Windows 10 IoT Enterprise: Image Development Workflow

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

September 2020

Windows 10 Version All

The New Last Chapter

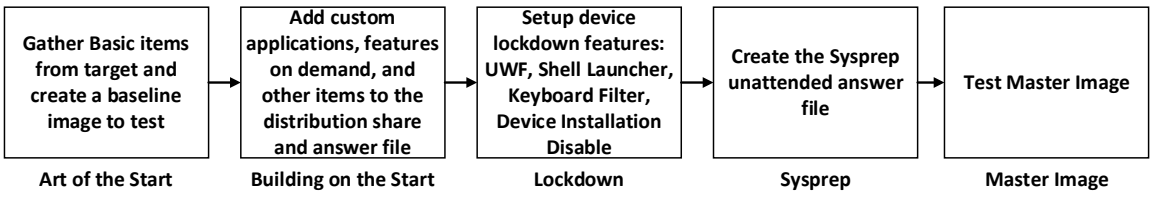
When Windows 10 launched in 2015, Microsoft was pushing the new universal Windows image build tool called Windows Image Configuration Designer (WICD pronounced "wicked"). At the time, the thinking was this: since everyone was going to be building UWP applications for Windows, WICD would help integrate applications easily into a custom Windows image. The problem was that many developers didn't jump on the UWP bandwagon. With deep investments in Win32, companies were not going to abandon their investments on a new application model that was limited and lacked a fully developed API set. To integrate Win32 applications into a custom image using WICD, one would have to use the User State Migration Tools (USMT), where the successful inclusion was a hit or miss. In the end, WICD was great for integrating UWP applications; but, living up to its name, it was pure evil when it came to Win32 applications.

Having written Windows custom image books since Windows NT Embedded, the problem was to pick a direction. There were several times that I had to warn clients off WICD and stick to the old tool System Image Manager (SIM) to build custom images. I went so far as to use The Onion News Networks: Sony's New product video to make the point. SIM still worked for Windows 10, but Microsoft broke some things, like the component settings for Shell Launcher and Unified Write Filter (UWF). I waited to see if Microsoft was going to fix WICD, but there were no improvements in the release that came out a year later. With customers asking for educational materials and seeing developers on the forums running into common problems, I scrambled to pull together *Starter Guide for Windows IoT Enterprise*. The problem with rushing to get a book published was leaving out some things that I knew could come out later. Windows 10 has some nuances that are slightly different than its predecessors, and that required some time to develop a proper workflow. With a few years of projects and some little discoveries, the goal of this paper is to add the missing chapter that brings all the individual topics together into a single workflow for design.

High Level Overview

Creating a Windows IoT image is more than just installing an OS from a DVD. Careful planning and design have to be used to address the needs for the full life cycle of the product. Customizing the OS shouldn't be the critical path. The workflow presented here is intended to make the customization and product management process a little easier.

The workflow goes back to the XP Embedded, and to some extent, the NT Embedded approach. As the OS has changed, so have the individual steps. Microsoft did provide a Development Manufacturing guide for Windows 10 IoT Enterprise. I know the folks who wrote this. They meant well, but I strongly do NOT recommend this as a development process. There are some clever ideas, but my philosophy is to make sure that the build process is set up to be as automated as possible, is documented so we remove human errors, has a consistent build process, and supports being able to make changes in the future. My workflow has the same goal, which is to create an image for production, but it takes a little work.



1. Art of the Start – The next section discusses the starting process that is the same for any projects. This step lays the foundation for the rest of the workflow.
2. Building on the start – Once the answer file and distribution share are set up, you can add your custom applications, device drivers, etc. that make your product unique. At this point, determine if features are set up in the master image or on each clone image.
3. Lockdown – Some of the lockdown features like Embedded Boot and Embedded Login can be added in an earlier step, but setting up and testing UWF, Shell Launcher, and the Keyboard filter has its own step. We want to make sure that the main application is set up correctly and working before locking down the system. I have some utilities that can help with this.
4. Sysprep – Running sysprep to create the master image is a critical step. You don't want to copy a live system as-is without sysprep or you could run into problems. Some planning has to go into the sysprep unattended file to address how each clone boots in manufacturing for the first time.
5. Master Image – sysprep and testing the master image are really one step, but what mechanism is used to capture and deploy the master image in manufacturing, Q&A, and development needs to be fleshed out.

