Session 3: Build UWP apps and drivers and making Imaging with them

# Build UWP apps , debugging it and create an image with it

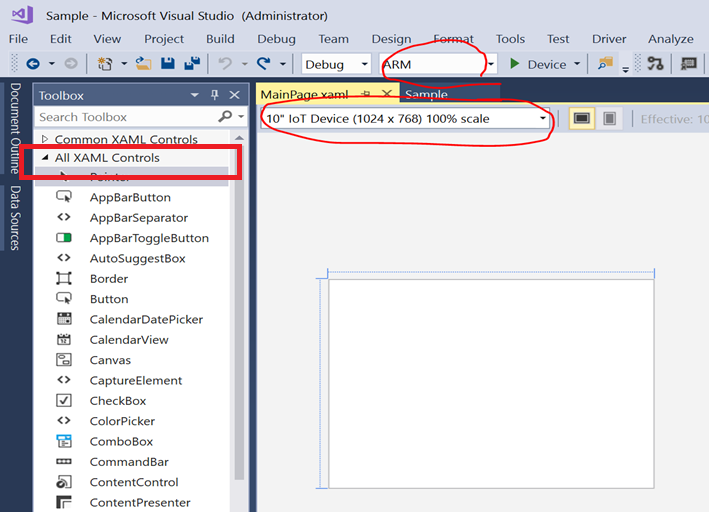
**PURPOSE: Understand how easy it is to develop app and create custom image**

Reference: Adding Apps – [Adding Apps](https://github.com/MicrosoftDocs/windows-iotcore-docs/blob/fabricam/windows-iotcore/manufacturing-guide/Customize-Image/AddingApps.md)

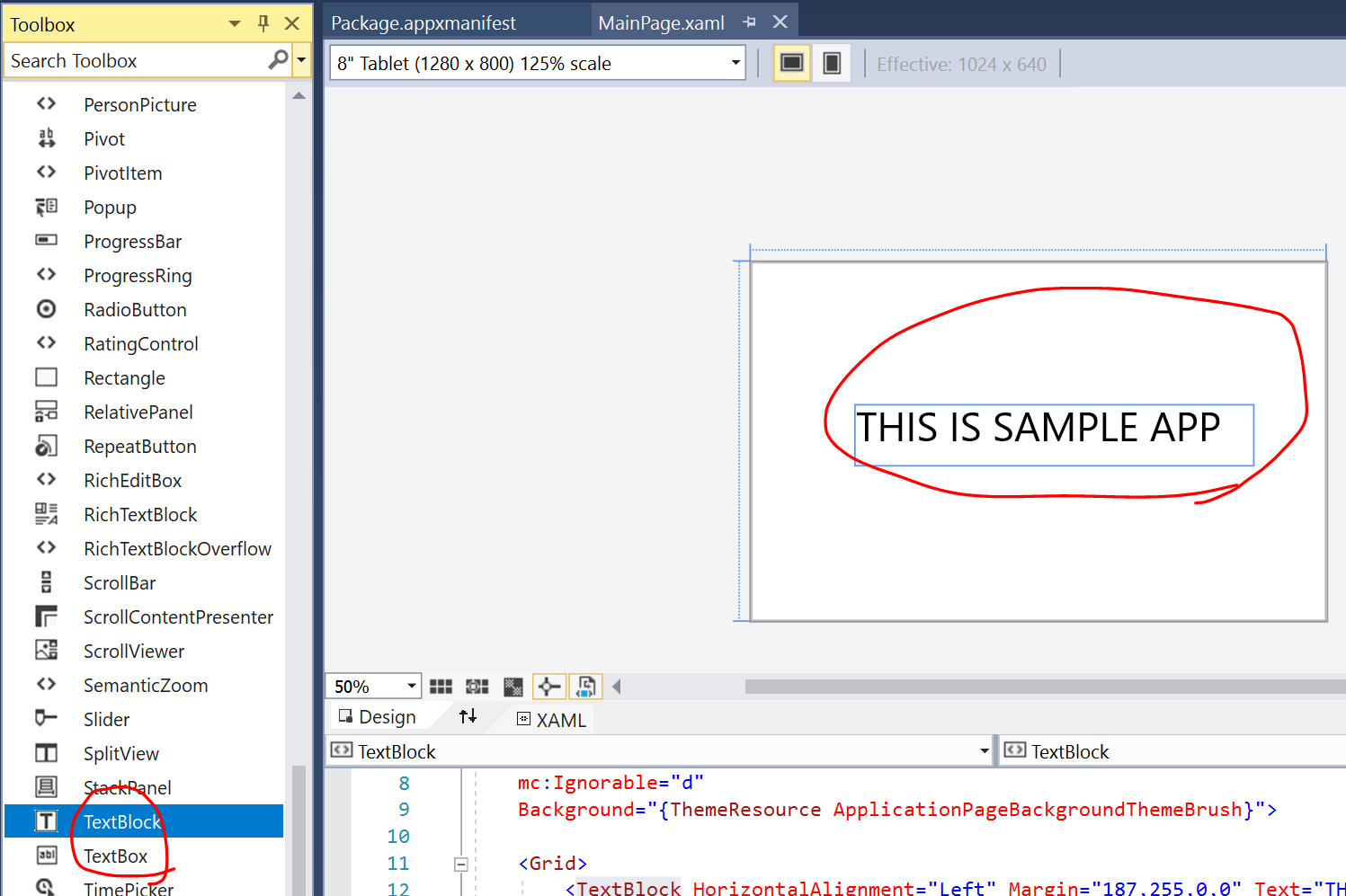
* Enable **Developer mode**, Settings -> Update & Security -> Developer mode
* Launch Visual Studio 2017
* Go to File -> New -> Projects -> Visual C# -> Windows Universal -> Blank App, create new project name
* Create new project “Sample” and select target version “1809” and keep minimum version to default.

