WIM (Windows Image Management) Tools – User Guide

March 20, 2022

General Information

I first created this program as an aid to injecting Windows updates into my Windows ISO images. When that task was completed, I wasn't satisfied with only being able to update one edition of Windows in an image at a time, so I added the ability to allow the updating of multiple editions at once. Along the way, I kept realizing that there were other things that I would like to be able to do and so I added those capabilities one by one. Along the way, I not only added new features, but I constantly refined the existing capabilities. The final result of what you have before you now is the ***WIM Tools*** program.

Here are just some of the tasks that can be performed with this program:

- Inject Windows updates into one or more editions of Windows and combine them all into a single ISO image. The Windows editions can come from multiple different ISO images and can even be a mix of x86 and x64 images. Updates can include all the standard Windows updates as well as dynamic updates, Servicing Stack Updates, generic files to add to the WinPE image, and more.

- Just as with Windows updates, drivers can be injected into one or more editions of Windows and combined into a single ISO image.

- Inject boot-critical drivers into the Windows PE image of Windows media.

- All 3 of the above items can be scripted. The program can automatically generate a script for you and play back a script. Since scripts are commented, it's easy to manually modify or "tweak" these scripts.

- Create bootable media such as HDs and flash drives that can include both x86 and x64 editions of Windows. This media can be booted on both BIOS and UEFI based systems.

- When creating bootable media, the program can create additional partitions for other data as well. It can even BitLocker encrypt the additional partitions if desired.

- When creating bootable media, the program can update the Windows boot partition using updated images without affecting any additional partitions created on the same media.

- The program can create a single bootable image from multiple different images without having to inject updates or drivers.

- A bootable image can be created from files in a folder.

- The contents of an ISO image can be reorganized.

Several tools for related tasks are available. These utilities include:

- Export all drivers from a system so that they can be added to Windows images.
- Expand drivers supplied in .CAB files so that they can be injected into Windows images.
- Create virtual disks to make working with images easier.
- Create a virtual disk, deploy Windows to it, and then add it to the boot menu of a computer to create a dual boot configuration with a bootable virtual disk.
- Create a generic ISO image with any data desired. This is especially handy for getting data to a VM easily since the ISO image can simply be mounted as a CD-ROM drive.
- Display and modify WIM metadata.
- The program includes built-in help which contains most of the information found in this user guide.

Program Editions

There are 2 editions of this program:

- Dual Architecture Edition
- X64 Only Edition

Both editions will work with either Windows 10 or Windows 11, but only the Dual Architecture Edition will work with x86 editions of Windows 10 or a combination of both x64 and x86 Windows editions.

If you want to work with x86 editions of Windows 10, or if you want to create images and bootable media that contain both x64 and x86 editions of Windows 10, then you will want to use the Dual Architecture Edition of this program.

The x64 Only Edition of this program is a little smaller in size because it does not include the code needed to work with x86 editions of Windows.

Note that Windows 11 is available only in x64 flavors – no x86 versions exist. As a result, if you plan to work only with Windows 11, you can use the x64 Only Edition of WIM Tools.

As you read this user guide, please just keep in mind that any operations related to dual architecture images do not apply to the x64 Only Edition of this program.

IMPORTANT: The version of the program available on GitHub is the Dual architecture Edition so contains the full complement of features. In the future, once Windows 10 is phased out and the need to support x86 editions of Windows is no longer needed, we will phase out the Dual Architecture Edition and switch over to the x64 Only Edition. We do not anticipate doing this until at least 2026.

System Requirements

This program is a 64-bit program designed to work only on 64-bit systems. It is also designed to work with Windows ISO images that contain an INSTALL.WIM file in the \sources folder, not an INSTALL.ESD. There is one exception which will be discussed in sections below to which it applies. Please also note that there is a routine for converting images with an install.esd file(s) into one with an install.wim file(s).

Note that while the program is designed to run on x64 hardware, the Dual Architecture Edition of this program can work with both x64 and x86 editions of Windows images although it should be run on an x64 edition of Windows.

This program requires the Windows ADK to be installed. Only the "Deployment Tools" component needs to be installed. The program will display a warning when it is started if the ADK is not installed. However, it will continue to operate since some functions will work without the ADK. If the user selects a feature from the menu that requires the ADK, the user will be warned and returned to the main menu.

You can download or install the Windows ADK from here:

Download and install the Windows ADK | Microsoft Docs

Run the program locally, not from a network location. I have not tested the program or designed it to run from a network location.

When operating on multiple editions of Windows in the same project (for example, Win 10 Pro, Home, Education editions, etc.), this program is designed to work with editions of the same version. For example, you do not want to mix version 20H2 and 21H1 in the same project. In addition, you should ensure that all editions of Windows in your image are of the same build number.

Disable QuickEdit Mode

The color of text in some places within the program may be displayed incorrectly if QuickEdit mode is enabled. If you encounter such difficulties, try disabling QuickEdit mode by following these steps:

Right-click on the title bar when the program is running and choose Properties, select the Options tab, uncheck QuickEdit Mode, click OK, restart the program. You should only need to do this once as this setting will be retained for the next time you run the program.

Memory and Storage Requirements

The program uses minimal amounts of memory. If your system meets the minimum memory requirements to run Windows, it should be able to run WIM Tools.

The storage space needed will depend upon the size of the image files that you are working with. You will need enough space to hold all the Windows images that you wish to work with, and as a general rule of thumb, at least three times the amount of space that your final image will occupy plus space for your Windows updates and drivers that will be injected into your project. It is not necessary to store all of these items on the same drive.

General Usage Tips

Terminology – Images, Editions, and Indices

Image - The Windows files are distributed by Microsoft on physical media such as a DVD or flash drive or as an "image" file that can be downloaded and used to make physical media. These images are also known as "ISO image" to describe the precise type of image format used. These images hold a collection of files in some ways like a ZIP file. These images, in turn, hold other images, known as WIM or **W**indows **IM**age files.

Edition - These image files hold information regarding one or more flavors of Windows such as Professional, Home, Education, etc. These various flavors of windows are also known as "editions".

Index - Each edition of Windows has an "index" number associated with it. For example, index number 1 may be associated with the Home Edition of Windows, Index 2 may be the Windows Professional edition of Windows, etc. The index numbers will always start with "1" increase sequentially (1, 2, 3, 4, etc.). Windows utilities work with the index number that is associated with each edition of Windows. In those places within the program where an index number(s) is needed, the program will give you an opportunity to get a list of indices and the edition of Windows associated with each index from any image files that you are working with.

