Starter Guide for Windows 10 IoT Enterprise 2nd Edition, Addendum 3: Language Package Integration

By Sean D. Liming and John R. Malin
Annabooks – www.annabooks.com

September 2024

Windows 10 LTSC 2021 and Windows 11 LTSC 2024

1.1 *Important Feature: Targeting Locals*

Multilanguage support has been a key feature of Windows for several decades now. For Embedded/IoT OEMs building products with Windows, the ability to provide the product in a local language opens up new markets and increases sales. OEMs typically make the main application as the shell, so the user only sees the main application and not the Windows desktop. All language controls and fonts are handled in the main application. There are scenarios when the OEM wants to provide a maintenance mode that allows access to the Windows desktop. In the maintenance mode scenario, being able to choose access to the Windows desktop needs to appear in the local language and local formats. Windows Language Packages (LP) (the old name was MUI packs) add the resources for Windows desktop to appear in the local language. The language packages come as .cab files. Besides the language packages, there are language extension packages, which come as .appx packages. Does this sound confusing? The focus of this addendum is to clarify the details and show how to add the Windows Language Packages to Windows IoT Enterprise.

1.2 *History: The Big Project going from Windows XP to Windows 7 and now Windows 10 and 11*

Adding language packages to Windows today is much simpler than it used to be in the past. Back in the Windows XP Embedded days, I had several medical clients who wanted to build a single Windows XP Embedded OS with multiple language package support. Since medical devices have to go through stringent regulations and testing, having one image supporting multiple languages was cheaper to test than multiple individual Windows images each dedicated to one of the supported languages. The problem was with all languages installed, it took several hours to build and install the Windows XP Embedded OS image. The image size would blow up, and sometimes the build would crash.

To that point in time, language support was very messy in Windows. As an example, there were several editions of Notepad to address different languages. If you installed all the support languages at the time, you would get multiple binaries of the same product written for different languages, which would blow up the image size. One of the big projects when taking a system from Windows XP to Windows 7, via pre-alpha Windows 7 release (aka Windows Vista, sorry couldn't resist), was to break out the languages, so there was only one binary. Microsoft accomplished this by making all the UI binaries language-neutral. The language packages would add the language-specific UI to the binary. Now, there is only one notepad binary.

For Windows 7, multiple language packages could be added to a Windows OS much easier, quicker, and with better success than Windows XP. Most important was the minimal impact on the image size. The languages would be available to all users. Windows 10 and 11 throw a little wrench into the works by making languages user-account-specific, or so it seems.

1.3 *ISO 639 Standard, LIP versus LXP, and the Important DVD ISOs,*

To help programmers reference the different languages around the world, the ISO 639 Standard defines a standard nomenclature to provide two or three-letter codes for all known languages. For example, English is en, French is fr, and Japanese is ja. Microsoft expands on the ISO 639 for

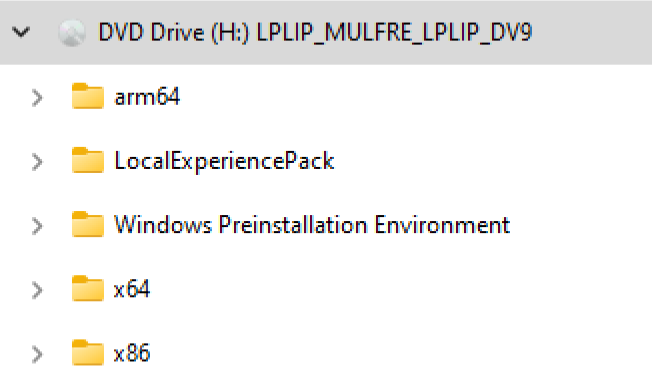
Windows by using a Language-Region code. For example, English (United States) is en-US, English (United Kingdom) is en-GB. The complete table of language packages and their codes can be found here: Available Language Packs for Windows | Microsoft Learn. The web page is important because it lists the available language packages and language interface packages (LIPs) / language extension packages (LXPs). Language packages provide a complete translation for the Windows UI, but there are only 38 of them. Windows supports 110+ languages. This is where the LIP/LXPs come in. The idea here is to have one of the base LPs installed and then add an LIP package to complete the translation to the specific local language. Going back to the Microsoft Learn webpage there is a second table with the remaining languages. The table lists the LIP/LXP and the base language to use. For example, to support Irish (Ireland), the en-US or en-GB language packages get installed first and then the LIP/LXP ga-IE gets installed second.

What is the difference between LIP and LXP? Nothing. LXP is the new name going forward, but the term LIP has been used for so long that not all the documentation and other little items have caught up to the change.

The Language packages and LXPs can be downloaded as DVD ISOs from a Visual Studio Subscription or the Digital Operation Center, if you are an OEM that has signed a license agreement.

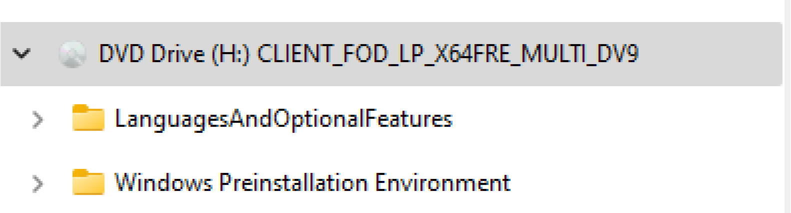
For Windows 10, there are three DVD ISOs. The first is the language pack ISO that contains the LPs, LXPs, and the Language packages for WinPE. The second and third are the two DVD ISOs that make up the Features-on-Demand (FOD) packages. The FODs provide language support for handwriting recognition, speech recognition, fonts, OCR, and text-to-speech. There is also the SSH server and client, .NET Framework 3 installation, retail demo, and a few other items.

