Qualcomm Technologies, Inc.

Windows 10 IoT Core Board Support Package (BSP) for DragonBoard™ 410c

Customization Guide

LM80-P0436-67 Rev. D

September 24, 2018

For additional information or to submit technical questions, go to: https://discuss.96boards.org/c/products/dragonboard410c

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	April 2017	Initial release
B	January 2018	Chapter 5 Building ACPI: Updated the procedure for compiling ACPI
C	April 2018	Chapter 5 *Building ACPI*: Edited steps 6 and 7 in the procedure for compiling ACPI
D	September 2018	Numerous updates to content. Read the document in its entirety.

Contents

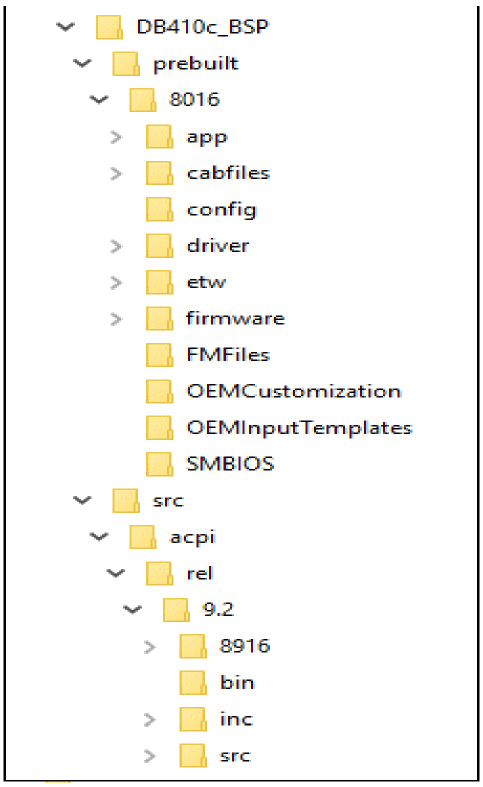
1 Windows Board Support Package (BSP)

Windows BSP downloaded from https://developer.qualcomm.com/hardware/dragonboard-410c/tools is zipped up in a file of the form <BUILD_ID>_DB410c_BSP.zip

For example, A8016AAATTNWZA21100000.1_DB410c_BSP.zip

NOTE: Use BUILD_ID as a reference when posting concerns

- Upon unzipping the file the following folder structure is visible:



- The folder "prebuilt" has the binaries delivered by Qualcomm® that are required for booting the device.
- The folder "src" has the ACPI source code that can help reprogram hardware resources or creating ACPI enumerated wdf drivers.
- The file "QC_Version.txt" has information about the Qualcomm Build ID

<qcversion>A8016AAATTNWZA21100000.1</qcversion>

2 Building FFU image

FFU stands for Full Flash Update. FFU image bundles boot loaders, Windows OS, drivers, peripheral images, and other required files into a single package that can be flashed onto the device.

By flashing FFU to the image all of the required software is flashed on to the target at once.

Required Software:

- Matching Microsoft WDK (refer to the qclabel.xml file for the required version) under "<target_root>\DB410c_BSP\prebuilt\8016\firmware\wpk

  Example: <build>14955.14955</build>

- Microsoft ADK (matching the installed WDK).
- Copy MobileOS folder from Microsoft WDK drop to <target_root>\DB410c_BSP\prebuilt\MobileOS
- [OPTIONAL] Prior to building the FFU image [follow only if a new driver is required in the FFU]:
- Copy all the required spkgs of new drivers into the <target_root>\DB410c_BSP\prebuilt\8016\cabfiles folder.

See https://msdn.microsoft.com/en-us/library/windows/hardware/dn756642(v=vs.85).aspx for information on generating a spkg with your driver.