Hard Disk vs. Removable Media

For the process of injecting Windows updates or drivers, one of the Microsoft utilities used (DISM) requires the use of a fixed disk, not a removable disk. Make sure that you use a fixed disk to store your projects. If you have no choice but to use removable media, you can work around this by creating a virtual hard disk on a removable disk and then using that virtual hard disk to store your project. There is an option on the main menu to help you create a virtual disk should you need to do so.

Responding to the Program

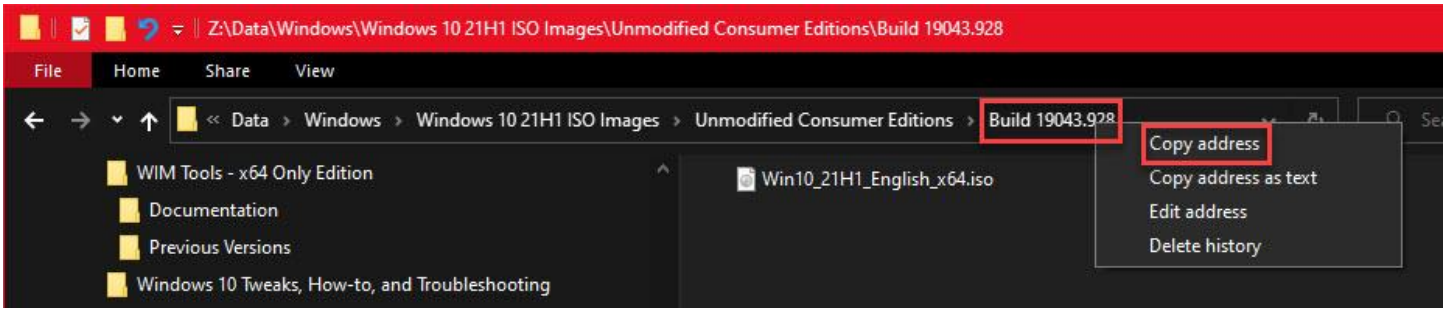
When the program asks for a path, you can enclose paths that have spaces in them in quotes if you wish, but this is not necessary. The program will handle paths either way.

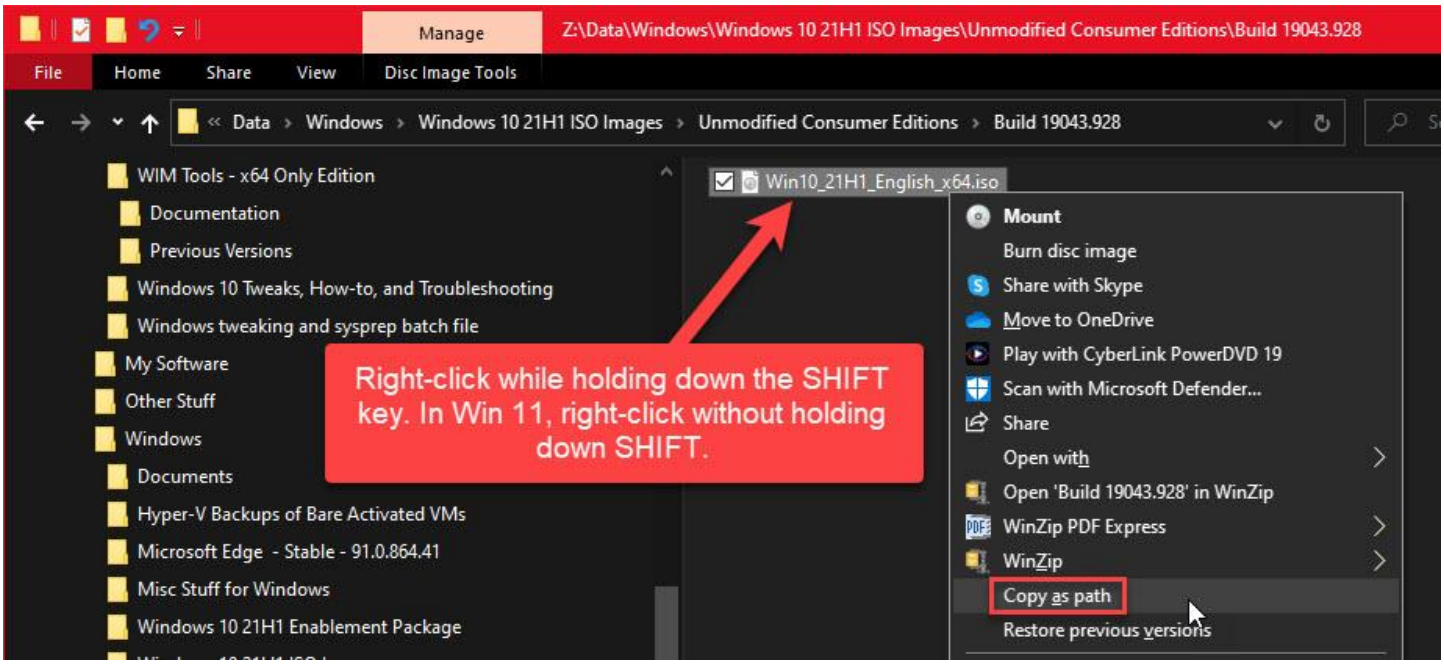
The program will allow you to paste into it, which can be especially helpful for long paths. To copy and paste a path do the following:

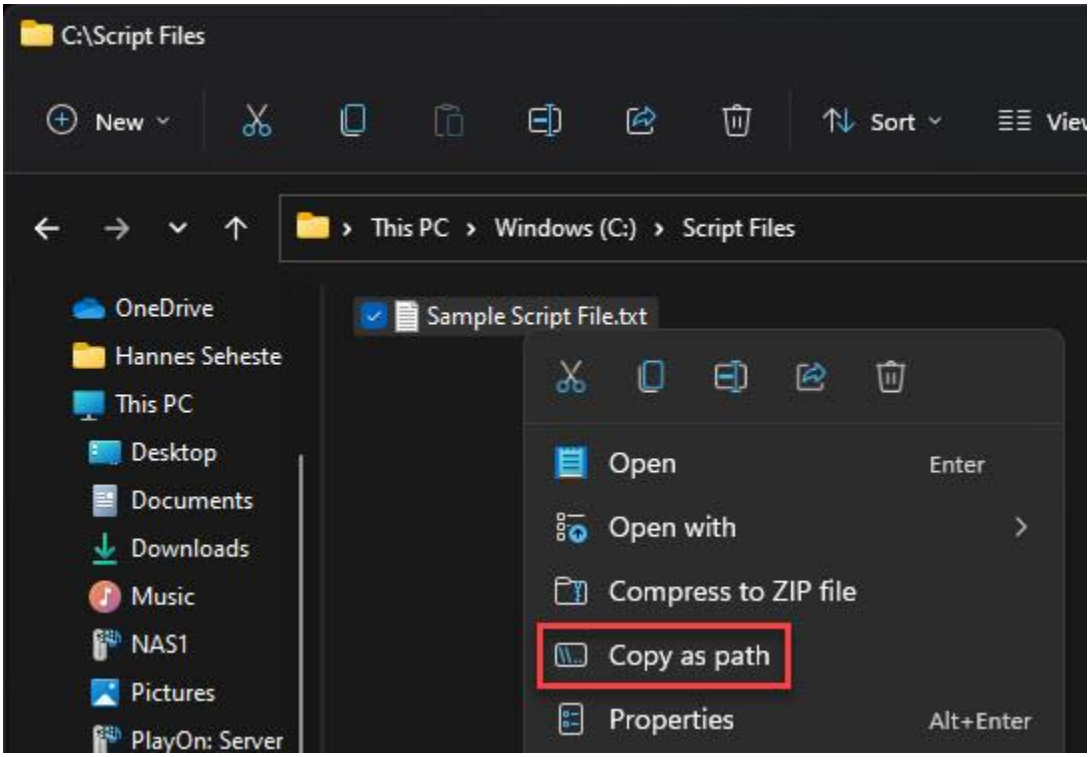
- Locate the path in File Explorer.
- With the folder of interest open in File Explorer, right-click in the Address Bar.
- Select “Copy address”.
- In the WIM Tools program, right-click to paste the address.
- For filenames, right-click on the file in the right pane of File Explorer while holding the SHIFT key select "Copy as path".

NOTE: In Windows 11, there is no need to hold down the shift key to get the "Copy as path" option.

Below are some samples of what this looks like in Windows 10 and 11.


Z:\Data\Windows\Windows 10 21H1 ISO Images\Unmodified Consumer Editions\Build 19043.928
File
Home
Share
View
Data
Windows
Windows 10 21H1 ISO Images
Unmodified Consumer Editions
Build 19043.928
WIM Tools - x64 Only Edition
Documentation
Previous Versions
Windows 10 Tweaks, How-to, and Troubleshooting
Win10_21H1_English_x64.iso
Copy address
Copy address as text
Edit address
Delete history


Manage
Z:\Data\Windows\Windows 10 21H1 ISO Images\Unmodified Consumer Editions\Build 19043.928
File
Home
Share
View
Disc Image Tools
Data
Windows
Windows 10 21H1 ISO Images
Unmodified Consumer Editions
Build 19043.928
WIM Tools - x64 Only Edition
Documentation
Previous Versions
Windows 10 Tweaks, How-to, and Troubleshooting
Windows tweaking and sysprep batch file
My Software
Other Stuff
Windows
Documents
Hyper-V Backups of Bare Activated VMs
Microsoft Edge - Stable - 91.0.864.41
Misc Stuff for Windows
Windows 10 21H1 Enablement Package
Win10_21H1_English_x64.iso
Right-click while holding down the SHIFT key. In Win 11, right-click without holding down SHIFT.
Mount
Burn disc image
Share with Skype
Move to OneDrive
Play with CyberLink PowerDVD 19
Scan with Microsoft Defender...
Share
Open with
Open 'Build 19043.928' in WinZip
WinZip PDF Express
WinZip
Copy as path
Restore previous versions


C:\Script Files
New
Sort
This PC
Windows (C:)
Script Files
OneDrive
Hannes Seheste
This PC
Desktop
Documents
Downloads
Music
NAS1
Pictures
PlayOn: Server
Sample Script File.txt
Open
Enter
Open with
Compress to ZIP file
Copy as path
Properties
Alt+Enter

Yes or No Responses – Respond with anything that starts with a “Y” or “N” in either uppercase, lowercase, or mixed case. Similarly, responses to questions with a limited number of responses can generally be responded to with the first letter of the item you wish to select. As an example, when you asked if you want to Record a script, Play a script, Skip scripting operations, or display Help, any response starting with the letters R, P, S, or H would be seen as valid responses.

Scripting – The program maintains a keyboard buffer, so you could paste the response to a series of input requests into the program at once, however, there is a much better way: scripting. For the routines that inject Windows updates, drivers, or boot-critical drivers, the program can record and playback a script. For example, suppose that you choose the option to inject Windows updates. When you select this option from the menu, the program will ask you if you want to record or play back a script. If you want to record a script, your responses will be recorded. When all the information needed is captured, a script file will be saved. The program will offer to save a script before carrying out any actions, so you can create a script without actually perform the scripted actions.

Note that scripts have comments. This makes it easy to understand what each response relates to and allows for easy manual modification of the script should you choose to do so.

The script will always be saved to the same folder from which the program was run as "WIM_SCRIPT.TXT". If you have a previous script that you want to keep, rename it, or move it.

Note that since the program does maintain a keyboard buffer, you should avoid typing random characters while the program is running and has focus as these keystrokes will be stored in the buffer and played back when the program seeks input.

Be careful using scripting, especially when manually modifying scripts! If the contents of a folder change after you have created a script, that script may no longer be usable as is.

A new feature of the program is the option to unambiguously specify file names to be updated rather than a folder containing images. By using the full path with file name in your responses, the script will be more reliable. This is because when a folder is chosen, your responses change depending upon how many files are in the folder. If that changes, your responses won't work when played back. When not scripting, it's easier to point to a folder, but for scripting, it's more reliable to take the time to specify each file to update. This will also make manually modifying the script responses easier.

Antivirus Exclusion

When the program is started, it will set an exclusion for itself in Windows Defender Antivirus. In addition, for the projects that inject files, drivers, or boot-critical drivers, the program will set a Windows Defender Antivirus exclusion for the destination folder to improve performance.

When choosing “Exit” from the Main Menu, exclusions will be removed. Note that if the program ends without choosing “Exit” from the main menu, the exclusions will remain in place. To remove the exclusion, simply start the program and choose “Exit” from the menu. The exclusions will be removed.

