Session 5: CUSTOMIZATION – II, ADD FILES / REGISTRIES & Unified writer filter

# ADDING FILES/REGISTRY KEYS TO THE IMAGE

**PURPOSE: Understand how to add files and registry setting to the image.**

Reference – [Add and Registries and Files](https://github.com/MicrosoftDocs/windows-iotcore-docs/blob/fabricam/windows-iotcore/manufacturing-guide/Customize-Image/AddFileRegistrySettings.md)

* Create a folder and move files to include to the image
* Open the **IoT Core Powershell Environment**: Rrun IoTCorePShell.cmd as an administrator from your Workspace and create a File package using [Add-IoTFilePackage](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/Add-IoTFilePackage.md):

|  |
| --- |
| *$myfiles = @(*  *("`$(runtime.system32)","C:\Temp\Files\TestFile1.txt", ""),*  *("`$(runtime.bootDrive)\OEMInstall","C:\Temp\Files\TestFile2.txt", "TestFile2.txt")*  *)*  *Add-IoTFilePackage Files.Configs $myfiles* |

This will create a new folder at C:\MyWorkspace\Common\Packages\Files.Configs and also add a FeatureID called FILES\_CONFIGS to the following file.

*C:\MyWorkspace\Common\Packages\OEMCOMMONFM.xml file*

**[Tip]** Variables like $(runtime.system32) are defined in C:\Program Files (x86)\Windows Kits \ 10\Tools\bin\i386\pkggen.cfg.xml

* Build the packages using [New-IoTCabPackage](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/New-IoTCabPackage.md)

*Ex) New-IoTCabPackage Files.Configs*

* Update the product test configuration to include the features using [Add-IoTProductFeature](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/Add-IoTProductFeature.md):

*Ex) Add-IoTProductFeature ProductA Test FILES\_CONFIGS -OEM*

* Create a Registry package using [Add-IoTRegistryPackage](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/Add-IoTRegistryPackage.md):

|  |
| --- |
| *$myregkeys = @(*  *("`$(hklm.software)\`$(OEMNAME)\Test","StringValue", "REG\_SZ", "Test string"),*  *("`$(hklm.software)\`$(OEMNAME)\Test","DWordValue", "REG\_DWORD", "0x12AB34CD")*  *)*  *Add-IoTRegistryPackage Registry.Settings $myregkeys* |

This will create a new folder at C:\MyWorkspace\Common\Packages\Registry.Settings and also add a FeatureID REGISTRY\_SETTINGS to the following file.

*Ex) C:\MyWorkspace\Common\Packages\OEMCOMMONFM.xml file*

* Build the packages using [New-IoTCabPackage](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/New-IoTCabPackage.md)

*Ex) New-IoTCabPackage Registry.Settings*

* Update the product test configuration to include the features using [Add-IoTProductFeature](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/Add-IoTProductFeature.md):

*Ex) Add-IoTProductFeature ProductA Test REGISTRY\_SETTINGS -OEM*

* Build the FFU image again. You should only have to run the New-IoTFFUImage command:

*Ex) New-IoTFFUImage ProductA Test*

# ADDING WIN32 SERVICE APP TO THE IMAGE

**PURPOSE: Understand how to add your service program to the image and how to configure it**

Reference – [Adding Win32 Service App](https://github.com/MicrosoftDocs/windows-iotcore-docs/blob/fabricam/windows-iotcore/manufacturing-guide/Customize-Image/AddingWin32Services.md) , Azure DM client device set up - [Link](https://github.com/ms-iot/iot-core-azure-dm-client/blob/master/docs/oem-device-setup.md)

* *SystemConfigurator.exe* which is a service to run Azure DM on client. You can find it under AzureDM.Services in HOL folder.
* There is no command for Win32 services so we should do some manually
* Create a subdirectory for your Win32 Service App under [Your Workspace]/Source-<arch>/Packages. This will contain the XML and EXE files to include when building the image. For example, refer to the ***AzureDM.Services*** subdirectory at [Your Workspace]/Source-<arch>/Packages/***AzureDM.Services*** for a working example.
* Create an XML file titled ***[The subdirectory name].wm.xml*** in the subdirectory you created from Step above. This file will specify how the package will be built. Here is an example of what that file should look like (you would replace the appropriate entries with your Win32 Service App information):