- Make an entry for all the new spkgs in the following feature manifest file: "<target_root>\DB410c_BSP\prebuilt\8016\FMFiles\QCFMOEMCustomized.xml" e.g. To add spkg "Custom.QC8016.ExampleDriver.spkg" at the location "<target_root>\ DB410c_BSP\prebuilt\8016\cabfiles" insert the following entry under

  <BasePackages>

  <PackageFile Path="**%QCPackageDir%\cabfiles**" Name="**Custom.QC8016.ExampleDriver.spkg**"/>

  </BasePackages>

NOTE: %QCPackageDir% is pointing to <target_root>\ DB410c_BSP\prebuilt\8016\cabfiles

- Open an Administrator-Mode command prompt window
- Setup the following environment variables
- set wdkcontentroot=c:\Program Files (x86)\Windows Kits\10\

- Point it to the location of the tools folder in the installed wdk (note the '\' at the end)
    - MSPackageDir=<target_root>\DB410c_BSP\prebuilt\MobileOS\MSPackages
    - OSContentRoot=<target_root>\DB410c_BSP\prebuilt\MobileOS\
    - Set device=8016
    - Set qcplatform=8916
    - Set QCNHDebug=\
    - Set bsproot=<target_root>\DB410c_BSP\prebuilt\8016
    - Call postbld.bat
    - set bsproot=<target_root>\DB410c_BSP\prebuilt\8016
    - Call postbld.bat

3 Mass storage mode

To set the device in to mass storage and directly access:

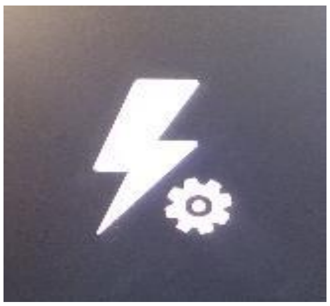
1. Open an Administrator-Mode command prompt
2. Ensure device enumerates in FFU mode [see FFU Mode section]
3. Run the following command:

```
"<wdkcontentroot>\Program Files\Windows Kits\10\Tools\bin\i386\ffutool.exe"
-massStorage
```

Device resets and enumerates as mass storage drive in the host PC.

4 FFU mode

FFU mode is very helpful for developers to directly access the file system on the dragon board. A developer can first put the device into mass storage mode and then view/modify the files with windows explorer. The device in FFU mode will show the following graphic on the display device:



- By default the FFU mode is disabled. A device flashed with FFU will boot directly to screen with tiles.
- To enable FFU mode, user can follow one of the following approaches:
  - Enabling FFU Mode by default during FFU generation:
  - To have the "FFU mode" enabled in the FFU by default, before generating the FFU, edit the file `<target_root>\DB410c_BSP\prebuilt\8016\oeminputtemplates\oeminput_productionos.x ml.`

    Under the following tag:

    ```
    <Microsoft>
    …
    <Feature>LABIMAGE</Feature>
    …
    </Microsoft>
    ```
- In the default FFU, user can also go to FFU mode by pressing and holding the **"Volume up"** key on the device as soon as the device powers up.
- Enable FFU mode from mass storage mode:
  - User can also set a BCD entry in the default image to have the device always halt in the
  - FFU mode.
  - Put the device in to mass storage [refer to Mass storage mode section]
  - Device now enumerates in host PC. Remember the drive letter.

- Run the following command:

```
bcdedit.exe /store <mass-storage drive
letter>:\efiesp\efi\Microsoft\boot\bcd /set
{globalsettings} custom:26000203 yes
```

5 Building ACPI

Windows BSP shipped via QDN packs in the ACPI source files for reconfiguring the DragonBoard hardware. BSP also packs few helper batch files, which when executed generates both the acpi binary- *dsdt.aml* and the acpi spkg - *ualcomm.M8016SOC_SBC.acpi.spkg*.

To compile ACPI perform the following steps:

- Open an Administrator-Mode command prompt
- Set wdkcontentroot=c:\Program Files (x86)\Windows Kits\10\
- Point it to the location of the tools folder in the installed wdk (note the '\' at the end)
- Execute "C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat" x86_arm

NOTE: If the development machine is used for the first time to build any of the windows software (drivers or applications) against WDK, install OEM Certificates. See section Installing OEM Certificates below.