To manually check or manage exclusions, press Windows_Key + R and type in “**ms-settings:windowsdefender**”. Press ENTER. Go to **Virus & threat protection**. Under the section "**Virus & threat protection settings**" choose "**Manage settings**" > **Add or remove exclusions**.

If you use other AV software, you can set exclusions manually if you wish.

Entering Index Numbers

Some Windows ISO images may have multiple editions of Windows. Each edition has a unique index number associated with it.

As an example, suppose an image file that has the following Windows editions on it:

Edition	Index Number
Home	1
Professional	2
Workstation	3

Some of the routines in WIM Tools will ask for the index number(s) to work with. These routines will have an option to allow you to display all the available editions of Windows as well as the indices associated with each edition.

Index numbers can be entered into the program as shown in the examples below:

- A single index number.
Example: 6

- Multiple index numbers: Separate indices with a space.
Example: 1 6 9

- A range of indices: Specify a range of indices by separating the lower and upper range with a dash.
Example: 1-3

- A combination of the above: Separate each group with a space.
Example: 1 4-6 9

- The word “ALL”.
This will automatically determine all available indices and select them.

IMPORTANT: Enter the index numbers from low to high. Don't specify a lower index number after a higher index number. The exception to this is the routine for reorganizing the contents of a Windows ISO image. Since the goal of this option is to reorder the Windows editions, it may be necessary to specify indices in any order.

Reviewing Log Files - See Appendix B for information regarding this topic. It is suggested that you review the "SANITIZED_ERROR_SUMMARY.log" after routines that update Windows images.

Auto Shutdown and Pause Program Execution

For the routines that inject Windows updates, drivers, or boot-critical drivers, you can choose to have the program automatically shut down the system when it is done running and you can pause the execution of the program to free up resources to other programs.

To have the program perform an automatic shutdown, create a file on your desktop named “Auto_Shutdown.txt”. If a file by that name exists, a shutdown will be performed when those routines have finished. To pause program execution, create a file on your desktop named "WIM_Pause.txt".

Note that for auto shutdown, you can change your mind at any time. The existence of this file will only cause a shutdown when the routine finishes running so you are free to create that file at any time even while the program is running, or you can delete / rename the file if you decide at some point that you do not want an automatic shutdown.

While the program is running, you will see a status indication on the upper right of the screen to remind you whether an automatic shutdown will occur or not. If program execution is paused, a flashing message will be displayed as a reminder that the program is paused. Note that when you make a change, the status will not update immediately. The status is updated each time the program advances to the next item in the displayed checklist. However, when the program is resumed by deleting or renaming the "WIM_Pause.txt" file, this status change will be reflected immediately.

Note that when the system is shut down automatically, any status messages or warning that would normally be displayed will not be shown due to the shutdown. Instead, this information is logged to a file. The next time the program is run that information will be automatically displayed. After viewing this information, that file will be automatically deleted.

Handling of autounattend.xml Answer Files

Including an answer file in your project can cause the system or VM booted from an image or bootable media to begin a fully automated installation of Windows that can result in one or more disks being wiped without warning. As a result, all routines in this program that can compile multiple Windows editions into a single image or bootable media will exclude the answer file from the original sources. One reason for this is that it is possible to have different answer files on different sources and as a result the final copied answer file could potentially be from any of these sources, possibly an unintended answer file. There are 2 exceptions to this:

1) For the routine to make or update a bootable drive from a Windows ISO image, we will ask the user if they want to exclude an answer file if one exists.

2) For the routine that creates a bootable ISO image from files in a folder we will INCLUDE the answer file if it is present since the user can simply delete it from the folder if they do not want it to be copied.

Note that for the routine that injects Windows updates, it will exclude answer files from the original sources, but if you include an answer file in the “Answer_File” subfolder of the folder that you specify as the Windows updates location, this answer file will be copied. See the information on the next page regarding the layout of Windows updates for more information regarding this.

Functions of This Program

The sections that follow describe each of the tools that this program provides.

Inject Windows updates into one or more Windows ISO images and create a multi edition, bootable ISO from the updated images

This routine will allow the user to pick Windows editions, inject Windows updates into them, and then create a single ISO image containing all these Windows editions. One or more Windows editions from an ISO image can be selected, as well as Windows editions from multiple different ISO images. A mix of both x86 and x64 images (in the Dual Architecture Edition of this program) can be used in the same project. Even sysprep images can be added.

This process will **NOT** inject driver updates. This routine only injects Windows updates. There is a separate option from the main menu that will inject drivers, and this is discussed later.

Organize Windows Updates

The program will expect to find Windows updates in a particular folder structure that looks like the one shown below. Begin by creating a folder and naming it anything you like. Example: Windows Updates. Then create the folder structure shown below:

```
\Answer_File          < Place autounattend.xml answer file here if you want to include one
\x64
    \LCU              < Place the LCU (Latest Cumulative Update) in this folder
    \MicroCode        < Keep CPU microcode updates here
    \Other            < Other updates such as .NET update or Flash Player update
    \PE_Files         < Used to copy generic files (not Windows updates) to the Win PE image
    \SafeOS_DU        < Safe OS Dynamic Update - Used to update Windows Recovery (WinRE)
    \Setup_DU         < Setup Dynamic Update
\x86
    \LCU
    \MicroCode
    \Other
    \PE_Files
    \SafeOS_DU
    \Setup_DU
```

NOTES: An autounattend.xml file will not be included from your original Windows image file(s). If you want to include one in the final image, place it into the Answer_File folder.

If you are working with only x64 editions of Windows, then an x86 folder is not needed but it won't hurt if it is present.

NOTES REGARDING FOLDER STRUCTURE AND OBTAINING UPDATES

Go to the Microsoft Update Catalog (https://www.catalog.update.microsoft.com/) to obtain updates. Search for "Windows 10 Version 21H1" (or whatever your version is) and then click on the "Last Updated" column to sort with the latest updates first.

For the MicroCode updates (for Windows 20H2), search for " KB4589212". Other versions of Windows will have different KB numbers associated with them. You may need search for these using yur favorite search engine.

Make sure download the correct updates (either x64 or x86).

IMPORTANT: When downloading updates from the Microsoft Update Catalog, you will see that some updates are described as “dynamic” updates. The Safe OS Dynamic Update and the Setup Dynamic Update are available only as dynamic updates. However, what may be confusing is that the Cumulative Update for Windows may be available as both a standard update AND a dynamic update. For the purposes of this program, DO NOT use the “dynamic” version of the cumulative update. Instead, download the version that is NOT described as a dynamic update.