The workflow is iterative. Each one of these steps will require the image to be built and possibly rebuilt. I know it is some work, but the process gets easier as you go. Most important is that you have the answer file and distribution share that get slightly modified over time so that you can go back to them to make changes. Compared to Microsoft's design guide process, you don't have to start over from scratch each time. The next section dives into different parts of the workflow.

Art of the Start

Those who have taken my Windows 10 IoT Enterprise course have seen the following presented as the "Art of the Start", and it is what we work through on the 3rd development day. The workflow start is broken into four parts:

Part 1: Gather Items from Target Installation

In this first part, the device drivers and other customizations are captured from the target system.

1. Install Windows 10 IoT Enterprise on your target system.

Note: In some rare cases, you may have to add disk drivers to support SSD drives or RAID drivers during installation.

2. Install all the device drivers for the target hardware.
3. Capture the device drivers:
    a. In File Explorer, create a folder in c:\ called "drivers".
    b. Open a command prompted with Administrative privileges.
    c. Run the following command:

```
DISM /online /export-driver /destination:c:\drivers
```

4. Plug in a USB flash disk, and copy the c:\drivers folder to a USB flash disk

Back in the day of XP Embedded, it could take a couple of days to create all the device driver components. Over time, I figure out how to get this completed in a few hours. Windows 10 makes gathering the device drivers supper easy. These steps above are important for four reasons. First, we need to make sure Windows 10 can run on the hardware. There is a wide range of PC platforms available, some new and some very old. If Windows 10 can install and run, then this is a good start. Second, by installing the device drivers, you can check to see if the drivers are working properly or if you need to download the latest drivers. Some device drivers require their installer to be set up properly, but most drivers can be installed via PnP via the "Out-of-box drivers" folder. Third, with the OS and drivers on the system, you can test your application to see how it performs. Forth, you

can gather other items needed to customize the image such as a custom security template, custom group policy settings, custom power plan, or custom firewall settings.

5. Create a custom security template to disable password time out. Following the Chapter 8 and Exercise 801 from Starter Guide for Windows 10 IoT Enterprise, create a security template to disable the password timeout. Copy the resulting INF file to a USB flash disk.

Note: The exercise in the chapter also has you configure to block a folder, but this is only for exercise demonstration. The steps to block the folder path are not needed.

6. Create custom group policy – Chapter 8 also discusses the creation of custom group policy settings. Below are the typical group policy settings that I set:

Computer Configuration

- Microsoft OneDrive for Business 2013 – Disabled
- Microsoft OneDrive for Business 2016 – Disabled
- Require a password when a computer wakes (on-battery) - Disabled
- Require a password when a computer wakes (plugged-in) – Disabled
- Save documents to OneDrive by default – Disabled
- Set the default behavior for AutoRun – Disabled
- Turn on Software Notifications – Disabled
- Disable Windows Error Reporting – Enabled
- Prevent the OneDrive files form syncing over a metered connection – Enabled
- Prevent the usage of OneDrive for file storage – Enabled
- Prevent the usage of OneDrive for file storage on Windows 8.1 – Enabled
- Turn off Autoplay – Enabled
- Turn off reminder balloons - Enabled
- Turn off System Restore - Enabled
- Turn off Windows Error Reporting – Enabled

