }

    return "Skip_specific_packages"

}

Function Extract_Process

{

    param($Package, $Name, $NewSaveTo)

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

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

    if ($ISO -eq "Auto") {

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

            ForEach ($item in $Package) {

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

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

```

```

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

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

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

    }

}

}

} else {

    ForEach ($item in $Package) {

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

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

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

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

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

        } else {

            Write-host "    Not found"

        }

    }

}

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

ForEach ($item in $Package) {

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

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

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

    } else {

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

    }

}

}

$AdvLanguage = @(

    @{

        Path = "Install\Install"

        Rule = @(

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

            "{ARCHC}\langpacks\Microsoft-Windows-Client-Language-Pack_x64_{Lang}.cab"

```

```
"Microsoft-Windows-LanguageFeatures-Basic-{Lang}-Package~31bf3856ad364e35~AMD64~~.cab"

"Microsoft-Windows-LanguageFeatures-Handwriting-{Lang}-Package~31bf3856ad364e35~AMD64~~.cab"

"Microsoft-Windows-LanguageFeatures-OCR-{Lang}-Package~31bf3856ad364e35~AMD64~~.cab"

"Microsoft-Windows-LanguageFeatures-Speech-{Lang}-Package~31bf3856ad364e35~AMD64~~.cab"

"Microsoft-Windows-LanguageFeatures-TextToSpeech-{Lang}-Package~31bf3856ad364e35~AMD64~~.cab"

"Microsoft-Windows-InternetExplorer-Optional-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"Microsoft-Windows-MSPaint-FoD-Package~31bf3856ad364e35~wow64~{Lang}~.cab"

"Microsoft-Windows-MSPaint-FoD-Package~31bf3856ad364e35~{ARCH}~{Lang}~.cab"

"Microsoft-Windows-Notepad-FoD-Package~31bf3856ad364e35~{ARCH}~{Lang}~.cab"

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

"Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

"Microsoft-Windows-PowerShell-ISE-FOD-Package~31bf3856ad364e35~wow64~{Lang}~.cab"

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

"Microsoft-Windows-Printing-WFS-FoD-Package~31bf3856ad364e35~AMD64~{Lang}~.cab"

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

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

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

"Microsoft-Windows-WordPad-FoD-Package~31bf3856ad364e35~wow64~{Lang}~.cab"

"Microsoft-Windows-InternationalFeatures-{Specific}-Package~31bf3856ad364e35~amd64~~.cab"

)

}

@{

    Path = "Install\WinRE"

    Rule = @(

        "Windows Preinstallation Environment\x64\WinPE_OCs\WinPE-FontSupport-{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\lp.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-securestartup_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-atbroker_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-audiocore_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-audiodrivers_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-enhancedstorage_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-narrator_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-scripting_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OCs\{Lang}\winpe-speech-tts_{Lang}.cab"
```

```
"Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-srh_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-srt_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-wds-tools_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-wmi_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-appxpackaging_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-storagewmi_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-wifi_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-rejuv_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-opcservices_{Lang}.cab"

"Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-hta_{Lang}.cab"

)

}

@{

    Path = "Boot\Boot"

    Rule = @(

        "Windows Preinstallation Environment\x64\WinPE_OC\WinPE-FontSupport-{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\lp.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\WinPE-Setup_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\WINPE-SETUP-CLIENT_{Lang}.CAB"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-securestartup_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-atbroker_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-audiocore_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-audiodrivers_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-enhancedstorage_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-narrator_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-scripting_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-speech-tts_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-srh_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-srt_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-wds-tools_{Lang}.cab"

        "Windows Preinstallation Environment\x64\WinPE_OC\s\{Lang}\winpe-wmi_{Lang}.cab"

    )

}

)
```

```

$NewFonts = Match_Required_Fonts -Lang $NewLang

$SpecificPackage = Match_Other_Region_Specific_Requirements -Lang $NewLang

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

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).Replace("{ARCH}", $NewArch).Replace("{ARCHC}", $NewArchC)

            }

            Extract_Process -NewSaveTo $itemList.Path -Package $Language -Name $item

        }

    }

}

}

Foreach ($item in $Extract_language_Pack) { Extract_Language -Act $item.Act -NewLang $item.Tag -NewArch $item.Arch -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:\OS_10\Sources\Install.wim"
```

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

LOOP OPERATING AREA, START,

2. Specify the path to mount Install.wim

```
New-Item -Path "D:\OS_10_Custom\Install\Install\Mount" -ItemType directory -ea SilentlyContinue
```

3. Start mounting Install.wim

Default index number: 1

```
Mount-WindowsImage -ImagePath "D:\OS_10\sources\install.wim" -Index "1" -Path "D:\OS_10_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:\OS_10_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"
```

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

3.1.2. Specify the path to mount WinRE.wim

```
New-Item -Path "D:\OS_10_Custom\Install\WinRE\Mount" -ItemType directory -ea SilentlyContinue
```

3.1.3. Start mounting WinRE.wim

Default index number: 1

```
Mount-WindowsImage -ImagePath "D:\OS_10_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim" -
Index "1" -Path "D:\OS_10_Custom\Install\WinRE\Mount"
```

3.1.4. Language pack: WinRE

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

3.1.4.1. Language pack: add

- WinRE.Instl.lang.ps1
 - [\Expand\Install\WinRE\WinRE.Instl.lang.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Install/WinRE/WinRE.Instl.lang.ps1

- Copy the code

```
$Mount = "D:\OS_10_Custom\Install\WinRE\Mount"
```

```
$Sources = "D:\OS_10_Custom\Install\WinRE\Language\Add\zh-CN"
```

```
$Initl_install_Language_Component = @()
```

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

```
    $Initl_install_Language_Component += $_.PackageName
```



```

}

Add-WindowsPackage -Path $Mount -PackagePath "$($Sources)\WinPE-FontSupport-zh-CN.cab"

$Language = @(

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

    @{ Match = "*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"; }

)