- cd to the location of build_acpi.bat [i.e. *<target_root>\DB410c_BSP\src\acpi\rel\9.2*]
- Run build_acpi.bat

ACPI binaries are found at <target_root>\DB410c_BSP\src\acpi\rel\9.2\TABLES

ACPI spkg is found at <target_root>\ DB410c_BSP\src\acpi\rel\9.2\arm\Debug

6 Replacing ACPI

- Put the device in mass storage mode [see Mass storage mode section.]
- Use the following command to replace with new dsdt.aml file:
```
copy /Y <source dsdt.aml>  \\?\HardDisk1partition26\acpi
```

7 Installing OEM certificates

To install OEM Certificates perform the following steps:

- Open an Administrator-Mode command prompt window
- Set wdkcontentroot=c:\Program Files (x86)\Windows Kits\10\
- Set wpdkcontentroot=c:\Program Files (x86)\Windows Kits\10\
- set PATH=%PATH%;%WDKContentRoot%\Tools\bin\i386;%WDKContentRoot%\bin\x86
- set SIGN_OEM=1
- installoemcerts.cmd

8 How to write ACPI code for my driver

A developer must define their custom ACPI entries in order to develop their own device drivers. The following section describes a sample SSDT.ASL file and provides a way for the developer to define their own bus & gpio entries based on the hardware schematics.

See Sample ACPI Code in Appendix for the entire ACPI sample.

NOTE: See ACPI specification at http://www.acpi.info/DOWNLOADS/ACPIspec50.pdf for complete details of the Resource descriptors and the ASL information.

- All developer written code must be present in the SSDT.ASL file. SSDT stands for Secondary System Description Table. Also, it is the responsibility of the developer to make sure that entries in SSDT.ASL do not conflict with Qualcomm entries in DSDT.ASL
- The SSDT.ASL file can be compiled to SSDT.AML using the compiler asl.exe.
- The SSDT.AML file can coexist along with Qualcomm's DSDT.AML file in the PLAT
- partition of the device.
- The SSDT.AML file can be copied to the target's PLAT partition from mass storage mode of the device.

8.1 DefinitionBlock

The following rectangle depicts the standard way of defining the SSDT.ASL DefinitionBlock. Please make sure the first line stays the same even in your custom implementations.

All Device () definitions must be scoped to system bus – "_SB_" as shown below

```
DefinitionBlock ("SSDT.AML", "SSDT", 0x02, "USERS ", "MSM8016 ", 1) --->
Use this line as is
{
// should be scoped under System Bus
Scope (\_SB_) ---> This entry should be used as is
{
// All device entries must fall in this block
Device (DEV1) {}
Device (DEV2) {}
...
}
}
```

8.2 Device entry

A device declaration entry must not be conflicting with existing Qualcomm entries. If there is a conflict the DragonBoard's behavior can be unpredictable. Please make the proper choice of names when writing a device entry.

In the device entry a developer can declare the resources that are consumed by the driver software once the device boots up. The following rectangle shows an example Device entry.

```
// User Sample Device
Device (USRS) ---> Make sure the name does not conflict with any other entries
{
Name (_HID, "USRS0000") ---> Make sure this Hardware ID does not conflict with other entries.
Name (_UID, 1)
Method (_CRS) {} ---> This method will hold the resources consumed by the driver.
Method (xxxx) {} ---> Developer can define their custom methods to parse in driver code
...
Method (zzzz) {}
}
```

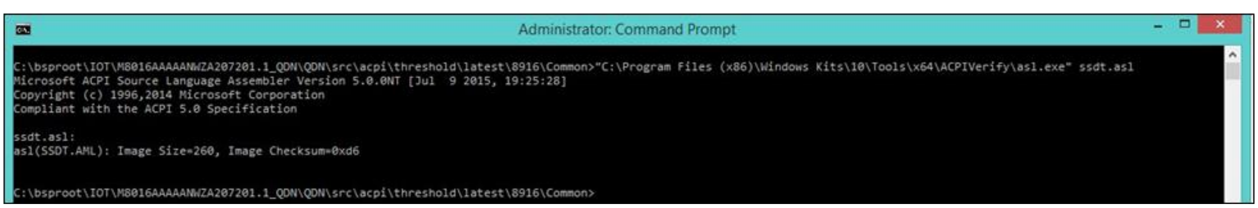
Each Device () entry is associated with a driver module with "_HID" value described as above. These _HID values must be non-conflicting with other _HID values. The _HID value referred to must also match up with the Hardware ID defined in the drivers .inf file for the driver loading to be successful

Please refer to the Appendix for samples on defining entries for I2C, SPI, UART and GPIO. The comments in the section will help understand the ACPI definition process.