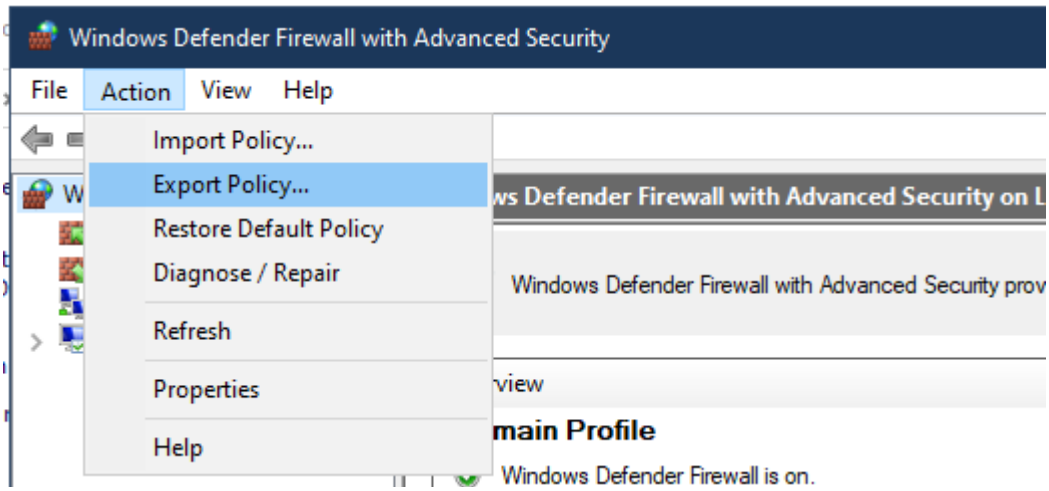
User Configuration

- Enable screen saver – Disabled
- Microsoft OneDrive for Business 2013 – Disabled
- Microsoft OneDrive for Business 2016 – Disabled
- Password protect the screen saver – Disabled
- Screen saver timeout – Disabled
- Set the default behavior for AutoRun – Disabled
- Desktop Wallpaper – Enabled
    - Wallpaper Name: <path to wallpaper bitmap >
    - Wallpaper Style – <option>
- Disable showing balloon notification toasts – Enabled
- Disable Windows Error Reporting – Enabled
- Load specific theme - Enabled
    - Specific Theme: <path to theme>
- Remove Balloon Tips on Start Menu items – Enabled
- Turn off all balloon notification – Enabled
- Turn off Autoplay – Enabled
- Turn off feature advertisement balloon notifications – Enabled
- Turn off reminder balloons
- Turn off toast notifications – Enabled
- Turn off toast notifications on the lock screen - Enabled

There are a lot of options that can be set in group policies, and some may require other support files. For example, specific background images, colors, and sounds can be put into a custom theme file, which then can be called by group policy. Some group policy settings shouldn't be set right away. For example, the group policy settings for blocking hardware devices from installing should set a little later. The paper on "Windows 10 IoT Enterprise (17763): Block Device Installation" discusses how this is performed. Follow the steps in the chapter to copy c:\Windows\System32\GroupPolicy to the USB flash disk.

7. Custom Power plan – Section 8.5 discusses creating a custom power plan. After you create your custom power plan, copy the resulting .pow file to the USB flash drive.

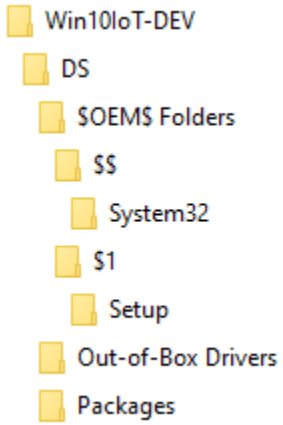
8. Custom Firewall settings – Custom firewall settings can be configured on the target and then exported to a .wfw file. The settings can be imported using the netsh.exe utility during OS installation. Copy the resulting file to the USB flash drive.



Part 2: Create Distribution Share and Answer File
With the device drivers captured and the custom items created, the next part takes place on the development system. We will set up a distribution share and create an answer file.

Note: It is assumed that you have completed Exercise 201 to generate a catalog file, and it can be opened in SIM. There have been issues with the latest SIM tool failing to generate a catalog. There are also KB updates to resolve the issues. You can always go back to an older version of the ADK, for example, ADK 1809, to get a version of SIM that isn't broken.

1. In file explorer, create a folder for your project. i.e. "c:\Win10IoT-Dev".
2. In this folder create a folder called DS (which is short for distribution share).
3. Open SIM, and create a new distribution share in the project folder "c:\Win10IoT-Dev\DS". The folders "OEM Folders", "Out-of-Box Drivers", and "Packages" will be created.
4. Using File Explorer, in the "c:\Win10IoT-Dev\DS" folder, create two new folders $$ and $1. $$ is the shorthand for "c:\Windows" and $1 is the shorthand for "c:\".
5. In $1 folder, create a folder called "setup".
6. In $$ folder, create a folder called System32



7. Plug in the USB flash disk

8. Copy the “drivers” folder on the USB flash disk to Out-of-Box Drivers. Rename “drivers” to the platform name or something a little more project name friendly.
9. Copy the custom power plan .POW file, the custom security template .INF file, and the custom firewall .WFW file from the USB flash disk to the “c:\Win10IoT-Dev\DS\$OEM$ Folders\$1\Setup” folder.
10. Copy the Group Policy folder from the USB flash disk to the “c:\Win10IoT-Dev\DS\$OEM$ Folders\$$\System32” folder.
11. If you set up a background image or a custom theme in the group policy, place the file in the appropriate location in the distribution share.