The picture below shows what is in the Windows 10 LP DVD ISO:



DVD Drive (H:) LPLIP_MULFRE_LPLIP_DV9
- arm64
- LocalExperiencePack
- Windows Preinstallation Environment
- x64
- x86

For Windows 11, the LPs, Features on Demand, and WinPE Language packages are all in one DVD ISO. LXPs are not included in the Windows 11 IoT LTSC 24H2 language ISO release. The picture below shows the high-level contents of the DVD:



DVD Drive (H:) CLIENT_FOD_LP_X64FRE_MULTI_DV9
- LanguagesAndOptionalFeatures
- Windows Preinstallation Environment

The actual Language Packs and FODs come as part of the ISOs from Microsoft. Please, be aware of websites that say they offer language packages as these might not be legitimate releases. The ISOs can be found in the Visual Studio subscription or the Digital Operation Center, if you have a signed license agreement. Finding the right version takes a little work.

Windows IoT Enterprise Release	Language Package DVD ISO	Notes
Windows 10 IoT LTSC 2021	Windows 10, version 2004, 20H2, 21H1, 21H2 or 22H2 Language Packs	The DVD ISO was found in the Visual Studio Subscription. A version was not found in the Digital Operation Center.
Windows 11 IoT LTSC 2024	X23-70026 Win 11 24H2 x64 MultiLang OPK LangPackAllLIP LoF	A version for LTSC 2024 (24H2) was not found in Visual Studio. The ISO was found in the Digital Operation Center. You must have a signed license agreement to download this ISO.

Important: There is an important order of operation that must be followed when setting up the local languages. The LPs must be installed first followed by any LXPs. Finally, any FODs are installed last. You will also need to run Windows Update as the languages need the latest updates to be configured properly.

Note: Language packages can only be applied to the correct version. The Windows 11 language packages cannot be applied to Windows 10. For Windows 10 LTSC 2021, the Version 19041 language package release is the correct version.

Note: The current Windows 11 IoT LTSC 24H2 language ISO release doesn't include the LXPs, but later versions include the LXPs. You cannot use the later versions with the Windows 11 IoT LTSC 24H2.

Note: WinPE and WinRE cannot use the big Windows language packages, the LXPs, or the FODs. There is a specific set of LPs for WinPE customization. These come with the ADK WinPE Add-on and the aforementioned Language package DVD.

1.4 DISM and PowerShell Cmdlets

There are several tools to install and configure the LPs, LXPs, and FODs in a Windows OS. The all-purpose tool DISM.EXE is used to install .CAB and .APPX files. Here is an example of installing an LP:

```
dism /online /add-package /packagepath:c:\setup\Microsoft-Windows-Client-Language-Pack_x64_ar-sa.cab
```

PowerShell can also be used to install a CAB file:

```
Add-WindowsPackage -Online -PackagePath "c:\setup\Microsoft-Windows-Client-Language-Pack_x64_ar-sa.cab "
```

PowerShell has a number of useful Cmdlets to manage languages.

Cmdlet	Description
Get-InstalledLanguage	Returns information about the installed languages on a device.
Get-SystemPreferredUILanguage	Returns the current System Preferred Language.

Install-Language	Downloads and Installs a language onto a device.
Set-Culture	Sets the user culture for the current user account.
Set-SystemPreferredUILanguage	Sets the provided language as the System Preferred UI Language.
Set-WinSystemLocale	Sets the system locale for the current computer.
Set-WinUILanguageOverride	Sets the Windows UI language override setting for the current user account.
Set-WinUserLanguageList	Sets the language list and associated properties for the current user account. The list can be viewed via Settings->Time & language->Language & region.
Set-WinHomeLocation	Sets the home location setting for the current user account.
Uninstall-Language	Uninstalls a language from a device.

Note: Windows 10 requires a Cumulative Update to get access to some of these Language Cmdlets.

With these details covered, the next few sections look at the challenges and solutions for getting the language packages installed.

1.5 Research Result 1: Basic Answer File and Installation Media

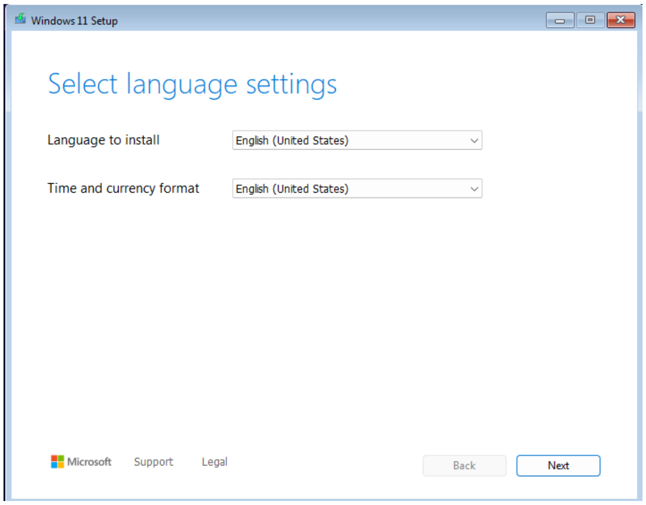
Rather than going through full exercises, this section and the next few sections provide the research results on setting up languages. For reference, Windows 11 IoT Enterprise LTSC 2024 and the build process covered in the book *Starter Guide for Windows® 10 IoT Enterprise 2nd Edition* were used to conduct the research.

Let’s look at setting up the languages in the answer file. Two components added to an answer file deal with language. One is for WinPE and the other is for the actual OS itself. The table below shows the different language settings that get set with the Language-Region code.

Component	Configuration Pass	Component Setting	Possible Value / Notes
amd64_Microsoft-Windows-International-Core-WinPE__neutral	1 WindowsPE	InputLocale	en-US
	1 WindowsPE	SystemLocale	en-US
	1 WindowsPE	SetupUILanguage: UILanguage	en-US
	1 WindowsPE	UILanguage	en-US
	1 WindowsPE	UserLocale	en-US
	1 WindowsPE	UILanguageFallback	en-US
amd64_Microsoft-Windows-International-Core__neutral	7 OOBE System	InputLocale	en-US
	7 OOBE System	SystemLocale	en-US
	7 OOBE System	UILanguage	en-US
	7 OOBE System	UserLocale	en-US

The actual Windows 10 IoT Enterprise LTSC 2024 OPK ISO that I have is English-based, which means that the English language package is part of the Windows OS installation. If I change the above settings to French (France) fr-FR instead of English (United States) en-US, here is what happens:

1. During the WinPE Pass, the wizard appears and asks to select the language. English is the only option so the setup Wizard will only be in English. If the component settings were set to en-US, this screen would be bypassed.