ForEach ($Rule in $Language) {

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

    ForEach ($Component in $Initl_install_Language_Component) {

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

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

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

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

            try{

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

                Write-host "Finish" -ForegroundColor Green

            } catch {

```

```

Write-host "Failed" -ForegroundColor Red

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

}

break

}

}

}

```

3.1.4.2. Components: All packages installed in the image

3.1.4.2.1. View

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

3.1.4.2.2. Export to csv

```
$SaveTo = "D:\OS_11_Custom\Install\WinRE\Report.$(Get-Date -Format
"yyyyMMddHHmmss").csv"
```

```
Get-WindowsPackage -Path "D:\OS_11_Custom\Install\WinRE\Mount" | Export-CSV -
NoType -Path $SaveTo
```

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

3.1.5. Save image

```
Save-WindowsImage -Path "D:\OS_10_Custom\Install\WinRE\Mount"
```

3.1.6. Unmount image

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

```
Dismount-WindowsImage -Path "D:\OS_10_Custom\Install\WinRE\Mount" -Discard
```

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

- WinRE.Rebuild.ps1
 - [\Install\WinRE\WinRE.Rebuild.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Install/WinRE/WinRE.Rebuild.ps1

- Copy the code

```
$FileName = "D:\OS_10_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"
```

```
Get-WindowsImage -ImagePath $Filename -ErrorAction SilentlyContinue | ForEach-Object {
```

```
Write-Host " Image name: " -NoNewline
```

```
Write-Host $_.ImageName -ForegroundColor Yellow
```

```

Write-Host " The index number: " -NoNewline

Write-Host $_.ImageIndex -ForegroundColor Yellow

Write-Host "`n Rebuild".PadRight(28) -NoNewline

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

Write-Host "Finish`n" -ForegroundColor Green

}

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

    Remove-Item -Path $Filename

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

    Write-Host "Finish" -ForegroundColor Green

} else {

    Write-host "Failed" -ForegroundColor Red

}

```

3.1.8. Backup WinRE.wim

- WinRE.Backup.ps1
 - [\Expand\Install\WinRE\WinRE.Backup.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Install/WinRE/WinRE.Backup.ps1

- Copy the code

```

$WimLibPath = "D:\OS_10_Custom\Install\Install\Update\Winlib"

$FileName = "D:\OS_10_Custom\Install\Install\Mount\Windows\System32\Recovery\WinRE.wim"

New-Item -Path $WimLibPath -ItemType Directory -ea SilentlyContinue

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

```

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

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

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

4. Language pack

- Automatically install language packs: Get "Component: All installed packages in the image" and match them. After matching the corresponding names, install the local corresponding language pack files, View the report "[Language installation package for Install.wim](#)".

- When adding languages, different schema versions must be corresponded, and if not, errors are reported during the addition process.

4.1. Language pack: add

- Install.Instl.lang.ps1
 - [\Expand\Install\Install.Instl.lang.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Install/Install.Instl.lang.ps1

- Copy the code

Function Language_Install

```
{
    param($Mount, $Sources, $Lang)

    $Initl_install_Language_Component = @()

    if (Test-Path $Mount -PathType Container) {

        Get-WindowsPackage -Path $Mount | ForEach-Object { $Initl_install_Language_Component += $_.PackageName }

    } else {

        Write-Host "Not mounted: $($Mount)"

        return

    }

    $Script:Init_Folder_All_File = @()

    if (Test-Path "$($Sources)\($Lang)" -PathType Container) {

        Get-ChildItem -Path $Sources -Recurse -Include "*.cab" -ErrorAction SilentlyContinue | ForEach-Object {

            $Script:Init_Folder_All_File += $_.FullName

        }

        Write-host "`n Available language pack installation files"

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

            ForEach ($item in $Script:Init_Folder_All_File) {

                Write-host " $($item)"

            }

        } else {

            Write-host "There are no language pack files locally"

            return

        }

    } else {

        Write-Host "Path does not exist: $($Sources)\($Lang)"

        return

    }
}
```

```

}

$Script:Init_Folder_All_File_Match_Done = @()

$Script:Init_Folder_All_File_Exclude = @()

$Script:Search_File_Order = @(

    @{

        Name = "Fonts"

        Description = "Fonts"

        Rule = @(

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

        )

    }

    @{

        Name = "Basic"

        Description = "Basic"

        Rule = @(

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

            @{ Match_Name = "*Client*Language*Pack*"; IsMatch = "Yes"; Capability = "Language.Basic~~~${$Lang}~0.0.1.0"; }

        )

    }

    @{

        Name = "OCR"

        Description = "Optical character recognition"

        Rule = @(

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

        )

    }

    @{

        Name = "Handwriting"

        Description = "Handwriting recognition"

        Rule = @(

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

        )

    }

    @{

```

```

        Name = "TextToSpeech"

        Description = "Text-to-speech"

        Rule = @(

            @{ Match_Name = "*LanguageFeatures-TextToSpeech*"; IsMatch = "Yes"; Capability =
"Language.TextToSpeech~~~$(Lang)~0.0.1.0"; }

        )

    }

    @{

        Name = "Speech"

        Description = "Speech recognition"

        Rule = @(

            @{ Match_Name = "*LanguageFeatures-Speech*"; IsMatch = "Yes"; Capability = "Language.Speech~~~$(Lang)~0.0.1.0"; }

        )

    }

    @{

        Name = "RegionSpecific"

        Description = "Other region-specific requirements"

        Rule = @(

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

        )

    }

    @{

        Name = "Retail"

        Description = "Retail demo experience"

        Rule = @(

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

        )

    }

    @{

        Name = "Features_On_Demand"

        Description = "Features on demand"

        Rule = @(

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

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

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

```

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    )

}

)

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

ForEach ($Wildcard in $Script:Init_Folder_All_File) {

    ForEach ($item in $Script:Search_File_Order) {

        ForEach ($NewRule in $item.Rule) {

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

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

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

            }

        }

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

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

        $new = $OSDefaultUser + $TempSave

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

            ForEach ($Component in $Initl_install_Language_Component) {

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

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

                }

            }

        }

    }

}