Placing items in the correct folder location is what makes the distribution share a better approach for including applications, drivers, etc. than USMT. There is no need for XML files, or crossing your fingers and hoping the files get put in the right location. As far as UWP applications, the app packages can be placed in the setup folder and installed using DISM. With the distribution share set up, the next step is to create an answer file.

12. In SIM, make sure the catalog is open in the Windows Image Pane.
13. Create a new answer file. Below is an example:

Configuration Pass	Component	Component Setting	Possible Value / Notes
1 WindowsPE	Microsoft-Windows-International-Core-WinPE	InputLocale	en-US
		SystemLocale	en-US
		SetupUILanguage: UILanguage	en-US
		UILanguage	en-US
		UserLocale	en-US
	Microsoft-Windows-Setup	UserData: AcceptEULA	True
		UserData : ProductKey : Key	<Enter the license key>
		ImageInstall: OSInstall : InstallFrom: MetaData: Key	/IMAGE/NAME
		ImageInstall: OSInstall : InstallFrom: MetaData: Value	Windows 10 Enterprise LTSC 2019
		UserData : ProductKey: WillShowUI	OnError
4 specialize	Microsoft-Windows-Shell-Setup	ComputerName	*
		ResteredOwner	ACME
7 OOBE System	Microsoft-Windows-International-Core	InputLocale	en-US
		SystemLocale	en-US
		UILanguage	en-US
		UserLocale	en-US
	Microsoft-Windows-Shell-Setup	AutoLogon: Username	ACME1
		AutoLogon: Enabled	True
		AutoLogon: LogonCount	4294967294
		AutoLogon: Password: Value	p@ssw0rd
		OOBE : ProtectYourPC	3
		OOBE : HideEULAPage	True
		OOBE : HideWirelessSetupInOOBE	True
		OOBE : HideOEMRegistrationScreen	True
		OOBE : HideLocalAccountScreen	True
		OOBE : HideOnlineAccountScreen	True
		OOBE: UnattendEnableRetailDemo	False
		TimeZone	Pacific Standard Time
		UserAccounts : LocalAccounts : LocalAccount : Name	ACME1
		UserAccounts : LocalAccounts : LocalAccount : Description	ACME1
		UserAccounts : LocalAccounts : LocalAccount : DisplayName	ACME1
		UserAccounts : LocalAccounts : LocalAccount :Group	Administrators

Configuration Pass	Component	Component Setting	Possible Value / Notes
		UserAccounts : LocalAccounts : LocalAccount : Password : Value	p@ssw0rd

14. Add the Windows Foundation Package to the answer file.
15. Enable and Disable the features that you want in the image. The following table is an example:

Package	Windows Feature Selection	Possible Value / Notes
Microsoft-Windows-Foundation-Package	Client-DeviceLockdown	Enabled
	Client-EmbeddedBootExp	Enabled
	Client-EmbeddedLogon	Enabled
	Client-EmbeddedShellLauncher	Enabled
	Client-KeyboardFilter	Enabled
	Client-UnifiedWriteFilter	Enabled
	Internet-Explorer-Optional	Disabled
	MediaPlayback	Disabled
	Printing-Foundation-Features : FaxServicesClientPackage	Disabled
	Printing-PrintToPDFDervices-Features	Disabled
	Printing-XPSServices-Features	Disabled

16. In the distribution share pane, expand the tree so you see the folder with all the device drivers.
17. Right-click on the folder and select “Install Driver Path Pass 2 offlineServicing”. This will add a component with the path to the drivers in the answer file.
18. Save the answer file.

Note: There is the Microsoft Deployment Tool (MDT), which his related to SIM, but this is for IT departments deploying to network computers. The answer file creation is very similar.

Part 3: Pass 7 Synchronous Commands

The answer file is not only for custom settings, but it acts as a setup script where you can install other applications, device drivers, and other custom settings. Again, the idea is to automate the build process as much as possible. Try to look for silent install switches, but it is perfectly fine to manually walk through an installer since we cannot automate everything. For this part, we will only add the synchronous commands to set up the items we gathered in Part 1. We only want to do a basic test at this point. You can add other items later. The commands will run during pass 7, when the system logs into an administrator account for the first time.

1. From the menu, you can select Insert-> Synchronous Command->Pass 7 oobSystem.
2. A dialog will appear asking you to enter the command to run. Below is an example list of synchronous commands.