8.3 Compiling the SSDT.ASL

Follow these steps to compile your ACPI:

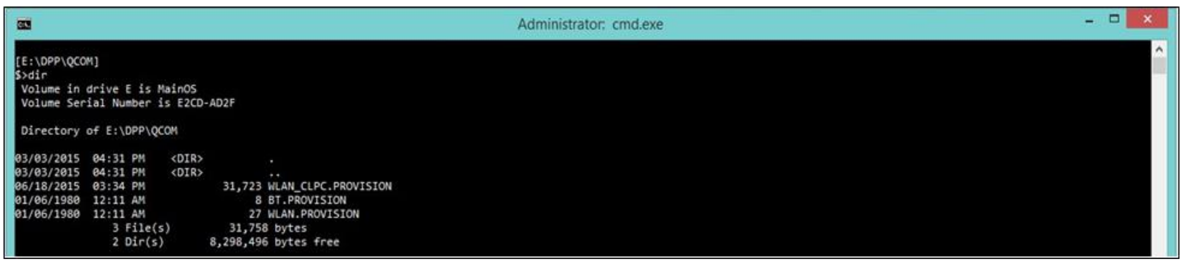
1. Launch an admin mode command prompt
2. CD to the location of ssdt.asl and execute:
   ```
   "C:\Program Files (x86)\Windows Kits\10\Tools\x64\ACPIVerify\asl.exe" ssdt.asl
   ```
3. Output will be stored in the current working directory



9 DPP – Device provisioning partition

Qualcomm BSP stores certain device related information in the DPP Partition of the DragonBoard. This partition essentially holds provisioning files that contain MAC addresses for Bluetooth and Wi- Fi connectivity.

After flashing the FFU for the first time and the device is left to boot to tiles the System creates the files WLAN.PROVISION and BT.PROVISION (These can be viewed from mass storage mode as shown below). These files have the auto generated MAC addresses for Bluetooth and Wi-Fi functionality.

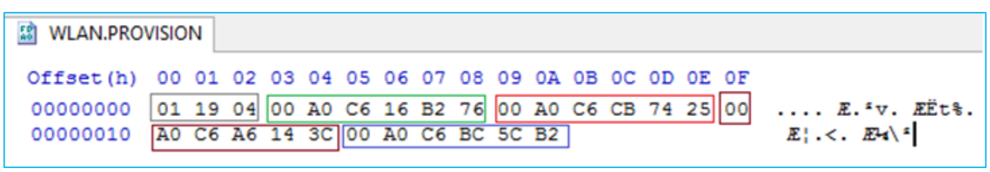


A developer can manually modify these files (using Hex Editor) to configure known static MAC addresses for Wi-Fi and Bluetooth. With known static MAC addresses a predictable Wi-Fi and Bluetooth behavior can be achieved. Notice that a MAC address is a unique 6 byte value and must not be conflicting with devices on the same network.

The format of the provision files are as shown below:

9.1 WLAN.PROVISION

This file holds 4 MAC address values as highlighted in the diagram below.



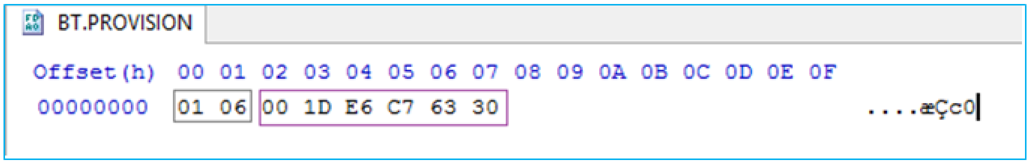
- Green Box – This is used for `Soft Access Point` connectivity
- Red Box – This is used for `Wi-Fi station` connectivity
- Maroon Box – This is used for `P2P client` connectivity
- Blue Box – This is used for `P2P Go` connectivity

The developer can hand edit these 4 6 byte values to configure a known mac address.

The developer **must not** edit the 3 byte file header which are used to distinguish between the provisioning files.

9.2 BT.PROVISION

This file holds 1 MAC address value that is used by Bluetooth hardware. The following picture shows the location of the BT MAC address in the BT.PROVISION file.



The developer can hand edit this 6 byte value to configure a known Bluetooth mac address.

The developer **must not** edit the 2 bytes of the file header which are used to distinguish between the provisioning files.

10 Must know

10.1 ACPI

A developer can add their own SSDT (Secondary System Description Table) tables without making any changes to the Qualcomm delivered ACPI code. But, the developer must ensure that the entries are not conflicting with Qualcomm delivered source code.