[Note] if you change the minimum to 1809 then you will not see appx but msix, so suggest to keep default for the lab.

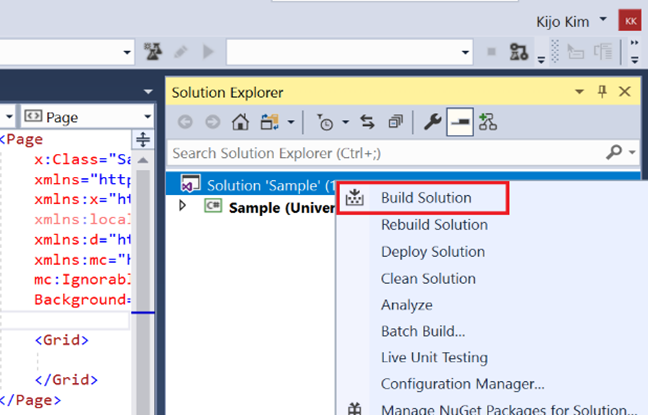
* Select target platform to **ARM**, optionally you can customize **resolution** information by double clicking the MainPage.xaml in Solution Explorer.



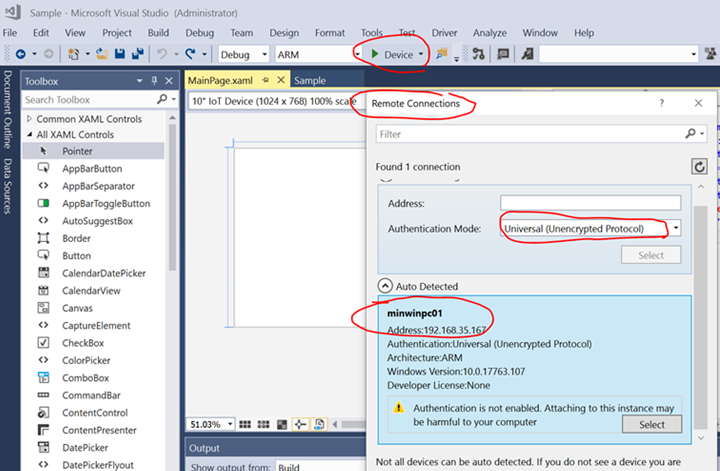
* Enable toolbox with Ctrl + Alt + X or goes to View -> Toolbox , enable All XAML Controls
* Select TextBox control and Put any text in UX page. You can customize text size in “Properties” box