Windows 11 Setup

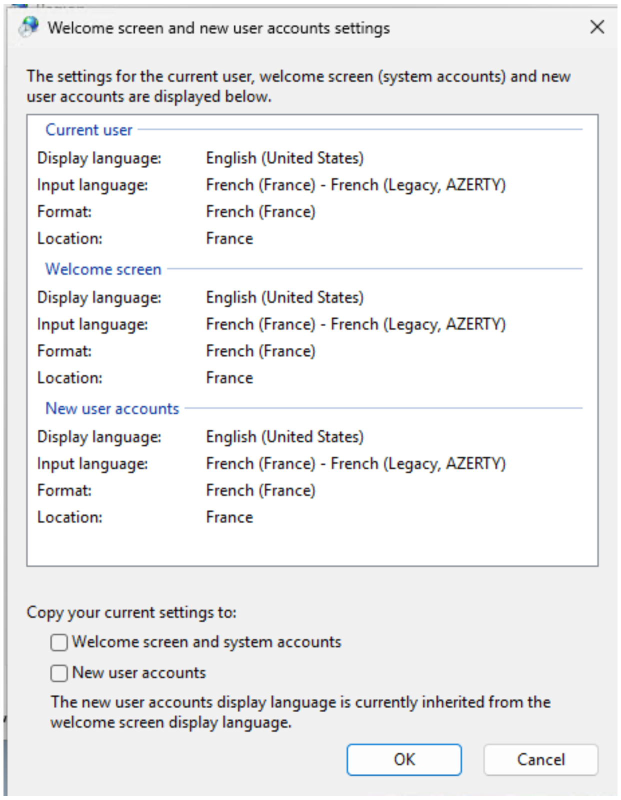
Select language settings

Language to install: English (United States)

Time and currency format: English (United States)

When the installation is in the WinPE pass, the system has booted to the boot.wim file. It is possible to inject the WinPE language package into boot.wim, but this would only be for installation and not the Windows OS being installed.

2. Since the French language package is not installed, the UI will appear in English, but all the other local settings are set to French (France). The picture below is from Control Panel-> Region->Copy Settings.



Welcome screen and new user accounts settings

The settings for the current user, welcome screen (system accounts) and new user accounts are displayed below.

Current user

Display language:	English (United States)
Input language:	French (France) - French (Legacy, AZERTY)
Format:	French (France)
Location:	France

Welcome screen

Display language:	English (United States)
Input language:	French (France) - French (Legacy, AZERTY)
Format:	French (France)
Location:	France

New user accounts

Display language:	English (United States)
Input language:	French (France) - French (Legacy, AZERTY)
Format:	French (France)
Location:	France

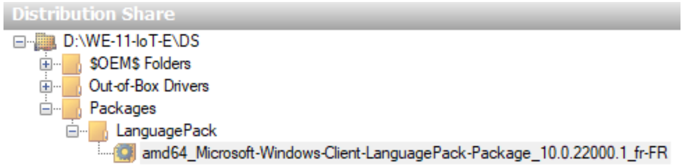
Copy your current settings to:

☐ Welcome screen and system accounts

☐ New user accounts

The new user accounts display language is currently inherited from the welcome screen display language.

If the language package is added to the distribution share and then added to the answer file, the installation will crash on the installation of the language package. I have never had the Package in the distribution share work correctly in a long time, so this is not a surprise.



Distribution Share
D:\WE-11-IoT-E\DS
OEM Folders
Out-of-Box Drivers
Packages
LanguagePack
amd64_Microsoft-Windows-Client-LanguagePack-Package_10.0.22000.1_fr-FR

I only have access to the Windows 10 IoT Enterprise LTSC 2024 OPK ISO English release, so I don't know if there are releases for other languages. For the Windows Embedded releases of the past, English was the base language for all installations. Some developers from other countries have raised awareness of this when trying to create a local-only installation. If there are releases in other languages, then the same issues apply when attempting to set up a different language via the Answer file.

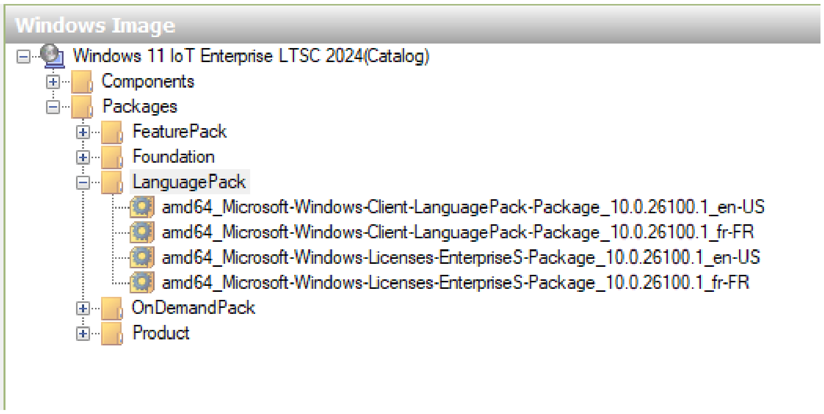
These first results looked at the basic issues with the answer file and the installation media out of the box. Now, we will turn to how to install the Language package.