10.2 USB mode detection

DB410c has one micro-B connector for peripheral mode and two standard-A connectors for host mode. The device can only be used in either host mode or peripheral mode at any one time. Windows USB mode dynamic detection relies on the presence of the micro-B connector port. The following table lists possible connector configurations and the mode detected:

Standard-A	Standard-B	USB Mode
No	Yes	Peripheral
Yes	No	Host
Yes	Yes	Peripheral

10.3 Asix Ethernet-USB dongle

DB410c has inbox support for Asix Ax88772 Ethernet-USB dongle in order to use wired Ethernet. Use of this dongle will provide you with faster internet speeds.

10.4 Wi-Fi caveat

It is very important to note that every flash of FFU generates a new random mac address. Since the MAC address changes for every flash it will also force repopulate your Wi-Fi access points with new mac addresses and might break your current test setup. Please make arrangements to take care of these situations when setting up test scenarios.

11 Secure Boot Enablement

11.1 Creating Signed FFU

Follow the below steps to sign the NHLOS images:

NOTE: The Machine should have Internet connectivity.

1. Install necessary tools 1) Openssl 1.0.1g or later 2) Python 2.7 or later (Python 3 is not supported)
2. Create working directory Create directory WP_SEC_BOOT, and it subdirectories OemPKCertificate, SSDKeys, fuse, firmware, signedmbn and cabfiles.
3. Generate OEM PK cert Go to WP_SEC_BOOT\OemPKCertificate, open command window as Administrator.

```
openssl.exe "genrsa" -out "qpsa_rootca.key" -3 "2048"
```

```
C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\OemPKCertificate>openssl.exe "genrsa" -out "qpsa_rootca.key" -3 "2048"
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
....+++
...+++
e is 3 (0x3)
```

qpsa_rootca.key is generated.

```
Set OPENSSL_CONF=C:\OpenSSL\bin\openssl.cnf
openssl.exe "req" -new -key "qpsa_rootca.key" -x509 -out "rootca_pem.crt" -
subj /C="CN"/ST="BJ"/L="Beijing"/OU="SecBoot"/OU="QTI"/O="QCT"/CN="RootCA"
-days "7300" -set_serial "1" -sha256 -config "C:\OpenSSL\bin\openssl.cnf"
```

```
C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\OemPKCertificate>openssl.exe "req" -new -key "qpsa_rootca.key" -x509 -out "rootca
_pem.crt" -subj /C="CN"/ST="BJ"/L="Beijing"/OU="SecBoot"/OU="QTI"/O="QCT"/CN="RootCA" -days "7300" -set_serial "1" -sha2
56 -config "C:\OpenSSL-Win64\bin\openssl.cfg"
Loading 'screen' into random state - done
```

rootca_pem.crt is generated.

OEM should change below strings to identify their own information.

/C="CN" – Country

/ST="BJ"" – State

/L="Beijing – Location

/OU="SecBoot" – Purpose

/OU="QTI" – Organization Unit

/O="QCT" – Organization

/CN="RootCA" – RootCA Common Name

`-config "C:\OpenSSL-Win64\bin\openssl.cfg” – openssl configuration file in openssl bin directory`

`openssl.exe "x509" -inform "PEM" -in "rootca_pem.crt" -outform "DER" -out "qpsa_rootca.cer"`

`qpsa_rootca.cer is generated.`

Create a plain txt file name “**v3_attestCA.ext**” with below content

authorityKeyIdentifier=keyid,issuer

subjectKeyIdentifier=hash

basicConstraints=CA:TRUE

keyUsage = cRLSign, keyCertSign

`openssl.exe "genrsa" -out "qpsa_attestca.key" -3 "2048"`

```
C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\OemPKCertificate>openssl.exe "genrsa" -out "qpsa_attestca.key" -3 "2048"
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
.....................................+++
..........................+++
e is 3 (0x3)
```

qpsa_attestca.key is generated.

`openssl.exe "req" -new -key "qpsa_attestca.key" -out "attestca.csr" -subj /C="CN"/ST="BJ"/L="Beijing"/OU="QTI"/O="QCT"/CN="AttCA" – days "7300" –config "C:\OpenSSL\bin\openssl.cnf"`

```
C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\OemPKCertificate>openssl.exe "req" -new -key "qpsa_attestca.key" -out "attestca.c
sr" -subj /C="CN"/ST="BJ"/L="Beijing"/OU="QTI"/O="QCT"/CN="AttCA" -days "7300" -config "C:\OpenSSL-Win64\bin\openssl.cfg
"
Loading 'screen' into random state - done
```

attestca.CSR™ is generated.