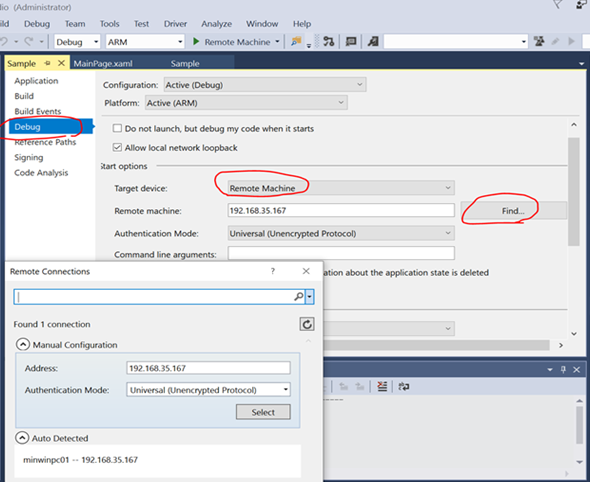
* Build solution



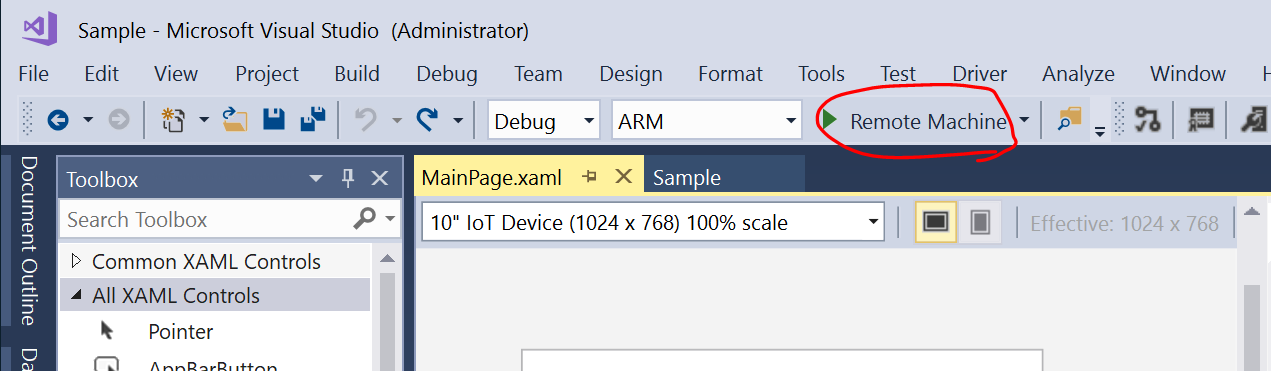
* Confirm our PC and RPi3 device is **under the same subnet**
* Configure Remote machine to deploy and debug**, Device -> Remote Machine**, or **Solution Explorer -> Property -> Debug -> Remote machine -> Find**, select your device name or IP address.



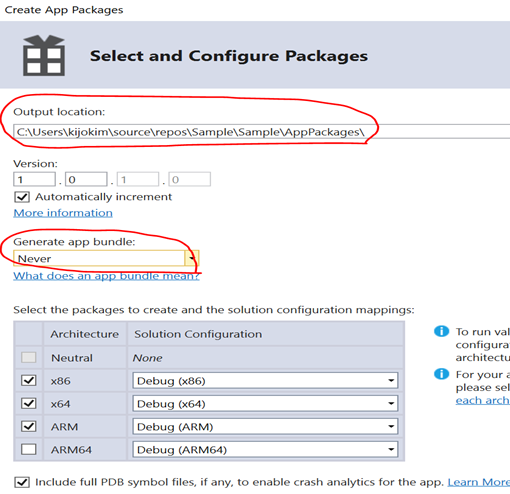
**[Note**] If you don’t see the list then you can just put the IP address directly