1.6 Research Result 2: Injecting into the Install.wim – Think Long Term!

Per Addendum 1 to the book: Starter Guide for Windows 10 IoT Enterprise 2nd Edition, Addendum 1, it is possible to inject the .CAB file into the install.wim. From the addendum, here is a modified PowerShell script, where the language.cab files are placed in the c:\WinData\MediaRefresh\Packages\Lang folder:

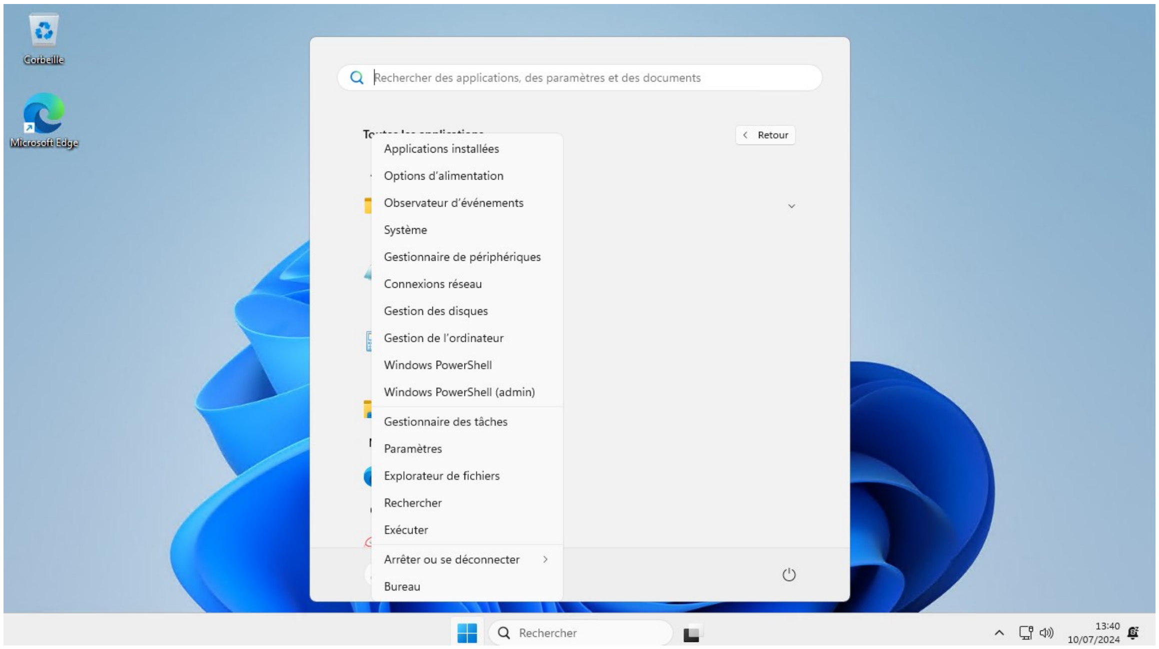
```
Set-ItemProperty -Path  c:\WinData\MediaRefresh\WIM\install.wim -Name IsReadOnly -Value
$false
Mount-WindowsImage  -ImagePath  c:\WinData\MediaRefresh\WIM\install.wim  -Index  2  -Path
c:\WinData\MediaRefresh\mounted
Add-WindowsPackage      -Path      c:\WinData\MediaRefresh\mounted       -PackagePath
c:\WinData\MediaRefresh\Packages\Lang
Dismount-WindowsImage -path c:\WinData\MediaRefresh\mounted -save
```

You can recreate the catalog file to see any changes to the install.wim file and the latest updates. We can see the newly added French (France) language package in the picture below.

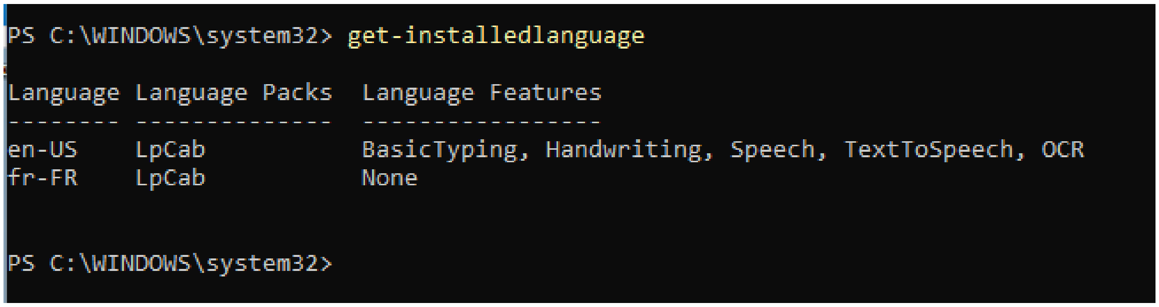


Windows Image
Windows 11 IoT Enterprise LTSC 2024(Catalog)
Components
Packages
FeaturePack
Foundation
LanguagePack
amd64_Microsoft-Windows-Client-LanguagePack-Package_10.0.26100.1_en-US
amd64_Microsoft-Windows-Client-LanguagePack-Package_10.0.26100.1_fr-FR
amd64_Microsoft-Windows-Licenses-EnterpriseS-Package_10.0.26100.1_en-US
amd64_Microsoft-Windows-Licenses-EnterpriseS-Package_10.0.26100.1_fr-FR
OnDemandPack
Product

In the answer file, fr-FR can be put into all the settings for amd64_Microsoft-Windows-International-Core__neutral. The installer will be in English, but the installed Windows OS will be in French.



Checking the installed languages via PowerShell, English (United States) and French (France) language packages are installed. The French doesn't include the other French language features. These other features are on the individual FODs. These can also be injected into the build but must come after the installation of the core language package. For Windows UI purposes, the core language package is all you need.



```
PS C:\WINDOWS\system32> get-installedlanguage

Language Language Packs  Language Features
-------- --------------  -----------------
en-US    LpCab           BasicTyping, Handwriting, Speech, TextToSpeech, OCR
fr-FR    LpCab           None


PS C:\WINDOWS\system32>
```

Warning: Per the Addendum 1, you can inject updates, language packages, and other supported cab files into the install.wim file. The problem is the long-term development support. One person is in charge of building the image, and they may do this for a few years. Suddenly, they leave or move on to a different company role. A new person comes along and has to pick up the project. The new person will have no idea what is in the wim file unless there is a process and documentation in place. In my experience, there never is a clean process or good documentation left for the new developer to follow. I have preferred to leave the Microsoft-provided install.wim alone and install features via the distribution share and the synchronous commands in the answer files. Working outside of the install.wim file provides a clean way to troubleshoot and quickly fix issues when they arise without having to question what is inside the wim file. Injecting Microsoft-provided updates and language packages has some advantages.