Description of each folder in the structure:

LCU - Installs the latest cumulative quality update. The "Title" field in the update catalog will indicate will show "Cumulative Update for Windows 10" (or Windows 11) for this update. Save only one LCU file in this folder. Please note that the LCU is now combined with the **SSU**. The SSU or Servicing Stack Update, contains fixes that are necessary to address the Windows servicing stack and is required to complete a feature update. It is no longer necessary to download a separate SSU.

MicroCode - Unlike the other folders, the program will take no action on the contents of this folder. This folder is merely a storeage place to keep the latest MicroCode update file. We do this because the MicroCode updates do not apply to every CPU so this is a good place to keep it if it will not be used. If you want the program to update your Windows editions with the MicroCode up, simply copy the MicroCode update file from the \x64\MicroCode or \x86\MicroCode folder to the \x64\Other or \x86\Other folder before running the program.

Other - Updates that do not fall into the category of any other folder here are placed into the \Other folder. These typically include .NET updates and Adobe Flash Player updates (prior to 2021 when Abobe Flash Player will no longer be updated). You can place multiple files in this folder, but you should save only the latest version of each update type. For example, save only one .NET update here.

PE_Files - No Microsoft update files are placed into this folder. Instead, you will place generic files that need to be accessible to Windows PE during Windows setup in this folder. This would typically include things like scripts. Simply place any such files in this folder. If you later run Windows setup from media created these files, you will find the files that you placed here on X:\ which is the RAM Drive that Windows setup creates during installation. To delete a file from the WinPE image, create a dummy file with the same name as the file that you want to delete preceded with a minus sign (-). For example, to delete a file called "MyScript.bat", create any file in the \PE_Files folder, then rename the file to "-MyScript.bat". The case of the filename does not matter. Note that most people will never need to use this folder.

SafeOS_DU - Fixes for the "safe OS" that are used to update Windows recovery environment (WinRE). Save your Safe OS Dynamic Update file here. When downloading files from the Microsoft Update Catalog, the Safe OS Dynamic Update will specifically indicate "Safe OS Dynamic Update" in the "Title" field.

Setup_DU - This contains fixes to Setup binaries or any files that Setup uses for feature updates. Note that when downloading files from the Microsoft Update Catalog, the Setup Dynamic Update will indicate "Windows 10 Dynamic Update" in the "Product" field and "Dynamic Update for Windows 10" (or Windows 11) in the "Title" field. Store only one Setup Dynamic Update file in this folder.

TIP: If you have a new update that you wish the apply to your Windows edition(s) and you already have other updates applied to your Windows edition(s), create an update folder with only the new update(s) and remove the contents of all the other folders. This will cause the updates to be applied faster since the other updates don't have to be parsed to see if their contents have already been applied. For example, if you download a new LCU (Latest Cumulative Update) and you have already previously applied the other available updates such as the "Other updates" category, etc., then you can create a folder with only an LCU subfolder and the LCU placed in that folder. Note that this is not mandatory – it is perfectly fine to have updates already applied in your updates folder, it will simply take longer.

Notes Regarding the SSU and LCU: Microsoft now distributes the SSU and LCU together as a combined package in the LCU download. The program will apply this update at the appropriate time to install the LCU and / or the SSU if one is included in the package.

Acceptable Images

Note that all Windows ISO images used need to have an INSTALL.WIM (not an INSTALL.ESD). The one exception to this is that the file used to build the base image for a multiarchitecture project (one that contains both x64 and x86 images), can have INSTALL.ESD files on it.

What is a base image? When injecting Windows updates or drivers into a Windows image, the INSTALL.WIM file is modified. The other files in the image remain unchanged. As a result, to build the new image we simply copy all the files from the original ISO image to create a base image, and then we replace the standard INSTALL.WIM file with the new updated file.

When creating a base image for a dual architecture project with both x86 and x64 editions of Windows, neither an x86 nor x64 image contain all the files that we need to create the base image. If any Windows editions contained in a dual architecture image are being used, the program will detect this and automatically use one of the dual architecture images to create the base image. However, if no dual architecture image is being used in your project, the program will prompt for the location of either a dual architecture image or a “Windows Boot Foundation” image.

To create a dual architecture image, you will need one of the following:

Option 1: Download a dual architecture image from the ***Windows Media Creation Tool*** Web Site here:

https://www.microsoft.com/en-us/software-download/windows10

- Select "Download tool now" and run it.
- When the tool runs, accept the license terms.
- Choose "Create installation media (USB flash drive, DVD, or ISO file) for another PC" and then click on "Next".
- Uncheck the checkbox for "Use the recommended options for this PC".
- Select the appropriate language.
- For "Architecture" make certain to select "Both".
- Click on "Next".
- Select "ISO file" and then "Next".
- Save the downloaded image.

Option 2: To create a Windows Boot Foundation ISO image perform these steps:

NOTE: "BOOT Foundation ISO image" is my own term. This special image is a creation entirely conceived of and created by me.

The advantage of this method is that a Boot Foundation ISO image is VERY small, and it is trivially simple to create. Currently, it is smaller than 50 MB. This means that the huge dual architecture ISO image does not need to be kept merely to be able to create dual architecture images.

- Download the dual architecture image as noted above for option 1.
- Copy all files and folders EXCEPT the x64 and x86 folder to a temporary location.
- Create an ISO image from the folders and files in the temporary location (this program has a routine to create a generic ISO image that can be used for this purpose).

NOTE: Don't create the ISO with those files and folders in a subfolder. They should be at the root of the ISO image.

That ISO image is your Boot Foundation ISO image.

At this point the dual architecture image is no longer needed. Feel free to discard it. This will save nearly 8GB of storage space.

TIP: The routine to inject updates into one or more Windows editions will check each edition as it is processed for any "Pending Operations". As an example, if you enable NetFX3, your Windows edition will have a Pending Install operation. This will prevent DISM Image Cleanup operations from being able to run. The program will display a warning about this in the header if this condition is detected. Note that the program will continue to run, but you are probably best served by reapplying updates to images where Windows editions do not have pending operations. After the routine completes, you will also find a log file in the logs folder named "PendingOps.log". This log file will tell you what source file and index number(s) in that source had the pending operations.

Inject drivers into one or more Windows ISO images and create a multi edition bootable image

Note: Please read the section called "Acceptable Images" within the section "Inject Windows updates into one or more Windows ISO images and create a multi edition, bootable ISO from the updated images" above. The information there also applies to this routine.