    A. Import the custom power plan and set the GUID

powercfg.exe /IMPORT c:\setup\custom.pow cd8e9657-ae6f-4c34-8c7b-4023fc8ee91b

    B. Set the custom power plan as the active plan

powercfg.exe /S cd8e9657-ae6f-4c34-8c7b-4023fc8ee91b

    C. Disabled Hibernation

powercfg.exe /H OFF

D. Set custom security policies

secedit.exe /configure /db temp.sdb /cfg c:\setup\Custom-Security.inf

E. Import Firewall settings

netsh advfirewall import c:\setup\Custom-Firewall.wfw

F. Delete the copy of the installation disk from ConfigSetRoot

cmd.exe /C rd /S /Q c:\Windows\ConfigSetRoot

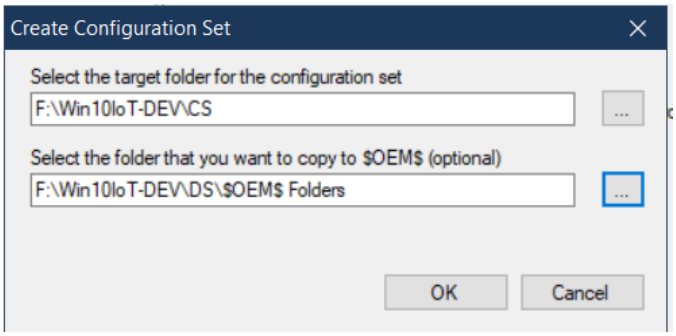
G. Set BCD to ignore all errors

bcdedit /set {current} bootstatuspolicy ignoreallfailures

3. When finished, save the answer file.

Part 4: Test the Basics
With the answer file created, it is now time to do a test of what has been created so far.

1. Create a Windows installation disk per Exercise 203.
2. In File Explorer, create a CS folder under the development directory: “c:\Win10IoT-Dev\CS”.
3. Follow the steps in Exercise 301 to create a configuration with the OEM Folder, but place the configuration in “c:\Win10IoT-Dev\CS”.



4. Copy the Configuration set to the USB flash disk that contains the Windows installer.
5. Plug the USB flash disk into the target systems and boot the system to the USB flash disk.
6. The system will boot to the partition screen. Partition the whole disk and begin the installation.
7. Once the installation has been completed, check that the device drivers, custom power plan, group policies, password timeout in local security policies, and windows firewall are set correctly.

These Four Parts define the basic workflow that I use for every single project. It doesn’t matter what the product is, the steps are the same. The goal is to lay a foundation for the rest of the project. At this point, you can build onto the answer file to add the items to lock down the system and address the needs of the product. To help with the Windows device lockdown features, there are some tools and approaches which are covered in the next section.

New Device Lockdown Utilities

As discussed in the book, Embedded Logon, Embedded Boot, and Keyboard Filter components can be added to the answer file, and the settings will be applied during OS installation. The same doesn’t go for Shell Launcher or Unified Write Filter (UWF). The funny thing is that these two

components worked fine with Windows Embedded 8.1 Industry. The reason the components don't work in Windows 10 goes back to the WICD tool that was supposed to be the Windows image build tool. After the book was released and with several client projects on the horizon, I looked to solve the problem.