* Press **F5** or click **“Remote Machine”** for debugging

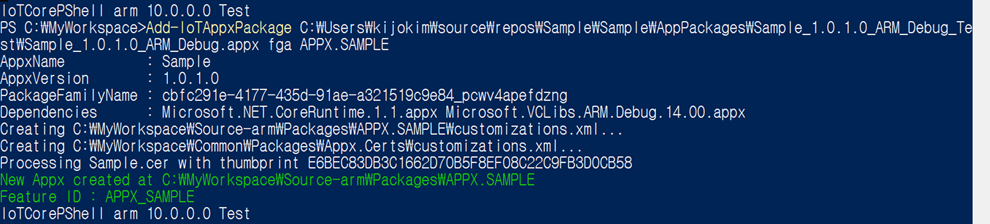


* To stop debugging, press **Shift + F5**
* In Visual Studio, to create your application as an Appx package. This is done by **clicking Project > Store > Create App Packages > I want to Create Packages for Sideloading > Next**
* Select **Generate app bundle**: **Never**, and select **ARM** only as we only target ARM device



* Click **Create**
* After completing package completion, you can access the package
* Open **IoTCorePShell.cmd** from your Workspace. It should prompt you to run as an administrator.
* Create the package for your Appx by using Add-IoTAppxPackage. In our example, the command is as follows:

*Add-IoTAppxPackage C:\Users\...\Sample\_1.0.1.0\_ARM\_Debug\_Test\Sample\_1.0.1.0\_ARM\_Debug.appx* ***fga*** *APPX.SAMPLE*

**

**[Note]** The **fga** parameter indicates the Appx file is a foreground application. If you specify your package as a background application (with the **bga** parameter) and have no other foreground applications in the image, the system will get stuck when booting up (displays a spinner indefinitely).

**[Note]**Be aware that if your Appx has **dependencies** you will need the Dependencies subdirectory to be present in the same location as your Appx when you run this command. If not then this will result in errors when you build your FFU image.

* From **IoT Core Powershell Environment**, you can now build the package into a .CAB file using [New-IoTCabPackage](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/New-IoTCabPackage.md). This will build the package into a .CAB file under [Your Workspace] \Build\<arch>\pkgs

*Ex) New-IoTCabPackage APPX.SAMPLE*

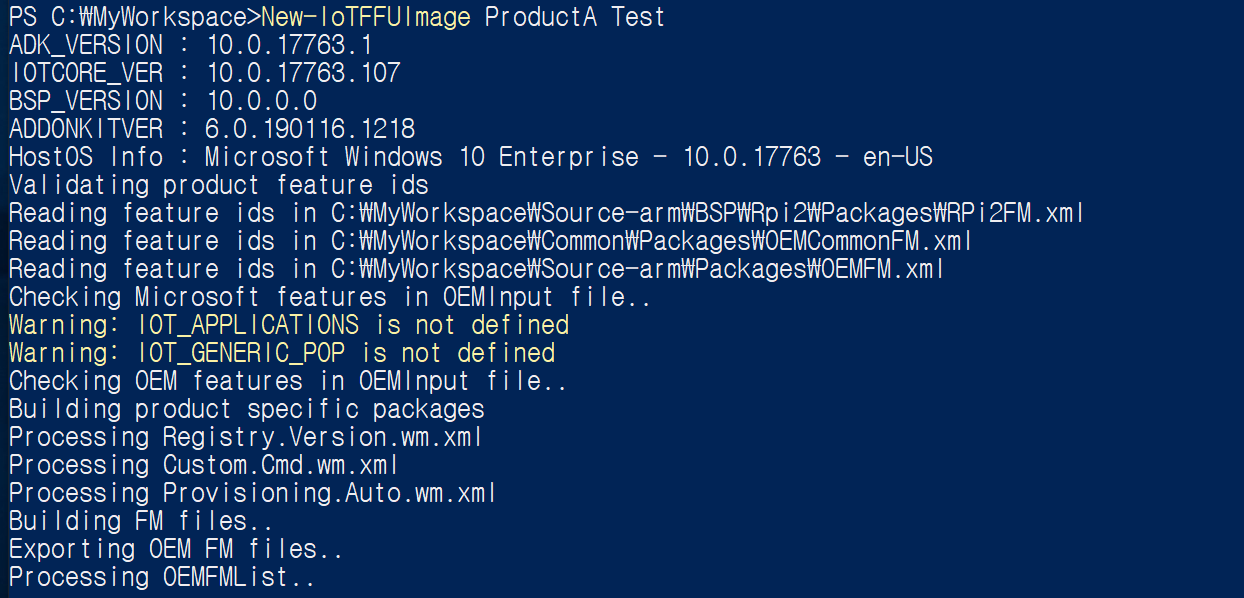
* Add the **Feature ID** for your app package using [Add-IoTProductFeature](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/Add-IoTProductFeature.md). This adds a FeatureID called APPX\_SAMPLE to the specified product's Test OEMInput XML file and include the package to the image.