`/CN="AttCA" – AttestationCA Common Name`

`openssl.exe "x509" -req -in "attestca.csr" -CA "rootca_pem.crt" -CAkey "qpsa_rootca.key" -out "attestca_pem.crt" -set_serial "5" -days "7300" –extfile "v3_attestCA.ext" -sha256`

```
C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\OemPKCertificate>openssl.exe "x509" -req -in "attestca.csr" -CA "rootca_pem.crt"
-CAkey "qpsa_rootca.key" -out "attestca_pem.crt" -set_serial "5" -days "7300" -extfile "v3_attestCA.ext" -sha256
Loading 'screen' into random state - done
Signature ok
subject=/C=CN/ST=BJ/L=Beijing/OU=QTI/O=QCT/CN=AttCA
Getting CA Private Key
```

attestca_pem.crt is generated .

```
openssl.exe "x509" -inform "PEM" -in "attestca_pem.crt" -outform "DER" -out
"qpsa_attestca.cer"
```

qpsa_attestca.cer is generated.

```
openssl.exe dgst -sha256 qpsa_rootca.cer >sha256rootcert.txt
```

sha256rootcert.txt is generated.

Now, OemPKCertificate generation is done, it contains below files

```
 Directory of C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\OemPKCertificate

05/03/2017  04:17 PM    <DIR>          .
05/03/2017  04:17 PM    <DIR>          ..
05/03/2017  04:04 PM               980 attestca.csr
05/03/2017  04:06 PM             1,310 attestca_pem.crt
05/03/2017  04:08 PM               925 qpsa_attestca.cer
05/03/2017  04:03 PM             1,675 qpsa_attestca.key
05/03/2017  04:03 PM               939 qpsa_rootca.cer
05/03/2017  03:34 PM             1,679 qpsa_rootca.key
05/03/2017  04:02 PM             1,326 rootca_pem.crt
05/03/2017  04:17 PM                90 sha256rootcert.txt
05/03/2017  03:06 PM               123 v3_attestCA.ext
               9 File(s)          9,047 bytes
               2 Dir(s)  69,952,258,048 bytes free
```

11.2 Generate SSD Keys

SSD key is only used for HDCP key provisioning, you can skip this step if not use HDCP.

1. Copy everything except “keys” subdirectory from WP\prebuilt\8016\app\Qwpct\projects\keys\SSDKeys to WP_SEC_BOOT\SSDKeys
2. Modify key_config.xml to replace the path string before /keys in <path> and <id_Path> to current WP_SEC_BOOT\SSDKeys. For example, the first one in my machine is

```
<path>C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\SSDKeys/keys/dvc_rsa/rsa_pkcs8
_p r_key.der</path>
```

3. Go to WP_SEC_BOOT\SSDKeys, open command window as Administrator

```
python.exe "gen_keys.py" --key_dir=keys
```

```
C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\SSDKeys>python.exe "gen_keys.py" --key_dir=keys
================================================================================
Generating AES 128 key in dvc_aes
================================================================================
================================================================================
Generating RSA 2048 Public/Private keypair in dvc_rsa
================================================================================
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
................+++
.+++
e is 65537 (0x10001)
writing RSA key
writing RSA key
writing RSA key
================================================================================
Generating RSA 2048 Public/Private keypair in oem_rsa
================================================================================
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
................+++
....................................+++
e is 65537 (0x10001)
writing RSA key
writing RSA key
writing RSA key
```

```
python.exe gen_keystore.py
```

```
C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\SSDKeys>python.exe gen_keystore.py
Reading key_id file: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\SSDKeys\keys\dvc_rsa\key_id.dat
Reading key file: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\SSDKeys\keys\dvc_rsa\rsa_pkcs8_pr_key.der
Reading key_id file: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\SSDKeys\keys\oem_rsa\key_id.dat
Reading key file: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\SSDKeys\keys\oem_rsa\rsa_pub_key.der
Reading key_id file: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\SSDKeys\keys\dvc_aes\aes_id.dat
Reading key file: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\SSDKeys\keys\dvc_aes\aes.dat
```

WP_SEC_BOOT\SSDKeys\keys\keystore.dat is generated, it will be used in factory floor if HDCP key provision required.