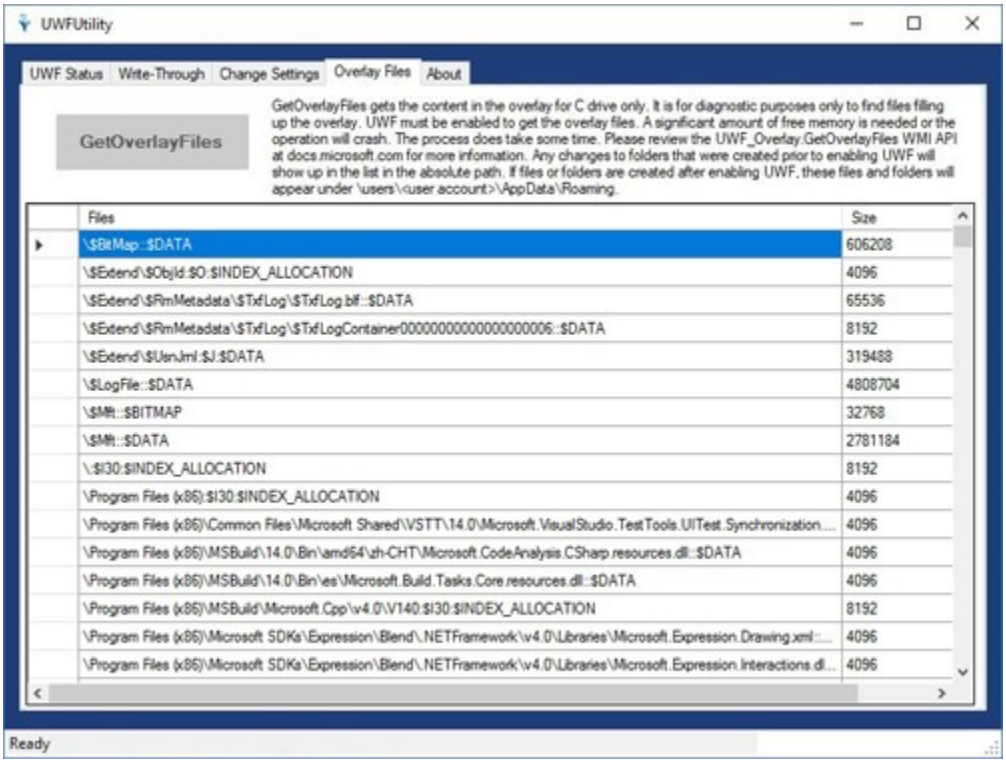
For UWF, you can create a batch file that makes calls to uwfmgr.exe. Here are the contents of a batch file that protects drive C with a RAM overlay and has some common folder and registry exclusions:

```
uwfmgr.exe overlay set-type ram
uwfmgr.exe volume protect c:
uwfmgr.exe file add-exclusion c:\Windows\System32\winevt\Logs
uwfmgr.exe file add-exclusion c:\Windows\assembly
uwfmgr.exe registry add-exclusion "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Time
Zones"
uwfmgr.exe                              registry                                add-exclusion
HKLM\SYSTEM\CurrentControlSet\Control\TimeZoneInformation
uwfmgr.exe                              registry                                add-exclusion
HKLM\SOFTWARE\Policies\Microsoft\Windows\WiredL2\GP_Policy
uwfmgr.exe registry add-exclusion HKLM\SYSTEM\CurrentControlSet\services\dot3svc
```

The batch file can be placed in the setup folder, and called with a synchronous command:

```
cmd.exe /C c:\setup\uwf.bat
```

You will notice that the filter is not set to be enabled. This is because the image needs to be sysprep'd to create the master image for production. UWF can be enabled on each clone of the system with the help of synchronous command. At this stage in the workflow, leave UWF setup and disabled. UWF can be manually enabled when you test the image, and you will want to test the application with UWF enabled before moving on to sysprep. Filling up the overlay is a real problem. We want to determine if other write-through sections have to be enabled so the system doesn't crash. As discussed in the book, large data files should be placed on separate, unprotected partitions. If you need to diagnose the UWF overlay, using the UWF WMI API set, I create a UWF GUI utility (https://annabooks.com/SW_UWFUtility.html) that has a tab that will list what is in the overlay.



UWF setup was a simple solution. Shell Launcher only has a WMI API set. Using the Shell Launcher WMI API set, I created a command line and graphical utility for shell launcher https://annabooks.com/SW_SLUtility.html. shlmgr.exe has command line switches similar to uwfmgr.exe

```
usage: shlmgr [command] [settings]
Commands:

enable | disable, Enable or Disable Shell Launcher
get-config, List the status and settings of Shell Launcher
set-defaultshell, Sets the default shell
set-usershell, Set up a shell for a specific user
remove-usershell, Remove a customer user shell
Help | ?, This help page

Examples:

shlmgr enable

shlmgr set-defaultshell <path to .exe> <DefaultAction>

where DefaultAcion - 0=Restart Shell, 1=Restart Device, 2=Shutdown
Device, 3=Do Nothing
shlmgr set-usershell <user name> <path to .exe> <DefaultAction>
where DefaultAcion - 0=Restart Shell, 1=Restart Device, 2=Shutdown
Device, 3=Do Nothing
User account must exist

shlmgr remove-usershell <user name>
User account must exist
```

A batch file can be created to make calls to shlmgr.exe to setup Shell Launcher. The batch file can be run during OS installation using a synchronous command.