```

```

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

        $Script:Init_Folder_All_File_Match_Done += $WildCard

        break
    }

}

} else {

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

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

    $Script:Init_Folder_All_File_Match_Done += $WildCard

}

}

}

}

}

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

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

ForEach ($WildCard in $Script:Search_File_Order) {

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

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

    if ($OSDefaultUser.Count -gt 0) {

        ForEach ($item in $OSDefaultUser) {

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

        }

    } else {

        Write-host "  Not available" -ForegroundColor Red

    }

}

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

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

ForEach ($item in $Script:Init_Folder_All_File) {

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

        $Script:Init_Folder_All_File_Exclude += $item

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

    }

}

```



```

}

Write-host "`n  Install" -ForegroundColor Yellow

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

ForEach ($WildCard in $Script:Search_File_Order) {

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

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

    if ($OSDefaultUser.Count -gt 0) {

        ForEach ($item in $OSDefaultUser) {

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

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

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

                try {

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

                    Write-host "Finish`n" -ForegroundColor Green

                } catch {

                    Write-host "Failed" -ForegroundColor Red

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

                }

            } else {

                Write-host "Does not exist`n"

            }

        }

    } else {

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

    }

}

}

Language_Install -Mount "D:\OS_10_Custom\Install\Install\Mount" -Sources "D:\OS_10_Custom\Install\Install\Language\Add" -
Lang "zh-CN"

```

4.2. Offline image language: change

- Starting Windows 11, the [default System UI Language](#) set by DISM is left unaltered on all editions except for Home edition. For [all commercial editions](#) the language chosen during the Out-of-Box Experience (OOBE) is set as the [System Preferred UI language](#) and Windows will be displayed in this language and for Home edition the language chosen at OOBE will continue to be the default System UI Language.
- As of Windows 10, version 2004, if an .appx-based Language Experience Pack (LXP) backed language is passed as an argument then the language will be set as the System Preferred UI language and its parent language will be set as the Default

System UI language. In prior versions only .cab based language packs were supported.

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

Region: zh-CN

```
Dism /Image:"D:\OS_10_Custom\Install\Install\Mount" /Set-AllIntl:zh-CN
```

4.2.2. View available language settings

```
Dism /Image:"D:\OS_10_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 Region tag: en-US, you need to extract the language pack in advance

- Install.Del.Specified.lang.Tag.ps1

- \Expand\Install\Install.Del.Specified.lang.Tag.ps1

- https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Install/Install/Install.Del.Specified.lang.Tag.ps1

- Copy the code

```
$Lang = "en-US"
```

```
$Mount = "D:\OS_10_Custom\Install\Install\Mount"
```

```
$Sources = "D:\OS_10_Custom\Install\Install\Language\Del\en-US"
```

```
$Initl_install_Language_Component = @()
```

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

```
    $Initl_install_Language_Component += $_.PackageName
```

```
}
```

```
$Language = @(
```

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

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

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

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

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

        @{ Match = "**Client-LanguagePack-Package*"; File = "Microsoft-Windows-Client-Language-Pack_x64_$(Lang).cab"; }

    )

    ForEach ($Rule in $Language) {

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

        ForEach ($Component in $InitL_install_Language_Component) {

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

                Write-host "  Component name: " -NoNewline

                Write-host $Component -ForegroundColor Green

                Write-host "  Language pack file: " -NoNewline

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

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

                try {

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

```

```

        Write-host "Finish" -ForegroundColor Green

    } catch {

        Write-host "Failed" -ForegroundColor Red

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

    }

    break

}

}

}

$InitlClearLanguagePackage = @()

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

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

        $InitlClearLanguagePackage += $_.PackageName

    }

}

if ($InitlClearLanguagePackage.count -gt 0) {

    ForEach ($item in $InitlClearLanguagePackage) {

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

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

        try {

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

            Write-host "Finish" -ForegroundColor Green

        } catch {

            Write-host "Failed" -ForegroundColor Red

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

        }

    }

}

```

4.4. Components: All packages installed in the image

4.4.1. View

```
Get-WindowsPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Out-GridView
```

4.4.2. Export to csv

```
$SaveTo = "D:\OS_10_Custom\Install\Install\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"
```

```
Get-WindowsPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo
```

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

5. InBox Apps

5.1. InBox Apps: Installed

5.1.1. View

```
Get-AppXProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Out-GridView
```

5.1.2. Export to Csv

```
$SaveTo = "D:\OS_10_Custom\Install\Install\Report.${(Get-Date -Format "yyyyMMddHHmmss")}.csv"
```

```
Get-AppXProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo
```

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

5.2. Remove all installed pre-applications

- Install.InBox.Appx.Clear.all.ps1
 - [\Expand\Install\Install.InBox.Appx.Clear.all.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Install/Install.InBox.Appx.Clear.all.ps1

- Copy the code

```
Get-AppXProvisionedPackage -path "D:\OS_10_Custom\Install\Install\Mount" -ErrorAction SilentlyContinue | ForEach-Object {
```

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

```
    try {
```

```
        Remove-AppxProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" -PackageName $_.PackageName -ErrorAction SilentlyContinue | Out-Null
```

```
        Write-host "Finish" -ForegroundColor Green
```

```
    } catch {
```

```
        Write-host "Failed" -ForegroundColor Red
```

```
    }
```

```
}
```

5.3. Region tag: adding method

5.3.1. Execute "Language Pack: Add"

5.3.2. Install “Local Language Experience Packages (LXPs)”

Microsoft officially provides the Local Language Experience Package (LXPS) installation file for Windows 10. It will no longer be provided for Windows 11. Want to get:

- 5.3.2.1.

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

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

After downloading, save to: [D:\OS_10_Custom\Install\Install\InBox.Appx](#)

File format: [LanguageExperiencePack.zh-CN.Neutral.Appx](#)
- 5.3.2.2.

Manual download

5.3.2.2.1.

Region

Download Region: zh-CN, application ID: [9NRMNT6GMZ70](#), Store link: <https://www.microsoft.com/store/productId/9NRMNT6GMZ70>

5.3.2.2.2.

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

5.3.2.2.2.1.

Search keywords:

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

5.3.2.2.2.2.