The routine to inject drivers is very similar to injecting Windows updates, however, in this routine, when specifying the location of the drivers, the program will search all subdirectories for drivers.

Drivers need to have the .INF file(s) accessible. If drivers are packaged in .CAB files, this program has a routine available to allow files to be extracted from the .CAB files. Run that routine first to extract the drivers.

This process will **NOT** inject Windows updates. This routine only injects drivers. There is a separate option from the main menu that will allow Windows updates to be injected.

TIP: Drivers from multiple systems can be injected. Simply export the drivers from multiple systems using the routine in this program to export drivers from a system, then place them in a folder structure like this:

```
\Drivers
    \x64
        \Desktop PC
        \Laptop
        \Media System
    \x86
        \Tablet
```

When this routine is run and pointed to the “Drivers” folder, the drivers for all systems will be injected since all folders are recursed. Note that only drivers of the appropriate architecture type will be installed. For example, if you are working with only x64 editions of Windows, then only x64 driver versions will be installed.

Inject boot-critical drivers into one or more Windows ISO images and create a multi edition bootable image

Note: Please read the section called "Acceptable Images" within the section "Inject Windows updates into one or more Windows ISO images and create a multi edition, bootable ISO from the updated images" above. The information there also applies to this routine.

If you have a device that needs to have a driver loaded to install Windows or access your hardware from the Recovery Environment, you can inject those drivers into the WinPE and WinRE images using this routine.

As an example, suppose that you have a RAID controller for which drivers are not already included in Windows. Without this driver, Windows setup would not see the disks attached to this controller. At the point during setup where the user is asked to select a disk to install Windows on, there is option to load a driver. However, by injecting the boot-critical drivers into the Windows WinPE and WinRE images you will not need to load a driver manually. In addition, having the drivers already available on the installation media is necessary for unattended installation.

Organize your drivers like this:

```
\Drivers
    \x64
        \Desktop PC
        \Laptop
        \Media System
    \x86
        \Tablet
```

Make or update a bootable drive from a Windows ISO image

The program can create bootable media from a Windows ISO image.

Please note that we create two partitions for the operating system. There are two reasons for this:

1) Not all systems will boot from NTFS formatted removable drives. Starting the boot process from FAT32 allows for compatibility with such systems. So why not simply use a single FAT32 partition? That brings us to point #2.

2) FAT32 has a limit of a 4GB maximum file size. However, a typical distribution of Windows may contain a file larger than this. The solution is to begin the process from FAT32 where we don't need to store the large Windows image file, then place the rest on an NTFS partition. Don't worry, the program will take care of those details

- The media created with this routine can be booted from both BIOS and UEFI based systems.
- This routine can create additional partitions which can be used for other purposes.
- Additional partitions can be automatically BitLocker encrypted.
- Multi-architecture images are fully supported. This means that bootable media including both x86 and x64 editions on the same disk can be created.
- Program can update Windows boot image with a new version without affecting other partitions on the same drive.

This routine will always create a first partition that is 2.5 GB in size. This ensures enough space for boot files needed in any circumstance (both single and dual architecture bootable media). For all the remaining partitions, except the last partition, the user will be asked what the desired size for each partition is. The last partition will automatically be sized to occupy all the remaining space that is left which has not been allocated to other partitions.

TIP: To make the bootable media into media that can also be used for other purposes, create one or two additional partitions. The program will automatically ask you if you wish to do this.

Disk Limitations

Be aware that using MBR (the option to create a single bootable partition) has these limitations:

- Maximum of 4 partitions
- A disk size limit of 2TB (a larger disk can be limited to 2TB and still function)

Performing Unattended Installation from a Multi-Edition Boot Disk

When creating a customized multi-edition boot disk, there are some things to be aware of that affect unattended installation of Windows. This is a slightly lengthy topic, but it still deserves to be discussed in this user guide. Please see ***Appendix A - Performing Unattended Installation from a Multi-Edition Boot Disk*** for everything that you need to know if you plan to perform unattended installations using this media.

Create a bootable Windows ISO image that can include multiple editions and architectures

Much like the routines for injecting Windows updates and drivers, this routine will allow the creation of a bootable image that can include both x86 and x64 editions of Windows. The difference is that this routine will leave the Windows editions that are added to the image as they are without injecting any Windows updates or drivers. It simply copies all the Windows editions selected by the user and creates one new image that includes those editions. Note that these editions of Windows can come from multiple images, can include both x64 and x86 images, and can even included syspreped images.

Create a bootable ISO image from Windows files in a folder

When the contents of a Windows image are extracted to disk to modify any files or to add / remove files, this routine can be used to recreate a bootable image from the files on disk.

Reorganize the contents of a Windows ISO image

If an image with multiple Windows editions needs to be reorganized, for example, to change the order in which the different Windows editions are displayed in the boot menu, use this routine to do so. In addition to reordering the Windows editions, editions of Windows can be entirely removed from the image.

Convert an image with an install.esd into an image with an install.wim

Servicing an offline Windows image requires an image with an install.wim file rather than an install.esd. Microsoft likes the install.esd file because they compress better resulting in files easier to distribute electronically. In fact, ESD stands for "Electronic Service Delivery". If you don't access to an image with an install.wim, this routine will create a new ISO image for you with install.wim file(s) converted from an image with install.esd file(s).