*Ex) Add-IoTProductFeature ProductA Test APPX\_SAMPLE -OEM*

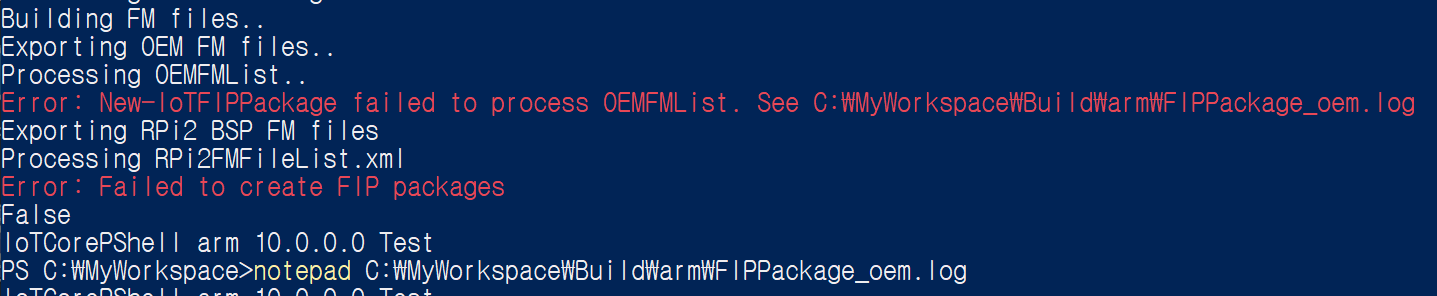
*Test OEMInput XML file : C:\MyWorkspace\Source-arm\<product name>\TestOEMInput.xml file*

* Build the FFU image again. You should only have to run the [New-IoTFFUImage](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/New-IoTFFUImage.md) command:

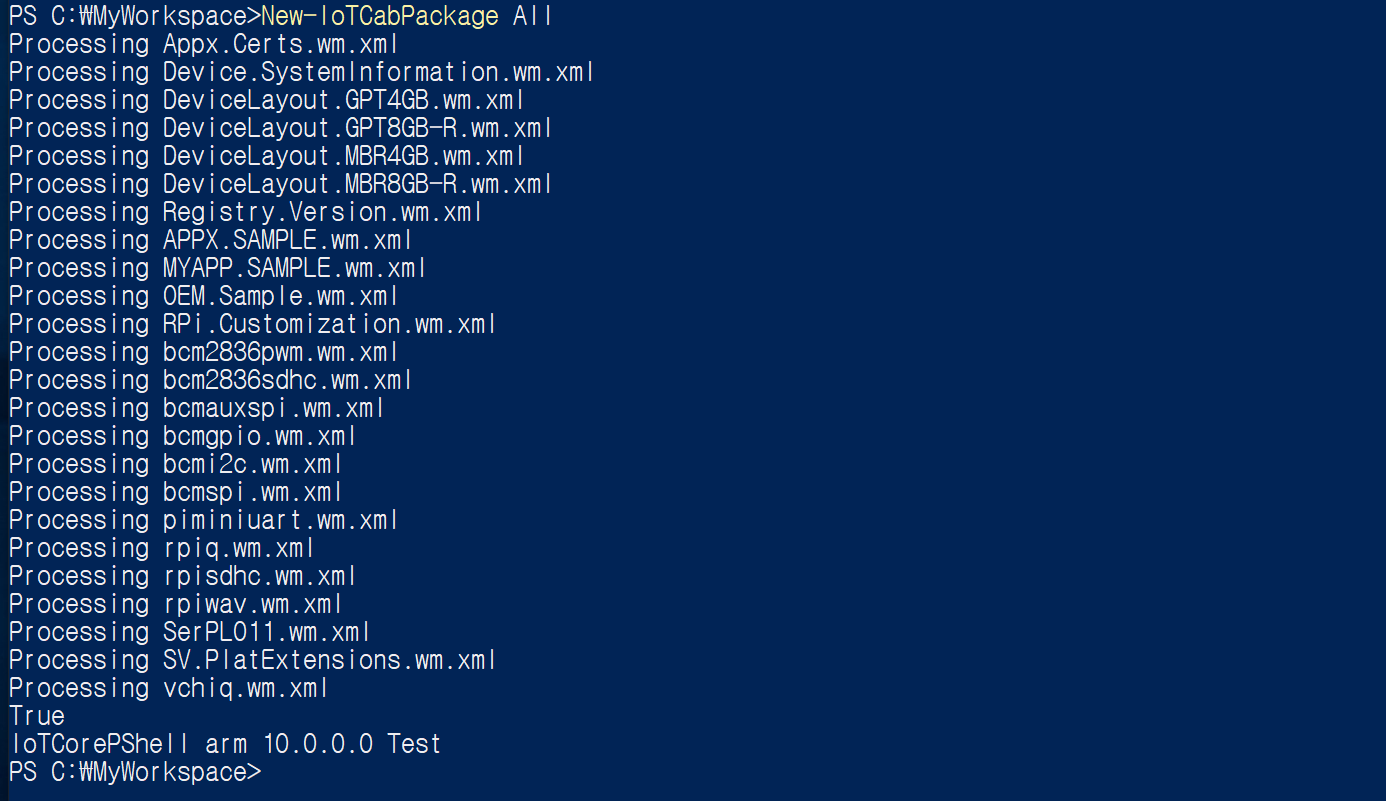
*Ex) New-IoTFFUImage ProductA Test*



**[Tip]**If you see New-IoTFIPPackage fail error as below then try New-IoTCabPackage All



Then, run New-IoTCabPackage All again and try build image again.



# Build KERNEL DRIVERS and Create an image with it

**PURPOSE : Understand how the driver is made and integrated to the image**

Reference, Adding Apps – [Adding Drivers](https://github.com/MicrosoftDocs/windows-iotcore-docs/blob/fabricam/windows-iotcore/manufacturing-guide/06d-AddingDrivers.md)

* As Community version is not supporting driver compile, so if you use Community version then skip to build KMDF drivers and use complied one under HOL folder
* Using Visual Studio, load gpiokmdfdemo project. Confirm Project -> Property
* Compile the gpiokmdfdemo project, located at [HOL folder]\Programs \Gpiokmdfdemo

**[Tip]** If you see code signing error, then please specify test cert as following

*“Windows OEM Test Cert 2017 (TEST ONLY)”*

* Copy the compiled driver files into a test folder (for example, c:\temp\gpiokmdfdemo) from your [HOL folder] \Programs\Gpiokmdfdemo\ARM\Debug

*gpiokmdfdemo.sys and gpiokmdfdemo.inf*

* Run IoT Core Powershell Environment from your Worksapce.
* Create a **driver package** using [Add-IoTDriverPackage](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/Add-IoTDriverPackage.md):