Search [22621](#) content in the web page, search results: [Microsoft.LanguageExperiencePackzh-CN_22621.*.neutral_8wekyb3d8bbwe.appx](#)

5.3.2.2.2.3.

After downloading, save it to the [D:\OS_10_Custom\Install\Install\InBox.Appx](#) directory and rename it: [LanguageExperiencePack.zh-cn.Neutral.Appx](#)

5.3.2.3.

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

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

```
Add-AppxProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" -PackagePath "D:\OS_10_Custom\Install\Install\InBox.appx\LanguageExperiencePack.zh-cn.Neutral.appx" -SkipLicense
```

5.3.2.4.

InBox Apps: An installed application package

5.3.2.4.1.

View

```
Get-AppXProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Out-GridView
```

5.3.2.4.2.

Export to Csv

```
$SaveTo = "D:\OS_10_Custom\Install\Install\Report.$(Get-Date -Format "yyyyMMddHHmmss").csv"

Get-AppXProvisionedPackage -Path "D:\OS_10_Custom\Install\Install\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green
```

5.4. InBox Apps: Install

5.4.1. Mount or decompress the InBox Apps installation file

Mount [19041.3031.230508-1728.vb_release_svc_prod3_amd64fre_InboxApps.iso](#) or extract to any location;

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

- [Auto](#) = Automatically search all local disks, default;
- Custom path, e.g. specify F drive: [\\$ISO = "F:\packages"](#)
- [Install.Inst.InBox.Appx.ps1](#)
 - [\Expand\Install\Install.Inst.InBox.Appx.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Install/Install.Inst.InBox.Appx.ps1

- Copy the code

```
$ISO = "Auto"

$Mount = "D:\OS_10_Custom\Install\Install\Mount"

try{

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

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

    Write-Host $Current_Edition_Version -ForegroundColor Green

} catch {

    Write-Host "Error" -ForegroundColor Red

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

    return

}

$Pre_Config_Rules = @(

    @{

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

        Apps = @(

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

            "Microsoft.NET.Native.Framework.2.2"; "Microsoft.NET.Native.Runtime.2.2"; "Microsoft.VCLibs.140.00";

            "Microsoft.VCLibs.140.00.UWPDesktop"; "Microsoft.HEIFImageExtension"; "Microsoft.HEVCVideoExtension";

            "Microsoft.SecHealthUI"; "Microsoft.VP9VideoExtensions"; "Microsoft.WebpImageExtension";

            "Microsoft.WindowsStore"; "Microsoft.GamingApp"; "Microsoft.Sticky.Notes"; "Microsoft.Paint";

            "Microsoft.PowerAutomateDesktop"; "Microsoft.ScreenSketch"; "Microsoft.WindowsNotepad";

            "Microsoft.WindowsTerminal"; "Clipchamp.Clipchamp"; "Microsoft.Solitaire.Collection"; "Microsoft.WindowsAlarms";

            "Microsoft.WindowsFeedbackHub"; "Microsoft.WindowsMaps"; "Microsoft.ZuneMusic"; "Microsoft.BingNews";

            "Microsoft.DesktopAppInstaller"; "Microsoft.WindowsCamera"; "Microsoft.Getstarted"; "Microsoft.Cortana";

            "Microsoft.BingWeather"; "Microsoft.GetHelp"; "Microsoft.MicrosoftOfficeHub"; "Microsoft.People";

            "Microsoft.StorePurchaseApp"; "Microsoft.Todos"; "Microsoft.WebMediaExtensions"; "Microsoft.Windows.Photos";
```

```

"Microsoft.WindowsCalculator"; "Microsoft.Windows.CommunicationsApps"; "Microsoft.WindowsSoundRecorder";

"Microsoft.Xbox.TCUI"; "Microsoft.XboxGameOverlay"; "Microsoft.XboxGamingOverlay";

"Microsoft.XboxIdentityProvider"; "Microsoft.XboxSpeechToTextOverlay"; "Microsoft.YourPhone";

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

"Microsoft.RawImageExtension"; "MicrosoftCorporationII.MicrosoftFamily";

    )

}

@{

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

    Apps = @(

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

    )

}

@{

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

    Apps = @(

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

    )

```



```

    }

)

$Allow_Install_App = @()

ForEach ($item in $Pre_Config_Rules) {

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

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

        Write-host $Current_Edition_Version -ForegroundColor Green

        $Allow_Install_App = $item.Apps

        break

    }

}

Write-host "`n  The app to install ( $($Allow_Install_App.Count) item )" -ForegroundColor Yellow

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

ForEach ($item in $Allow_Install_App) {

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

}

$InBoxApps = @(

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

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

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

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

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

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

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

    @{ Name = "Microsoft.WindowsStore"; File = "Microsoft.WindowsStore_8wekyb3d8bbwe.msixbundle"; License =
    "Microsoft.WindowsStore_8wekyb3d8bbwe.xml"; }

    @{ Name = "Microsoft.HEIFImageExtension"; File = "Microsoft.HEIFImageExtension_8wekyb3d8bbwe.appxbundle";
    License = "Microsoft.HEIFImageExtension_8wekyb3d8bbwe.xml"; }

    @{ Name = "Microsoft.HEVCVideoExtension"; File = "Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.x64.appx";
    License = "Microsoft.HEVCVideoExtension_8wekyb3d8bbwe.x64.xml"; }

    @{ Name = "Microsoft.SecHealthUI"; File = "Microsoft.SecHealthUI_8wekyb3d8bbwe.x64.appx"; License =
    "Microsoft.SecHealthUI_8wekyb3d8bbwe.x64.xml"; }

    @{ Name = "Microsoft.VP9VideoExtensions"; File = "Microsoft.VP9VideoExtensions_8wekyb3d8bbwe.x64.appx";
    License = "Microsoft.VP9VideoExtensions_8wekyb3d8bbwe.x64.xml"; }

    @{ Name = "Microsoft.WebplImageExtension"; File = "Microsoft.WebplImageExtension_8wekyb3d8bbwe.x64.appx";
    License = "Microsoft.WebplImageExtension_8wekyb3d8bbwe.x64.xml"; }