Get WIM info – display basic info for each WIM in an ISO image

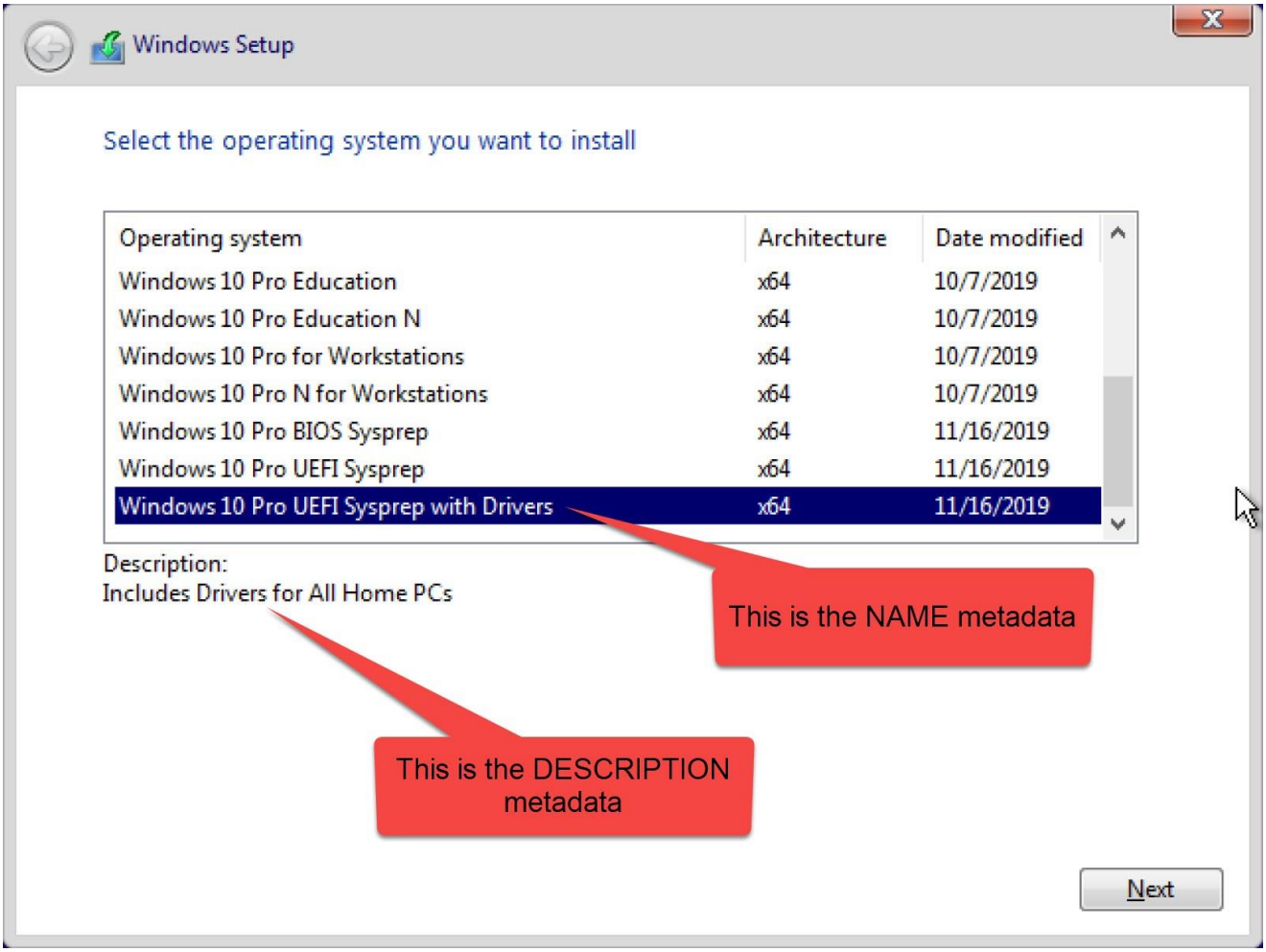
There are times where it may be necessary to know what index number is associated with a particular Windows edition, or how many editions are stored in an image, or to view the NAME and DESCRIPTION metadata for Windows editions. This routine will display that information and optionally save the output to a text file.

This routine will also display the Windows build number.

NOTE: The build number is obtained from the Windows edition at index #1 (for a single architecture image) or from the x86 edition at index #1 (for a dual architecture image). Since all images used with this program SHOULD have the same version of Windows for all editions within that image, we assume that all of the other editions will have the same build number.

Modify the NAME and DESCRIPTION values for entries in a WIM

When creating a Windows image with multiple editions of Windows, each edition must have valid NAME and DESCRIPTION metadata. This information is displayed in the boot menu when booting from the image or media created from that image. Here is a sample of what this looks like:



This routine will allow you to modify both the NAME and DESCRIPTION metadata.

Export drivers from this system

This routine will export all drivers from the system on which it is run. Those drivers can then be injected into a Windows image. Even if not injected into a Windows image, if you ever perform a clean install on the system from which the drivers were exported, all drivers can be easily reinstalled. When this routine exports the drivers, it will also create a batch file called "Install_Drivers.bat". By simply running this batch file, all drivers will be installed on the system.

Expand drivers supplied in a .CAB file

In order to inject drivers into a Windows image, the .INF file(s) need to be accessible. This routine will expand drivers supplied in a .CAB file so that the .INF file(s) are accessible.

Create a Virtual Disk (VHDX)

This routine will create a Virtual Disk. This can be helpful in several different scenarios. For routines that won't allow the use of a removable drive, a VHD can be created on the removable drive as a workaround. It can also be used as temporary storage for data, etc.

Create a VHD, deploy Windows to it, and add it to the boot menu to make a dual boot configuration

This routine can create an entirely new installation of Windows on a VHD that can be booted on a physical machine without the need for any virtualization software. Since this copy of Windows runs from a VHD it requires no separate disks or partitions on the system.

Create a generic ISO image and inject files and folders into it

This routine takes the contents of any folder, including all subdirectories, and create an ISO image from them. An ISO image can be handy because it can often be created much quicker than a ZIP or other compressed file format. It can also be easily attached to a VM to get files into a VM without the need for any network connectivity. Another good use for an ISO image is to simply store an answer file for unattended installation. This can then be mounted to a VM to allow fully unattended installation since Windows setup will search the root of all drives for an answer file.

Cleanup files and folders

Routines that inject Windows updates and drivers into Windows images will mount an image using the DISM utility. If an operation fails or is interrupted, it can leave stale mounted images in place. These mounts can persist even after a reboot and will prevent you from having full access to those folders. Running the routines that inject Windows updates and drivers will automatically run a cleanup routine, but this standalone routine will perform a cleanup without needing to run those routines.

Exit

Take a wild guess.

Appendix A - Automating Installation for a Multi Architecture Boot

When a multi-architecture boot disk (one that has both x64 and x86 editions of Windows) is created, unattended installation can still be run but there are several methods that can be used. The simplest method will require some initial interaction by the user, the second method requires a little more configuration but eliminates some need for user interaction, and the third and final method requires the most configuration but no user interaction.

Method 1

Simply place the autounattend.xml answer file on any partition of the boot media (Windows searches all partitions for this file). When booting from the media, Windows will still ask the user to pick x86 or x64 from the first menu if your image is dual architecture, then it displays all Windows editions matching the edition specified in the answer file. As an example, suppose that the answer file was created for Windows 10 Pro, x64. When booting, choose the x64 menu option. On the list of available Windows editions, only Windows 10 Pro editions will be displayed. If there is only one Windows 10 Pro edition, it won't be displayed. Installation will simply continue automatically. If more than one Windows 10 Pro based edition is available, choose the edition to install and from there on installation will proceed automatically.