Ex) *AzureDM.Services – Refer* [*git*](https://github.com/ms-iot/iot-adk-addonkit/tree/master/Workspace/Source-arm/Packages/AzureDM.Services) *or C:\ iot-adk-addonkit\Workspace\Source-arm\Packages\AzureDM.Services*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <identity xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  name="Services"  namespace="AzureDM"  owner="$(OEMNAME)"  legacyName="$(OEMNAME).AzureDM.Services" xmlns="urn:Microsoft.CompPlat/ManifestSchema.v1.00">  <onecorePackageInfo  targetPartition="MainOS"  releaseType="Production"  ownerType="OEM" />  <files>  <file  destinationDir="$(runtime.system32)"  source="SystemConfigurator.exe" />  </files>  <service  name="SystemConfigurator"  start="auto"  type="win32OwnProcess"  objectName="LocalSystem"  errorControl="normal"  displayName="AzureDM System Configurator"  description="Provides Device Management JSON Interface"  imagePath="c:\Windows\System32\SystemConfigurator.exe">  <failureActions  resetPeriod="86400">  <actions>  <action  type="restartService"  delay="1000" />  <action  type="restartService"  delay="1000" />  <action  type="restartService"  delay="1000" />  <action  type="none"  delay="0" />  </actions>  </failureActions>  </service>  </identity> |

* Copy your EXE file to the subdirectory from Step above. This is your Win32 Service application executable.
* Open IoTCorePShell.cmd from your Worksapce. It should prompt you to run as an administrator.
* Build the package into a .CAB file with the subdirectory name(using [New-IoTCabPackage](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/New-IoTCabPackage.md))

*Ex) New-IoTCabPackage AzureDM.Services*

* Need to add the package name and add new Feature ID to the OEMFM.xml file manually

Example.

|  |
| --- |
| ……  <PackageFile Path="%PKGBLD\_DIR%" Name="%OEM\_NAME%.AzureDM.Services.cab">  <FeatureIDs>  <FeatureID>AZUREDM\_SERVICES</FeatureID>  </FeatureIDs>  </PackageFile>  </OEM>  <OEMFeatureGroups />  </Features>  </FeatureManifest> |

* The device management library used by the UWP application communicates with the NT Service, SystemConfigurator.exe, over a capability-protected RPC channel. So, we need to configure SystemConfigurator service to be ready. To configure the SystemConfigurator service, create a cmd file and invoke it from the main configuration script ***OEMCustomization.cmd*** (which is called on every boot).Put following commands to install service and start it when it boots.

|  |
| --- |
| ….  reg query HKLM\Software\IoT /v FirstBootDone >nul 2>&1  if %errorlevel% == 1 (  REM Enable Administrator User  net user Administrator p@ssw0rd /active:yes  *REM Azure DM service setup*  *c:\windows\system32\systemconfigurator.exe -install*  *c:\windows\system32\sc.exe config systemconfigurator start=auto*  *c:\windows\system32\sc.exe failure systemconfigurator reset= 0 actions= restart/0/restart/0/restart/0*  *net start systemconfigurator*  reg add HKLM\Software\IoT /v FirstBootDone /t REG\_DWORD /d 1 /f >nul 2>&1  ) |

**[Tip]** “ > /dev/null” redirects the stdout , “2>&1” redirects the stderr to the stdout

* Add the Feature ID for your service package using [Add-IoTProductFeature](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/Add-IoTProductFeature.md),

*Ex) Add-IoTProductFeature ProductA Test AZUREDM\_SERVICES -OEM*

* Build the FFU image again. You should only have to run the New-IoTFFUImage command:

*Ex) New-IoTFFUImage ProductA Test*

# ENABLE UWF ON THE IMAGE, CONFIGURE IT and verify it

**PURPOSE: How to enable UWF feature and how to use**

Reference – [Unified Write Filter](https://github.com/MicrosoftDocs/windows-iotcore-docs/blob/fabricam/windows-iotcore/manufacturing-guide/Secure-IoT-Core/UnifiedWriteFilter.md)

* Open IoTCorePShell.cmd from your Workspace
* Add IOT\_UNIFIED\_WRITE\_FILTER, **NOT** with “-OEM” parameter

*Ex) Add-IoTProductFeature ProductA Test IOT\_UNIFIED\_WRITE\_FILTER*

* Rebuild New-IoTCabPackage All

*Ex) New-IoTCabPackage All*

* Build the FFU image

*Ex) New-IoTFFUImage ProductA Test*

* Flash FFU image and check default setting was done
* Boot and connect PowerShell to the device
* Enable write filter Volume C:\ , need reboot to apply the setting.

*Ex) uwfmgr.exe filter enable*

*Ex) uwfmgr.exe volume protect c:*

*Ex) shutdown /r /t 0*

* Connect powershell and create file and check WUF working.

*Ex) Dir > list.txt*

*Ex) Shutdown /r /t 0*

* Connect powershell and check list.txt doesn’t exist.

# QUIz

Q1: Download MemoryStatus console tool from [Git Link](https://github.com/Microsoft/Windows-iotcore-samples/tree/develop/Samples/MemoryStatus) or HOL folder and add it to *(runtime.system32).* This will be useful to make your own environments for testing or manufacturing.

[Tip] If you got "link : fatal error lnk1104: cannot open file 'msvcprtd.lib'" error put following to your VC++ Directory > Library Directories and apply it.

*Ex) C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\VC\Tools\MSVC\14.16.27023\lib\$(PlatformTarget)*