```

```
@{ Name = "Microsoft.GamingApp"; File = "Microsoft.GamingApp_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.GamingApp_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Sticky.Notes"; File = "Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe.msixbundle"; License  
= "Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Paint"; File = "Microsoft.Paint_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.Paint_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.PowerAutomateDesktop"; File =  
"Microsoft.PowerAutomateDesktop_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.PowerAutomateDesktop_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.ScreenSketch"; File = "Microsoft.ScreenSketch_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.ScreenSketch_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsNotepad"; File = "Microsoft.WindowsNotepad_8wekyb3d8bbwe.msixbundle";  
License = "Microsoft.WindowsNotepad_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsTerminal"; File = "Microsoft.WindowsTerminal_8wekyb3d8bbwe.msixbundle";  
License = "Microsoft.WindowsTerminal_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Clipchamp.Clipchamp"; File = "Clipchamp.Clipchamp_yxz26nhyzhsrt.msixbundle"; License =  
"Clipchamp.Clipchamp_yxz26nhyzhsrt.xml"; }
```

```
@{ Name = "Microsoft.Solitaire.Collection"; File =  
"Microsoft.MicrosoftSolitaireCollection_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.MicrosoftSolitaireCollection_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsAlarms"; File = "Microsoft.WindowsAlarms_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.WindowsAlarms_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsFeedbackHub"; File =  
"Microsoft.WindowsFeedbackHub_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.WindowsFeedbackHub_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsMaps"; File = "Microsoft.WindowsMaps_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.WindowsMaps_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.ZuneMusic"; File = "Microsoft.ZuneMusic_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.ZuneMusic_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "MicrosoftCorporationII.MicrosoftFamily"; File =  
"MicrosoftCorporationII.MicrosoftFamily_8wekyb3d8bbwe.msixbundle"; License =  
"MicrosoftCorporationII.MicrosoftFamily_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.BingNews"; File = "Microsoft.BingNews_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.BingNews_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.DesktopAppInstaller"; File = "Microsoft.DesktopAppInstaller_8wekyb3d8bbwe.msixbundle";  
License = "Microsoft.DesktopAppInstaller_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsCamera"; File = "Microsoft.WindowsCamera_8wekyb3d8bbwe.msixbundle"; License  
= "Microsoft.WindowsCamera_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Getstarted"; File = "Microsoft.Getstarted_8wekyb3d8bbwe.msixbundle"; License =  
"Microsoft.Getstarted_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Cortana"; File = "Microsoft.CortanaApp_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.CortanaApp_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.BingWeather"; File = "Microsoft.BingWeather_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.BingWeather_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.GetHelp"; File = "Microsoft.GetHelp_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.GetHelp_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.MicrosoftOfficeHub"; File = "Microsoft.MicrosoftOfficeHub_8wekyb3d8bbwe.appxbundle";  
License = "Microsoft.MicrosoftOfficeHub_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.People"; File = "Microsoft.People_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.People_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.StorePurchaseApp"; File = "Microsoft.StorePurchaseApp_8wekyb3d8bbwe.appxbundle";  
License = "Microsoft.StorePurchaseApp_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Todos"; File = "Microsoft.Todos_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.Todos_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WebMediaExtensions"; File = "Microsoft.WebMediaExtensions_8wekyb3d8bbwe.appxbundle";  
License = "Microsoft.WebMediaExtensions_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Windows.Photos"; File = "Microsoft.Windows.Photos_8wekyb3d8bbwe.appxbundle"; License  
= "Microsoft.Windows.Photos_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsCalculator"; File = "Microsoft.WindowsCalculator_8wekyb3d8bbwe.appxbundle";  
License = "Microsoft.WindowsCalculator_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Windows.CommunicationsApps"; File =  
"Microsoft.WindowsCommunicationsApps_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.WindowsCommunicationsApps_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.WindowsSoundRecorder"; File =  
"Microsoft.WindowsSoundRecorder_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.WindowsSoundRecorder_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.Xbox.TCUI"; File = "Microsoft.Xbox.TCUI_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.Xbox.TCUI_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.XboxGameOverlay"; File = "Microsoft.XboxGameOverlay_8wekyb3d8bbwe.appxbundle";  
License = "Microsoft.XboxGameOverlay_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.XboxGamingOverlay"; File = "Microsoft.XboxGamingOverlay_8wekyb3d8bbwe.appxbundle";  
License = "Microsoft.XboxGamingOverlay_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.XboxIdentityProvider"; File = "Microsoft.XboxIdentityProvider_8wekyb3d8bbwe.appxbundle";  
License = "Microsoft.XboxIdentityProvider_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.XboxSpeechToTextOverlay"; File =  
"Microsoft.XboxSpeechToTextOverlay_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.XboxSpeechToTextOverlay_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.YourPhone"; File = "Microsoft.YourPhone_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.YourPhone_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "Microsoft.ZuneVideo"; File = "Microsoft.ZuneVideo_8wekyb3d8bbwe.appxbundle"; License =  
"Microsoft.ZuneVideo_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "MicrosoftCorporationII.QuickAssist"; File =  
"MicrosoftCorporationII.QuickAssist_8wekyb3d8bbwe.appxbundle"; License =  
"MicrosoftCorporationII.QuickAssist_8wekyb3d8bbwe.xml"; }
```

```
@{ Name = "MicrosoftWindows.Client.WebExperience"; File =  
"MicrosoftWindows.Client.WebExperience_cw5n1h2txyewy.appxbundle"; License =  
"MicrosoftWindows.Client.WebExperience_cw5n1h2txyewy.xml"; }
```

```
@{ Name = "Microsoft.RawImageExtension"; File = "Microsoft.RawImageExtension_8wekyb3d8bbwe.appxbundle";
```

```

License = "Microsoft.RawImageExtension_8wekyb3d8bbwe.xml"; }

)

Function Install_Appx

{

    param ( $File, $License )

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

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

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

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

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

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

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

            try {

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

                Write-Host "Done" -ForegroundColor Green

            } catch {

                Write-Host "Failed" -ForegroundColor Red

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

            }

        } else {

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

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

            try {

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

                Write-Host "Done" -ForegroundColor Green

            } catch {

                Write-Host "Failed" -ForegroundColor Red

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

            }

        }

    } else {

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

    }

}