With that warning out of the way, Addendum 1 covered the method for adding Windows Updates to the install.wim file, documentation, and binary backup so there is a running record of the changes. As far as performing the update, all the folder setup steps mention in Addendum 1 apply. A font language folder needs to be provided. If you want to add specific FOD packages, then create an FOD folder as well. The PowerShell script can be updated with the inclusion of the new path to the languages. Here is an updated MediaRefresh.ps1 Script from Addendum 1 with the languages being added first followed by the updates.

```
#updates the install.wim with the latest cumulative update and install languages

Set-ItemProperty -Path  c:\WinData\MediaRefresh\WIM\install.wim -Name IsReadOnly -Value
$false
Mount-WindowsImage  -ImagePath  c:\WinData\MediaRefresh\WIM\install.wim  -Index  2  -Path
c:\WinData\MediaRefresh\mounted
Add-WindowsPackage       -Path       c:\WinData\MediaRefresh\mounted       -PackagePath
c:\WinData\MediaRefresh\Packages\Lang
Add-WindowsPackage       -Path       c:\WinData\MediaRefresh\mounted       -PackagePath
c:\WinData\MediaRefresh\Packages\SSU
Add-WindowsPackage       -Path       c:\WinData\MediaRefresh\mounted       -PackagePath
c:\WinData\MediaRefresh\Packages\LCU
Dismount-WindowsImage -path c:\WinData\MediaRefresh\mounted -save
Split-WindowsImage  -ImagePath  c:\WinData\MediaRefresh\WIM\install.wim  -SplitImagePath
c:\WinData\MediaRefresh\WIM\install.swm -FileSize 4000 -CheckIntegrity
```

Once the update has been completed, I recommend that you create a folder structure to store each install.wim, install.swm file, and catalog file for each update starting with the Windows installer ISO install.wim as the base. Always go back to the original install.wim when adding in new updates.

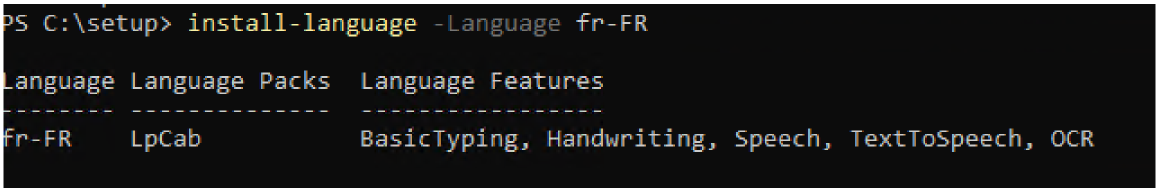
\Windows 10 LTSC 2021 - ISO
\Windows 10 LTSC 2021 – October 2023 – Language – French Japanese
\Windows 10 LTSC 2021 – November 2023 – Language – French Japanese

The folder structure provides a fallback for future developers and can assist with troubleshooting. The next section looks at the install.wim external approach to installing the language packages.

1.7 Research Result 3: Language Pack CAB File Install versus LP Download

Now, let’s look at the results for installing the language package without having to inject them into the install.wim. The installed image had the English language package only to start, so two methods can be tested to install a language package. The first method requires an Internet connection and uses the PowerShell Cmdlet to install a language package:

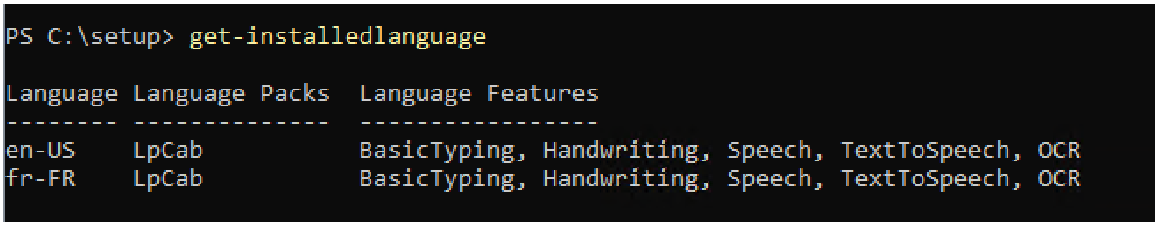
```
Install-language -Language fr-FR
```



```
PS C:\setup> install-language -Language fr-FR

Language Language Packs  Language Features
-------- --------------  -----------------
fr-FR    LpCab           BasicTyping, Handwriting, Speech, TextToSpeech, OCR
```

The process takes some time to download and install. The final result is the language package and the supported language features are installed into the image. Running the get-installedlanguage Cmdlet shows that English and French with all the Language Features are installed.



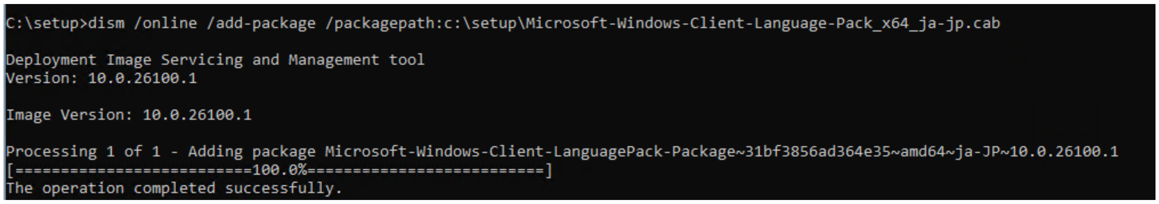
```
PS C:\setup> get-installedlanguage

Language Language Packs  Language Features
-------- --------------  -----------------
en-US    LpCab           BasicTyping, Handwriting, Speech, TextToSpeech, OCR
fr-FR    LpCab           BasicTyping, Handwriting, Speech, TextToSpeech, OCR
```