This creates a new folder at C:\MyWorkspace\Source-<arch>\Packages\Drivers.TestDriver. This also adds a FeatureID called DRIVERS\_TESTDRIVER to the C:\MyWorkspace\Source-<arch>\Packages\OEMFM.xml file

*Ex) Add-IoTDriverPackage C:\temp\gpiokmdfdemo\gpiokmdfdemo.inf Drivers.TestDriver*

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

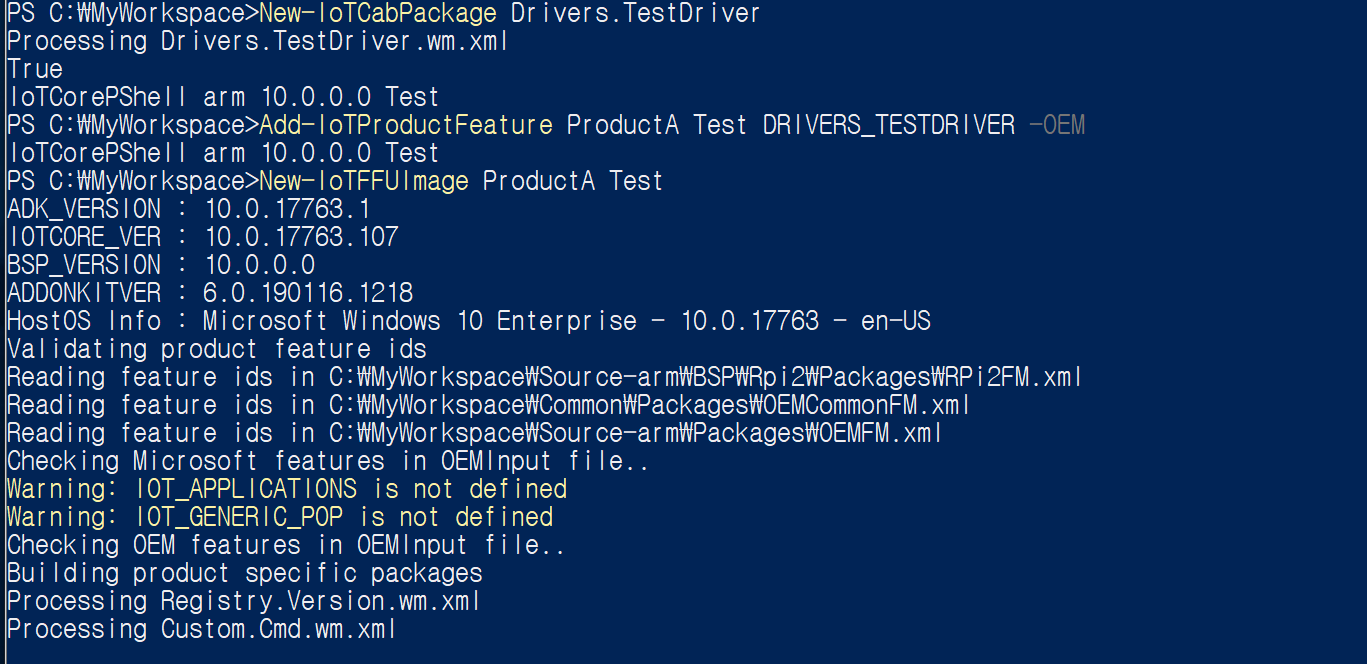
*Ex) New-IoTCabPackage Drivers.TestDriver*

* Enable the feature to add the driver package to the image. Update the product test configuration file using [Add-IoTProductFeature](https://github.com/ms-iot/iot-adk-addonkit/blob/master/Tools/IoTCoreImaging/Docs/Add-IoTProductFeature.md):

*Ex) Add-IoTProductFeature ProductA Test DRIVERS\_TESTDRIVER -OEM*

* Build the FFU image again, as specified in Creating a Basic IoT Core Image. You should only have to run the New-IoTFFUImage command:

*Ex) New-IoTFFUImage ProductA Test*



[Tip]if you see following Get-ChildItem error then run it again after reboot.