```

```

}

ForEach ($Rule in $InBoxApps) {

    Write-host "`n  Name: " -NoNewline

    Write-host $Rule.Name -ForegroundColor Yellow

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

    if($Allow_Install_App -contains $Rule.Name) {

        Write-host "  Search for apps: " -NoNewline

        Write-host $Rule.File -ForegroundColor Yellow

        Write-host "  Search for License: " -NoNewline

        Write-host $Rule.File -ForegroundColor Yellow

        if ($ISO -eq "Auto") {

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

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

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

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

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

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

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

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

                    } else {

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

                    }

                    Install_Appx -File $AppPath -License $LicensePath

                    return

                }

            }

        } else {

            Install_Appx -File "$($ISO)\$($Rule.File)" -License "$($ISO)\$($Rule.License)"

        }

    } else {

        Write-host "  Skip the installation" -ForegroundColor Red

    }

}

```

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

```
Dism /Image:"D:\OS_10_Custom\Install\Install\Mount" /Optimize-ProvisionedAppxPackages
```

6. Cumulative updates

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

6.1. Feature enablement package

- Learn: [KB5015684: Featured update to Windows 10, version 22H2 by using an enablement package](#)
- Note: [When using the same version as Windows 10 22H2 19045.2006 and above, the feature enablement package can be skipped.](#)
- When making Windows 10 versions 2004, 20H2, 21H1, and 21H2, you need to install the "Feature Enablement Package" in advance before you can install the cumulative update of 19045.*. After downloading, save it to: [D:\OS_10_Custom\Install\Install\Update](#), select according to the architecture download:

6.1.1. x64, default

- Direct download

https://catalog.s.download.windowsupdate.com/c/upgr/2022/07/windows10.0-kb5015684-x64_523c039b86ca98f2d818c4e6706e2cc94b634c4a.msu

- Install

```
$KBPath = "D:\OS_10_Custom\Install\Install\Update\windows10.0-kb5015684-x64_523c039b86ca98f2d818c4e6706e2cc94b634c4a.msu"
```

```
Add-WindowsPackage -Path "D:\OS_10_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.2. Initial version

Cumulative update KB5017308 cannot be searched from <https://www.catalog.update.microsoft.com/Search.aspx?q=KB5017308>. Download it through direct connection and save it to: [D:\OS_10_Custom\Install\Install\Update](#), choose to download according to the architecture:

6.2.1. x64, default

- Direct download

https://catalog.s.download.windowsupdate.com/c/msdownload/update/software/secu/2022/09/windows10.0-kb5017308-x64_2027053968a06948b45d139d95475ab4feee5654.msu

- Install

```
$KBPath = "D:\OS_10_Custom\Install\Install\Update\windows10.0-kb5017308-
x64_2027053968a06948b45d139d95475ab4feee5654.msu"

Add-WindowsPackage -Path "D:\OS_10_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.3. Other versions

Check "[Windows 10 version information](#)", for example, download the cumulative update: [KB5032278](#), go to the download page: <https://www.catalog.update.microsoft.com/Search.aspx?q=KB5032278>, and save it to: [D:\OS_10_Custom\Install\Install\Update](#), or download through direct connection, select download according to the architecture:

6.3.1. x64, default

- Direct download

https://catalog.s.download.windowsupdate.com/d/msdownload/update/software/updt/2023/11/windows10.0-kb5032278-x64_3a2b384f690eaa92c5030c535ac4a7e47c71a635.msu

- Add to

```
$KBPath = "D:\OS_10_Custom\Install\Install\Update\windows10.0-kb5032278-
x64_3a2b384f690eaa92c5030c535ac4a7e47c71a635.msu"

Add-WindowsPackage -Path "D:\OS_10_Custom\Install\Install\Mount" -PackagePath $KBPath
```

6.4. 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:\OS_10_Custom\Install\Install\Mount" /cleanup-image /StartComponentCleanup /ResetBase
```

6.4.1. Clean components after curing and updating, optional

```
$Mount = "D:\OS_10_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. 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.

8. Health

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

```
Repair-WindowsImage -Path "D:\OS_10_Custom\Install\Install\Mount" -ScanHealth
```

9. Replace WinRE.wim

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

```
$WinRE = "D:\OS_10_Custom\Install\Install\Update\Winlib\WinRE.wim"
```

```
$CopyTo = "D:\OS_10_Custom\Install\Install\Mount\Windows\System32\Recovery"
```

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

10. Save image: Install.wim

```
Save-WindowsImage -Path "D:\OS_10_Custom\Install\Install\Mount"
```

11. Unmount image: Install.wim

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

```
Dismount-WindowsImage -Path "D:\OS_10_Custom\Install\Install\Mount" -Discard
```

LOOP OPERATING AREA, END.

12. Rebuilding Install.wim reduces file size

- Install.Rebuild.wim.ps1
 - [\Expand\Install\Install.Rebuild.wim.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Install/Install.Rebuild.wim.ps1

- Copy the code

```
$InstallWim = "D:\OS_10\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

}

```

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

13.1. Obtain WimLib

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

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

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

- Install.WinRE.Extract.ps1
 - [\Expand\Install\Install.WinRE.Extract.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Install/Install.WinRE.Extract.ps1

- Copy the code

```

$Arguments = @(

    "extract",

    "D:\OS_10\sources\install.wim", "1",

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

    "--dest-dir=""D:\OS_10_Custom\Install\Install\Update\Winlib""

)

New-Item -Path "D:\OS_10_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

- Install.WinRE.Replace.wim.ps1
 - [\Expand\Install\Install.WinRE.Replace.wim.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Install/Install.WinRE.Replace.wim.ps1

- Copy the code

```

Get-WindowsImage -ImagePath "D:\OS_10\sources\install.wim" -ErrorAction SilentlyContinue | ForEach-Object {