Keyboard Filter requires some extra setup that involve enabled the keyboard filter service as discussed in Section 6.4. To assist managing the keyboard filter after OS installation, I created a GUI and command line utilities: https://annabooks.com/SW_KBFUtility.html

Again, you should test the lockdown features interacting with the whole system before moving on to sysprep.

Sysprep for Production

Once you have tested the application with the lockdown features, you can now look to create the master image. As discussed in Chapter 4, there have been many discussions about running sysprep or not running sysprep. Microsoft recommends running sysprep for deploying the image to different systems. The only rare cases where you wouldn't use sysprep is when a CRC on boot is needed. For example, slot machines require a CRC check on boot. Everyone else is going to run sysprep to create a master image. Since each clone of the master is going to run through a mini-setup, a sysprep unattended file is created using SIM:

1. Open SIM.
2. Create a new answer file that has the following settings:

Configuration Pass	Component	Component Setting	Possible Value / Notes
3 Generalize	Microsoft-Windows-PnpSysprep	PersistAllDevices	True
4 specialize	Microsoft-Windows-Shell-Setup	ComputerName	*
		ResteredOwner	ACME
7 OOBE System	Microsoft-Windows-International-Core	InputLocale	en-US
		SystemLocale	en-US
		UILanguage	en-US
		UserLocale	en-US
	Microsoft-Windows-Shell-Setup	AutoLogon: Username	ACME1
		AutoLogon: Enabled	True
		AutoLogon: LogonCount	4294967294
		AutoLogon: Password: Value	p@ssw0rd
		OOBE : ProtectYourPC	3
		OOBE : HideEULAPage	True
		OOBE : HideWirelessSetupInOOBE	True
		OOBE : HideOEMRegistrationScreen	True
		OOBE : HideLocalAccountScreen	True
		OOBE : HideOnlineAccountScreen	True
		OOBE : SkipMachineOOBE	True
		OOBE : SkipUserOOBE	True
		OOBE: UnattendEnableRetailDemo	False
		TimeZone	Pacific Standard Time
		UserAccounts : LocalAccounts : LocalAccount : Name	ACME1
		UserAccounts : LocalAccounts : LocalAccount : Description	ACME1
		UserAccounts : LocalAccounts : LocalAccount : DisplayName	ACME1
		UserAccounts : LocalAccounts : LocalAccount :Group	Administrators
		UserAccounts : LocalAccounts : LocalAccount : Password : Value	p@ssw0rd

3. You can also add synchronous commands to the sysprep unattended answer file. Here are some examples to consider:

   A. Enable Shell Launcher.
   B. Enable Keyboard Filter.
   C. Enable UWF, which will require a reboot that can be called in a later synchronous command.
   D. Disable Windows Update as discussed in the article: "Windows 10 IoT Enterprise (14393): Turn Off Windows Update and Managing Updates".
   E. Install SQL Express that is tied to a specific user account.
   F. Delete any setup folders used to create the image.

4. Save the answer file to the distribution share so that it is already in the image: "c:\Win10IoT-Dev\DS\$OEM$ Folders\$$\System32\Sysprep".

Once the master image has been set up, sysprep can be run from a command prompt with administrator privileges:

C:\windows\system32\sysprep> **sysprep /generalize /oobe /shutdown /unattend:c:\windows\system32\sysprep\myunattend.xml**

When the system shuts down, the system has the master image which can be duplicated to other systems. There are software and hardware solutions for capturing the image. Another article covers the WinPE, DISM, and FFU capture and restore solution. When each clone of the master boots, it will go through a mini setup and run the synchronous commands when the system logs into an administrator account for the first time.

Summary: Horse Meets Water

Many copies of my books have been sold throughout the years. I have taught many classes on the subject. Even though I try to evangelize a process, developers seem to go off the rails when building custom Windows images. Some developers make Windows image development more complicated than it is, and some make Windows image development complicated for job security. As a consultant, I am sometimes called in to fix the mess. Developing custom Windows images can be boring and a chore when there are other job responsibilities to perform, but it should not be treated lightly. The reason for the books, white papers, and training courses is to provide information about the features available. With the right knowledge, one should be able to successfully design a product for the whole product lifecycle. The workflow presented here is one that I have developed over the years and will continue to develop as Windows IoT changes. There is more than one way to design, build, and maintain custom Windows images. This workflow is not the only way, but has been developed and implemented successfully over the past 20 years.