If you had to install all 38 language packages to support your device, the download and installation would take a few hours. If Windows updates are being installed in the background, the installation process could break. Rather than depending on an internet connection, the .CAB files from the

Language package DVD ISO can be installed directly. DISM was used to install the Japanese language package:

```
Dism /online /add-package /packagepath:c:\setup\Microsoft-Windows-Client-
Language-Pack_x64_ja-jp.cab
```



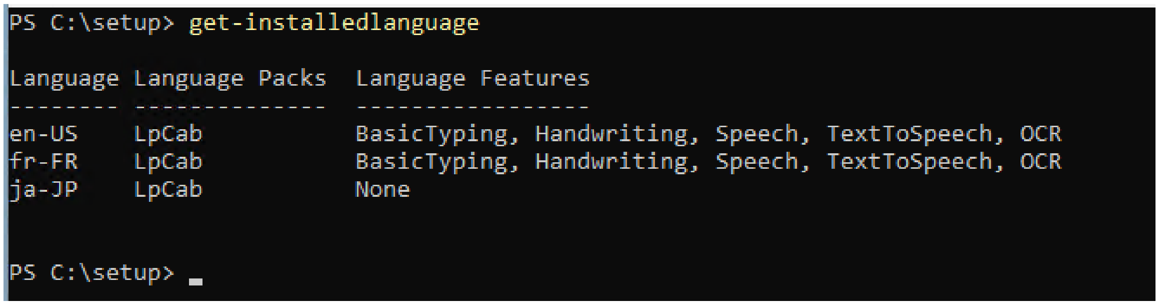
```
C:\setup>dism /online /add-package /packagepath:c:\setup\Microsoft-Windows-Client-Language-Pack_x64_ja-jp.cab

Deployment Image Servicing and Management tool
Version: 10.0.26100.1

Image Version: 10.0.26100.1

Processing 1 of 1 - Adding package Microsoft-Windows-Client-LanguagePack-Package~31bf3856ad364e35~amd64~ja-JP~10.0.26100.1
[==========================100.0%==========================]
The operation completed successfully.
```

After the package is installed, the get-installedlanguage Cmdlet shows the newly installed language package, but it doesn't have the other FOD language features installed. Just like injecting into the install.wim, the FOD language features can be installed after the core language package has been installed. For Windows UI purposes, the core language package is all you need.

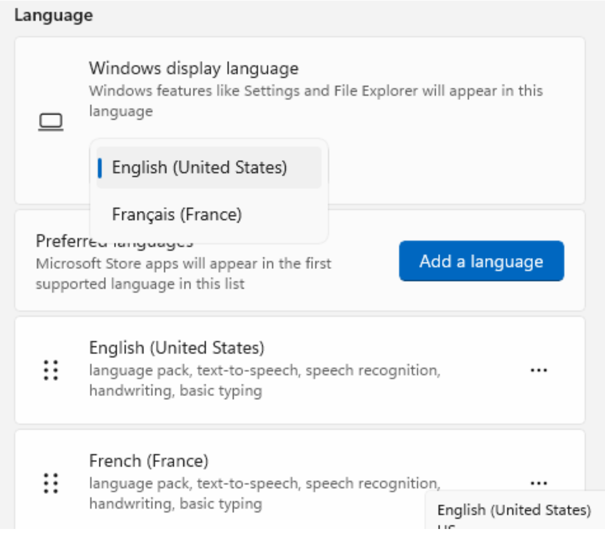


```
PS C:\setup> get-installedlanguage

Language Language Packs  Language Features
-------- --------------  -----------------
en-US    LpCab           BasicTyping, Handwriting, Speech, TextToSpeech, OCR
fr-FR    LpCab           BasicTyping, Handwriting, Speech, TextToSpeech, OCR
ja-JP    LpCab           None


PS C:\setup> _
```

There is a catch. Even though the language package has been installed, the language doesn't show up in the available language list. Windows 10 has this issue. Windows 11 might just require a reboot.



Language

Windows display language
Windows features like Settings and File Explorer will appear in this language

English (United States)
Français (France)

Preferred languages
Microsoft Store apps will appear in the first supported language in this list

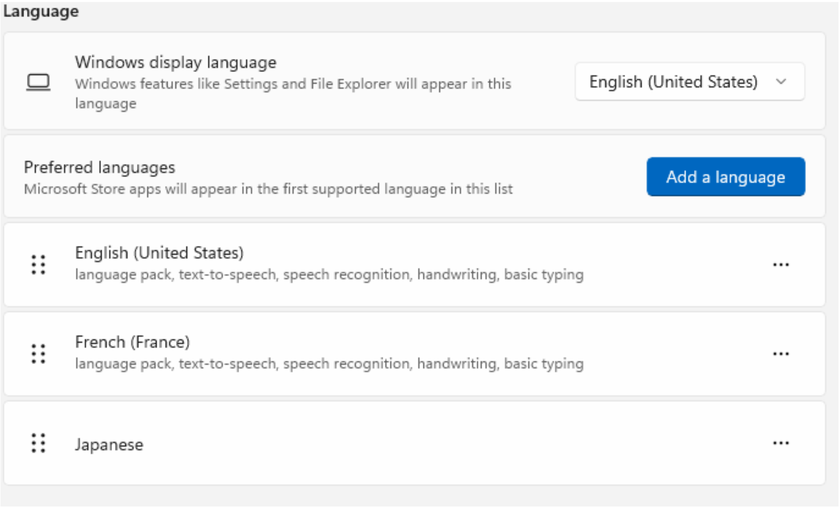
English (United States)
language pack, text-to-speech, speech recognition, handwriting, basic typing

French (France)
language pack, text-to-speech, speech recognition, handwriting, basic typing

In either case, to correct this problem a simple PowerShell script can be run to add the language to the list.

```
$LangList = Get-WinUserLanguageList
$LangList.Add("ja-jp")
Set-WinUserLanguageList -LanguageList $LangList -Force
```

After running the script, Japanese shows up as one of the available languages. If multiple languages were installed, the script could be expanded with multiple $LanList.Add lines before the final SetWinUserLanguageList is called.