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

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

    Write-Host "`n  Replacement "

```

```

$Arguments = @(
    "update",
    "D:\OS_10\sources\install.wim", $_.ImageIndex,
    "--command=""add 'D:\OS_10_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
}

```

III Custom deployment image boot.wim

1. View Boot.wim details

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

```

$ViewFile = "D:\OS_10\Sources\Boot.wim"

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

```

2. Specify the path to mount Boot.wim

```
New-Item -Path "D:\OS_10_Custom\Boot\Boot\Mount" -ItemType directory -ea SilentlyContinue
```

3. Start mounting Boot.wim

Default index number: 2

```
Mount-WindowsImage -ImagePath "D:\OS_10\sources\boot.wim" -Index "2" -Path "D:\OS_10_Custom\Boot\Boot\Mount"
```

4. Language pack: Boot

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

4.1. Language pack: Add

- Boot.Instl.lang.ps1
 - [\Expand\Boot\Boot.Instl.lang.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/Boot/Boot.Instl.lang.ps1

- Copy the code

```

$Mount = "D:\OS_10_Custom\Boot\Boot\Mount"

$Sources = "D:\OS_10_Custom\Boot\Boot\Language\Add\zh-CN"

```

```

$Initl_install_Language_Component = @(

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

    $Initl_install_Language_Component += $_.PackageName

}

Add-WindowsPackage -Path $Mount -PackagePath "$($Sources)\WinPE-FontSupport-zh-CN.cab"

$Language = @(

    @{ Match = "*WinPE*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) {

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

    ForEach ($Component in $Initl_install_Language_Component) {

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

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

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

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

            try {

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

                Write-host "Finish" -ForegroundColor Green

            } catch {

                Write-host "Failed" -ForegroundColor Red
            }
        }
    }
}

```

```

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

    }

    break

}

}

}

```

4.2. Components: All packages installed in the image

4.2.1. View

```
Get-WindowsPackage -Path "D:\OS_10_Custom\Boot\Boot\Mount" | Out-GridView
```

4.2.2. Export to csv

```

$SaveTo = "D:\OS_10_Custom\Boot\Boot\Report.${(Get-Date -Format "yyyyMMddHHmmss")}.csv"

Get-WindowsPackage -Path "D:\OS_10_Custom\Boot\Boot\Mount" | Export-CSV -NoType -Path $SaveTo

Write-host $SaveTo -ForegroundColor Green

```

4.3. Language: Repair

4.3.1. Extract

Open: [D:\OS_10_Custom\Install\Install\Language\Add\zh-CN\Microsoft-Windows-Client-Language-Pack_x64_zh-CN.cab](#), enter the directory: [Setup\sources\zh-cn\cli](#), and copy the following files to the deskto:

- 4.3.1.1. [arunres.dll.mui](#)
- 4.3.1.2. [spwizres.dll.mui](#)
- 4.3.1.3. [w32uires.dll.mui](#)

4.3.2. Copy

Copy the extracted files to: [D:\OS_10_Custom\Boot\Boot\Mount\sources\zh-CN](#)

4.4. Language packs: sync to ISO installer

```
Copy-Item -Path "D:\OS_10_Custom\Boot\Boot\Mount\sources\zh-CN" -Destination "D:\OS_10\sources\zh-CN" -Recurse -Force
```

4.5. Regenerate Lang.ini

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

4.5.1. Regenerate the mounted directory lang.ini

Regenerated Lang.ini file location: D:\en-us_windows_server_2022_x64_dvd_620d7eac_Custom\Boot\Boot\Mount\Sources\lang.ini

```
Dism /image:"D:\OS_10_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS_10_Custom\Boot\Boot\Mount"
```

4.5.2. 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:\OS_10_Custom\Boot\Boot\Mount" /gen-langini /distribution:"D:\OS_10"
```

5. Save image: Boot.wim

```
Save-WindowsImage -Path "D:\OS_10_Custom\Boot\Boot\Mount"
```

6. Unmount image: Boot.wim

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

```
Dismount-WindowsImage -Path "D:\OS_10_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:\OS_10\Autounattend.xml](#)

```
Copy-Item "D:\Multilingual\Unattend\Mul.Unattend.xml" -Destination "D:\OS_10\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:\OS_10\Sources\Unattend.xml](#)

```
Copy-Item "D:\Multilingual\Unattend\Mul.Unattend.xml" -Destination "D:\OS_10\Sources\Unattend.xml" -Force
```

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

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

1.1.1.3.1. Create \$OEM\$ path

```
New-Item -Path "D:\OS_10\sources\`$OEM$\`$$\Panther" -ItemType Directory
```

1.1.1.3.2. Copy

Copy D:\Multilingual\Unattend\Mul.Unattend.xml to
D:\OS_10\Sources\\$OEM\$\Panther\Unattend.xml

```
Copy-Item "D:\Multilingual\Unattend\Mul.Unattend.xml" -Destination  
"D:\OS_10\sources\`$OEM$\`$$\Panther\Unattend.xml" -Force
```

1.1.2. Deployment engine: add

Add "Automatically add installed languages for Windows systems" to D:\OS_10\sources\\$OEM\$\\$1\Yi\Engine in the directory.

1.1.2.1. Deployment Engine: Copy

Copy D:\Multilingual\Engine to D:\OS_10\Sources\\$OEM\$\\$1\Yi\Engine

```
Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS_10\sources\`$OEM$\`$1\Yi\Engine" -Recurse -  
Force
```