Method 2

This method will require some modification of the autounattend.xml answer file.

Load the answer file in WSIM (Windows System Image Manager). In the Windows Image pane, expand the following tree:

Windows-Setup > ImageInstall > OSImage > InstallFrom

Right click on **MetaData** and select **Add Setting to Pass1 windowsPE**

In the Answer File pane, click on **InstallFrom** in the tree.

In the InstallFrom Properties pane, set the following value:

Path **\install.wim**

In the Answer File pane, click on **MetaData** in the tree.

Set the following values:

Key **/IMAGE/INDEX**
Value **<the index number of the Edition you want to install>**

NOTE: Each x64 Windows edition will have a unique index number: 1, 2, 3, 4, etc. Likewise, the x86 Editions will have index numbers 1, 2, 3, 4, etc.

This program includes a routine to display WIM information that will include a list of all the Windows editions and associated index numbers.

Example:

To install the edition of Windows that corresponds to index 6, set these values:

Key	**/IMAGE/INDEX**
Value	**6**

As always, perform a validity check on your answer file, and save it. Place it on the root of any partition on the boot media.

Now, when booting, it will still be necessary to select x64 or x86 from the boot menu, but then installation will be automatic without having to make any further selections.

Method 3:

To also automate the selection of the x86 or x64 option from the first menu, perform the steps for method 2 above and then perform the following steps:

In the example that follows, assume that the Windows bootable media is drive E:. Run this command:

bcdedit /store "E:\boot\bcd" /enum

NOTE: The above path references the location of the boot store for bootup on BIOS based systems. To work with the settings for UEFI based systems, change the path to **E:\efi\microsoft\boot\bcd**. You can apply the changes to the items below to either or both and can make different settings to each if you wish.

Output from the above command will resemble this:

```
Windows Boot Manager
--------------------
identifier              {bootmgr}
description             Windows Boot Manager
locale                  en-US
inherit                 {globalsettings}
default                 {default}
displayorder            {default}
                        {21f3fac8-05c4-11ea-824f-001583eeba66}
toolsdisplayorder       {memdiag}
timeout                 30

Windows Boot Loader
-------------------
identifier              {default}
device                  ramdisk=[boot]\x64\sources\boot.wim,{7619dcc8-fafe-11d9-b411-
000476eba25f}
path                    \windows\system32\boot\winload.exe
description             Windows 10 Setup (64-bit)
locale                  en-US
inherit                 {bootloadersettings}
osdevice                ramdisk=[boot]\x64\sources\boot.wim,{7619dcc8-fafe-11d9-b411-
000476eba25f}
systemroot              \windows
```

```
bootmenupolicy          Legacy
detecthal               Yes
winpe                   Yes
ems                     No

Windows Boot Loader
-------------------
identifier              {21f3fac8-05c4-11ea-824f-001583eeba66}
device                  ramdisk=[boot]\x86\sources\boot.wim,{7619dcc8-fafe-11d9-b411-
000476eba25f}
path                    \windows\system32\boot\winload.exe
description             Windows 10 Setup (32-bit)
locale                  en-US
inherit                 {bootloadersettings}
osdevice                ramdisk=[boot]\x86\sources\boot.wim,{7619dcc8-fafe-11d9-b411-
000476eba25f}
systemroot              \windows
bootmenupolicy          Legacy
detecthal               Yes
winpe                   Yes
ems                     No
```

From the output above, observe the following:

There are 2 Windows Boot Loader sections. One has a description of Windows 10 Setup (64-bit) with an identifier of {default}, the other, with the description of Windows 10 Setup (32-bit) has a GUID as the identifier, in this case {21f3fac8-05c4-11ea-824f-001583eeba66}.

Note also that in the Windows Boot Manager section, a timeout of 30 seconds is specified.

To fully automate Windows setup, set the timeout to 0 (meaning, don't display the menu, proceed immediately) and set the appropriate architecture type to be the default.

To change the timeout to 0, run this command from an elevated command prompt:

bcdedit /store "E:\boot\bcd" /timeout 0

Don't forget to modify the path for the UEFI boot store!

To change the default architecture type from x64 to x86, use this command:

bcdedit /store "E:\ISO_Files\boot\bcd" /default <GUID>

In my example, it would look like this:

bcdedit /store "E:\ISO_Files\boot\bcd" /default {21f3fac8-05c4-11ea-824f-001583eeba66}

If **bcdedit /store "E:\boot\bcd" /enum** is run again, the timeout should now be 0 and the x86 entry should show as {default} and a GUID is displayed for the x64 entry.

NOTE: To set the boot manager screen to have no timeout at all, meaning that it will wait forever for the user to make a selection, set the timeout to 4294967295. In case you are curious, that number is the equivalent of FFFF FFFF in hexadecimal.

Appendix B - Reviewing Log Files

After running any routine that injects files into the Windows image, you should review log files to make sure that all is well.

The most important log file is the "SANITIZED_ERROR_SUMMARY.log" file. Another log file, the "ERROR_SUMMARY.log" file is a summary of all errors generated by the use of the Microsoft DISM utility. At the time of this writing, Microsoft is aware that there are issues causing errors to be generated. As a result, you are certain to see errors in this log.

To make things easier, the program now features a routine that will "sanitize" the log file and clean it up by removing all the errors that can be ignored. The results are found in the file "SANITIZED_ERROR_SUMMARY.log". This is the log that you want to pay close attention to. Normally, this is the only log file that you should need to monitor after any routine to update Windows editions has been run.

Other Log Files

The original raw logs from DISM will have names such as dism.log_x64_1.txt. There will be one such log for each Windows edition updated, with the number after the second underscore representing the index number of that image. x86 editions of Windows will have x86 in the file name rather than x64.

There will be one such log for each Windows edition processed. These files can be very large and you will not typically need to look at these. If you find errors in the "SANITIZED_ERROR_SUMMARY.log", the errors listed there will reference the log file from which the error was taken and the timestamp of the error. You can then refer to that log file to review the error in the context of operations that were taking place at the time.

In addition, you will find log files with names such as "x64_1UpdateResults.txt". These log files will allow you to see what updates have been applied to your WIM images. One of these logs will be present for each x64 edition present. x86 editions of Windows will have identically named logs but starting with x86 rather than x64. Note that the number after the underscore will match the index number of that Windows edition in the final image.

Finally, the log named OSCDIMG.log will hold the results of the OSCDIMG utility used to create the final windows image.