Language

Windows display language
Windows features like Settings and File Explorer will appear in this language
English (United States)

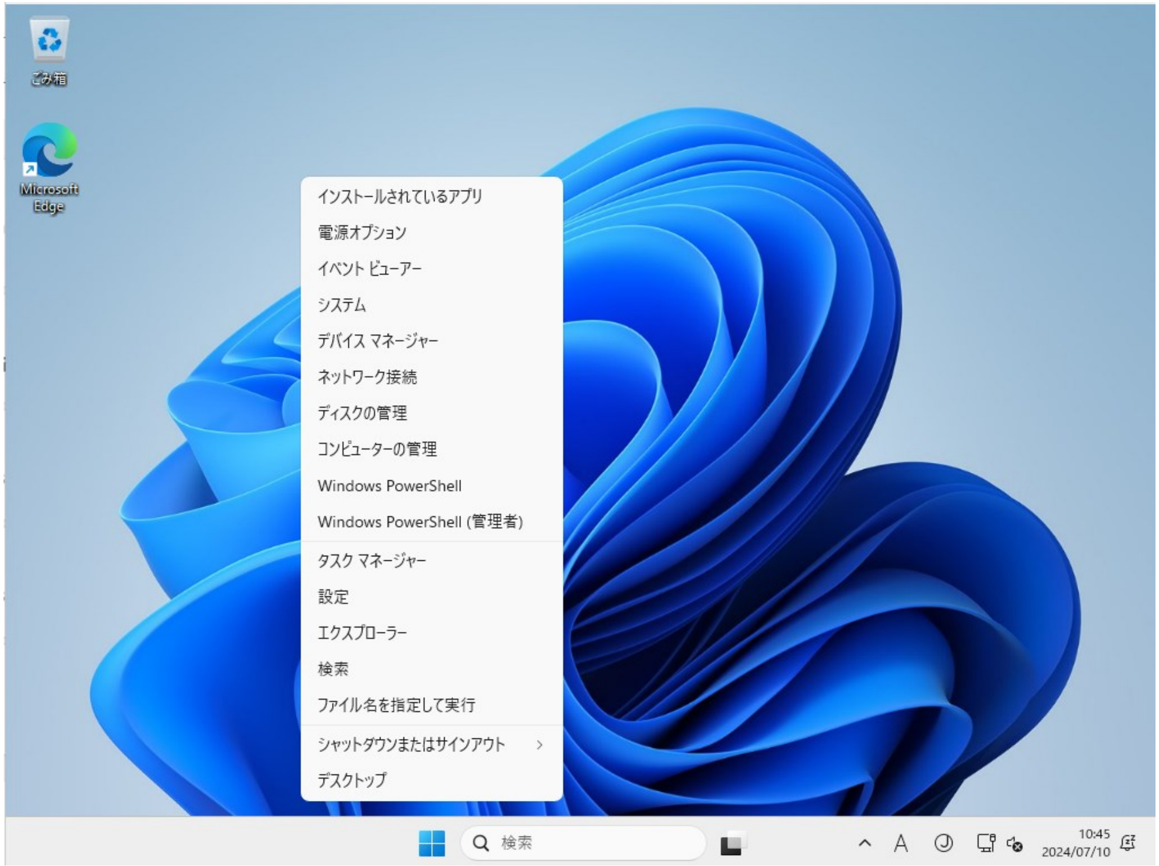
Preferred languages
Microsoft Store apps will appear in the first supported language in this list
Add a language

English (United States)
language pack, text-to-speech, speech recognition, handwriting, basic typing

French (France)
language pack, text-to-speech, speech recognition, handwriting, basic typing

Japanese

Switching to Japanese works successfully.

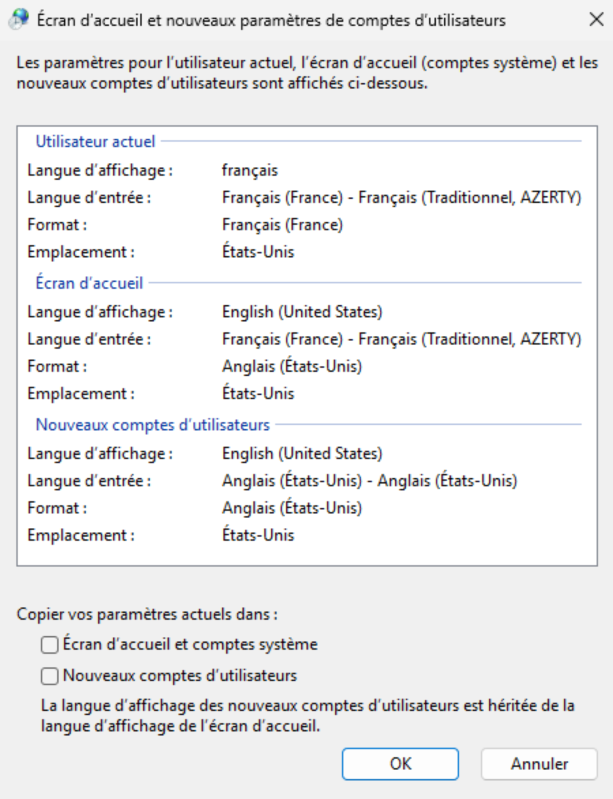


A Pass 7 synchronous command could call a PowerShell script to install the language package .CAB file, any language FOD .CAB files, and then update the language list. Installing the .CAB file requires a little more work but achieves the same goal for getting the language packages installed without the Internet connection or injecting into the install.wim file. The external approach allows for quick troubleshooting if something doesn’t work correctly, but there is a little more work to do to get the system set up.