1.1.2.2. Deployment engine: custom deployment tags

```
$Flag = @(
```

```
"Is_Mark_Sync" # Allow global search and synchronization of deployment tags
```

```
# Prerequisite deployment
```

```
# "Auto_Update" # Allow automatic updates
```

```
# "Use_UTF8" # Beta: Global language support using Unicode UTF-8
```

```
"Disable_Network_Location_Wizard" # Network Location Wizard
```

```
"Disable_Cleanup_Appx_Tasks" # Appx Cleanup and maintenance tasks
```

```
"Disable_Cleanup_On_Demand_Language" # Prevent cleanup of unused on-demand feature language  
packs
```

```
"Disable_Cleanup_Unsed_Language" # Prevent cleaning of unused language packs
```

```
"Prerequisites_Reboot" # Restart your computer
```

```
# Complete first deployment
```

```
# "Popup_Engine" # Allow the deployment engine main interface to pop up for the first time
```

```
# "Allow_First_Pre_Experience" # Allow first preview, as planned
```

```
"Reset_Execution_Policy" # Restore PowerShell execution policy: Restricted
```

```
"Clear_Solutions" # Delete the entire solution
```

```

"Clear_Engine" # Delete the deployment engine and keep the others

# "First_Experience_Reboot" # Restart your computer

)

ForEach ($item in $Flag) {

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

    New-Item -Path "D:\OS_10\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -
    ErrorAction SilentlyContinue | Out-Null

    Out-File -FilePath "D:\OS_10\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:\OS_10_Custom\Install\Install\Mount](#)

1.2.1. Unattended

Copy [D:\Multilingual\Unattend\Mul.Unattend.xml](#) to [D:\OS_10_Custom\Install\Install\Mount\Panther\Unattend.xml](#)

```
Copy-Item "D:\Multilingual\Unattend\Mul.Unattend.xml" -Destination "D:\OS_10_Custom\Install\Install\Mount\Panther" -
Force
```

1.2.2. Deployment engine: add

Add "[Automatically add languages installed on Windows systems](#)" to the
[D:\OS_10_Custom\Install\Install\Mount\Yi\Engine](#) directory.

1.2.2.1. Deployment Engine: Copy

Copy [D:\Multilingual\Engine](#) to [D:\OS_10_Custom\Install\Install\Mount\Yi\Engine](#)

```
Copy-Item "D:\Multilingual\Engine" -Destination "D:\OS_10_Custom\Install\Install\Mount\Yi\Engine" -
Recurse -Force
```

1.2.2.2. Deployment engine: custom deployment tags

```

$Flag = @(

    "Is_Mark_Sync" # Allow global search and synchronization of deployment tags

    # Prerequisite deployment

    # "Auto_Update" # Allow automatic updates

    # "Use_UTF8" # Beta: Global language support using Unicode UTF-8

    "Disable_Network_Location_Wizard" # Network Location Wizard

    "Disable_Cleanup_Appx_Tasks" # Appx Cleanup and maintenance tasks

    "Disable_Cleanup_On_Demand_Language" # Prevent cleanup of unused on-demand feature language

```

```

packs

"Disable_Cleanup_Unsed_Language" # Prevent cleaning of unused language packs

"Prerequisites_Reboot" # Restart your computer

# Complete first deployment

# "Popup_Engine" # Allow the deployment engine main interface to pop up for the first time

# "Allow_First_Pre_Experience" # Allow first preview, as planned

"Reset_Execution_Policy" # Restore PowerShell execution policy: Restricted

"Clear_Solutions" # Delete the entire solution

"Clear_Engine" # Delete the deployment engine and keep the others

# "First_Experience_Reboot" # Restart your computer

)

ForEach ($item in $Flag) {

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

    New-Item -Path "D:\OS_10\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow" -ItemType Directory -
    ErrorAction SilentlyContinue | Out-Null

    Out-File -FilePath "D:\OS_10\sources\`$OEM$\`$1\Yi\Engine\Deploy\Allow\$( $item)" -Encoding utf8 -
    ErrorAction SilentlyContinue

}

```

2. Deployment Engine: Advanced

2.1. Deployment engine: adding process

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

2.2. Unattended solution

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

- [D:\OS_10\Autounattend.xml](#)
- [D:\OS_10\Sources\Unattend.xml](#)
- [D:\OS_10\sources\\\$OEM\\$\\\$\\$\Panther\unattend.xml](#)
- [D:\OS_10_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>%OSDSysSystemLocale%</SystemLocale>

<UILanguage>%OSDUILanguage%</UILanguage>

<UILanguageFallback>%OSDUILanguageFallback%</UILanguageFallback>

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

2.2.1.2. Monolingual

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

```
<UILanguage>zh-CN</UILanguage>

<InputLocale>zh-CN</InputLocale>

<SystemLocale>zh-CN</SystemLocale>

<UILanguage>zh-CN</UILanguage>

<UILanguageFallback>zh-CN</UILanguageFallback>

<UserLocale>zh-CN</UserLocale>
```

2.2.2. User plan

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

2.2.2.1. Self-created user Administrator

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

```
<UserAccounts>

<LocalAccounts>

<LocalAccount wcm:action="add">

<Password>

<Value></Value>

<PlainText>true</PlainText>

</Password>

<Description>Administrator</Description>

<DisplayName>Administrator</DisplayName>

<Group>Administrators</Group>

<Name>Administrator</Name>

</LocalAccount>

</LocalAccounts>

</UserAccounts>
```

```
<AutoLogon>

<Password>

<Value></Value>

<PlainText>>true</PlainText>

</Password>

<Enabled>>true</Enabled>

<Username>Administrator</Username>

</AutoLogon>
```

2.2.2.2. Custom user

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

2.2.2.2.1. Delete

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

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

2.2.2.2.2. Replace

From the beginning <OOBE> to </OOBE>

```
<OOBE>

<ProtectYourPC>3</ProtectYourPC>

<HideEULAPage>true</HideEULAPage>

<HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>

</OOBE>
```

D. Generate ISO

II Download OScdimg

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

1.1. x64

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

1.2. x86

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

1.3. arm64

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

III Use the oscding command line to generate an ISO file and save it to: [D:\OS_10.iso](#)

- ISO.ps1
 - [\Expand\ISO.ps1](#)
 - https://github.com/ilikeyi/solutions/blob/main/_Learn/Attachment/OS.10/22H2/Expand/ISO.ps1

- Copy the code

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

```
$ISO = "D:\OS_10"
```

```
$Volume = "OS_10"
```

```
$SaveTo = "D:\OS_10.iso"
```

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

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



Author: Yi

Website: <https://fengyi.tel>

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

Document version: 1.0

Documentation model: [Lite Version](#)

Translation: [Chinese to English version](#)

Updated: 2024 - 4

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