1.8 Research Result 4: Multiple User Accounts

Multiple user accounts introduce an issue. Here is the scenario:

Two user accounts are put into the answer file, User1 and User2. The goal is to have the French UI for all accounts. The system is set up in the User1 account where the original language was English, but the French language package gets installed. User1 is then switched to the French UI. Logging into User2, the system comes up in English. The UI language selection is user-dependent. Looking at the User1 Regional Language settings, you can see that the Welcome Screen and New Users are going to be in English.



Écran d’accueil et nouveaux paramètres de comptes d’utilisateurs

Les paramètres pour l’utilisateur actuel, l’écran d’accueil (comptes système) et les nouveaux comptes d’utilisateurs sont affichés ci-dessous.

Utilisateur actuel

Langue d’affichage : français
Langue d’entrée : Français (France) - Français (Traditionnel, AZERTY)
Format : Français (France)
Emplacement : États-Unis

Écran d’accueil

Langue d’affichage : English (United States)
Langue d’entrée : Français (France) - Français (Traditionnel, AZERTY)
Format : Anglais (États-Unis)
Emplacement : États-Unis

Nouveaux comptes d’utilisateurs

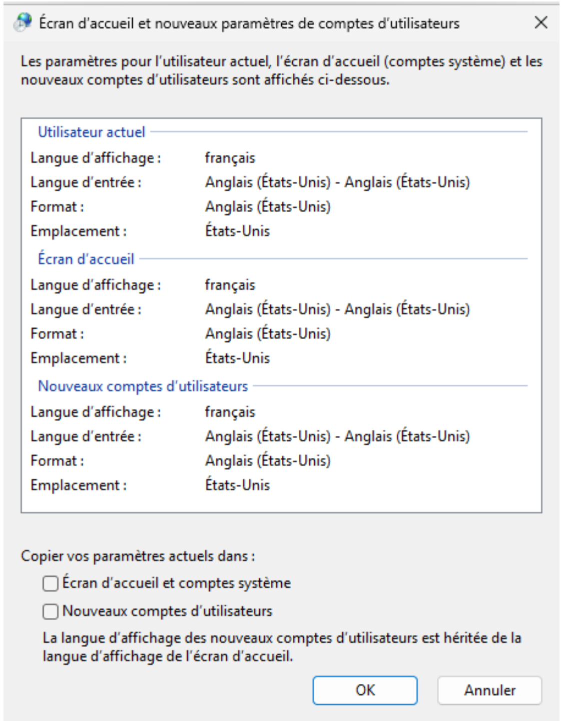
Langue d’affichage : English (United States)
Langue d’entrée : Anglais (États-Unis) - Anglais (États-Unis)
Format : Anglais (États-Unis)
Emplacement : États-Unis

Copier vos paramètres actuels dans :

☐ Écran d’accueil et comptes système
☐ Nouveaux comptes d’utilisateurs

La langue d’affichage des nouveaux comptes d’utilisateurs est héritée de la langue d’affichage de l’écran d’accueil.

If you run the Set-SystemPreferredUILanguage fr-FR Cmdlet, this will change the system UI so that the Welcome screen and new user accounts will have the French UI.



Écran d'accueil et nouveaux paramètres de comptes d'utilisateurs

Les paramètres pour l'utilisateur actuel, l'écran d'accueil (comptes système) et les nouveaux comptes d'utilisateurs sont affichés ci-dessous.

Utilisateur actuel

Langue d'affichage :	français
Langue d'entrée :	Anglais (États-Unis) - Anglais (États-Unis)
Format :	Anglais (États-Unis)
Emplacement :	États-Unis

Écran d'accueil

Langue d'affichage :	français
Langue d'entrée :	Anglais (États-Unis) - Anglais (États-Unis)
Format :	Anglais (États-Unis)
Emplacement :	États-Unis

Nouveaux comptes d'utilisateurs

Langue d'affichage :	français
Langue d'entrée :	Anglais (États-Unis) - Anglais (États-Unis)
Format :	Anglais (États-Unis)
Emplacement :	États-Unis

Copier vos paramètres actuels dans :

☐ Écran d'accueil et comptes système

☐ Nouveaux comptes d'utilisateurs

La langue d'affichage des nouveaux comptes d'utilisateurs est héritée de la langue d'affichage de l'écran d'accueil.

There are other PowerShell Cmdlets that can be used to set the rest of the location, format and input to Fench. When adding languages via the distribution share rather than injecting them into the install.wim file, there is a little more work to do. If you are building an image to support multiple languages, allow the end user to select the local language, and have multiple user accounts. Then a PowerShell script needs to be called with a pass 7 sync command in the sysprep unattended file, so that all other accounts get the correct language.

```
Set-SystemPreferredUILanguage $PSUICulture
Set-Culture $PSUICulture
Set-WinSystemLocale -SystemLocale $PSUICulture
Set-WinUserLanguageList $PSUICulture -Force
$geoID = [System.Globalization.RegionInfo]::new((Get-UICulture).Name).GeoId
Set-WinHomeLocation -GeoId $geoID

```

1.9 Summary

Most of the time OEMs create their main application to support multiple languages. As the main application is the shell of the system, Windows-specific language support is not needed. There are solutions that allow a technician to access the system via the Explorer shell, and allowing the UI to be in the native language makes the system easier to support locally. This addendum looked at the limitations of the Windows IoT Enterprise ISO and how to get language packages into the image. The choice of injecting the language package .CAB file into the install.wim file or installing a language package .CAB file after OS installation is up to the developer, but it should be well documented for future product life cycle changes.

1.10 References

There were a number of important references that were helpful in the creation of this addendum:

[Available Language Packs for Windows | Microsoft Learn]()

[List of ISO 639 language codes - Wikipedia]()

Language Packs, Language Experience Packs, Language Interface Packs… what?!- SysManSquad | Systems Management Squad

Command line tools to completely change region/input language for default user and welcome screen | Microsoft Learn

How to change language/region and speech in Windows 10 with Powershell script - Stack Overflow