11.3 Generate fuse data (sec.dat)

1. Open command window, enter WP_SEC_BOOT directory

   Set environment variable SECTOOLSPATH, for example:

   ```
   set SECTOOLSPATH=C:\WP_SEC_BOOT\SECTOOLS\common\tools\sectools
   ```

2. Edit %SECTOOLSPATH%\config\8916\8916_fuseblower_USER.xml, update root cert path and several flags, changed parts are highlighted in below table. oem_product_id can be any value between 0x0001 and 0xffff.

```
<entry ignore="true">
<description>contains the OEM public key hash as set by OEM</description>
<name>root_cert_hash</name>
<value>0000000000000000000000000000000000000000000000000000000000000000</value>
</entry>
```

```
<entry ignore="false">
<description>SHA256 signed root cert to generate root hash</description>
<name>root_cert_file</name>
<value>C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\OemPKCertificate\qpsa_rootca.cer</value>
</entry>
```

```
<entry ignore="false">
<description>PK Hash is in Fuse for SEC_BOOT1 : Apps</description>
<name>SEC_BOOT1_PK_Hash_in_Fuse</name>
<value>true</value>
</entry>
```

```
<entry ignore="false">
<description>PK Hash is in Fuse for SEC_BOOT2 : MBA</description>
<name>SEC_BOOT2_PK_Hash_in_Fuse</name>
<value>true</value>
</entry>
```

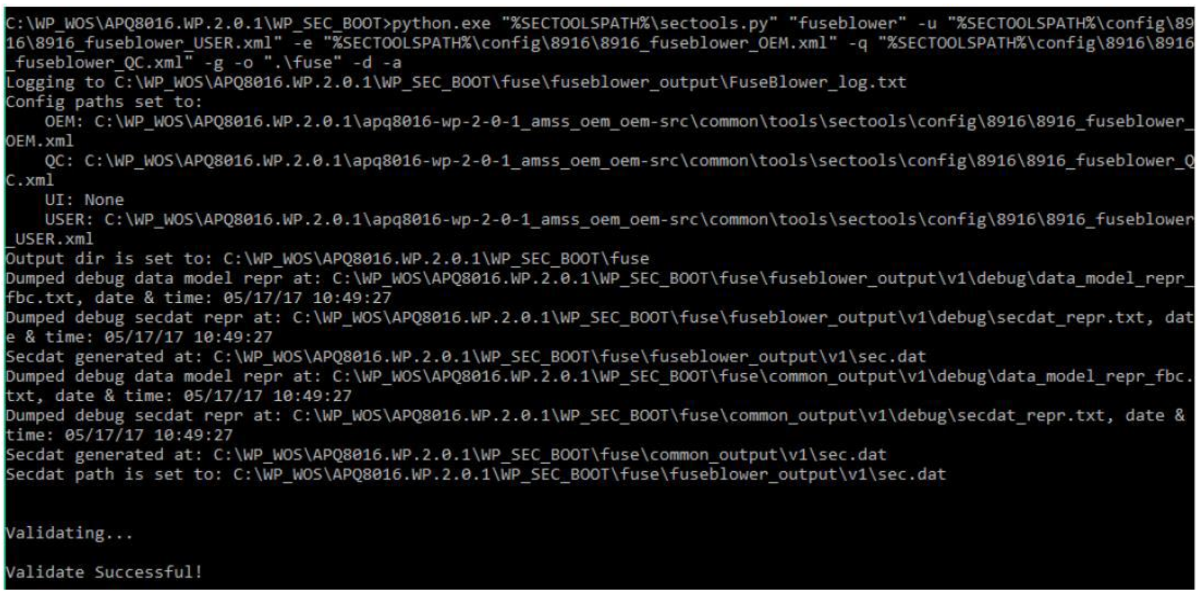
```
<entry ignore="false">
<description>PK Hash is in Fuse for SEC_BOOT3 : MPSS</description>
<name>SEC_BOOT3_PK_Hash_in_Fuse</name>
<value>true</value>
</entry>
```

```
<entry ignore="false">
<description>The OEM product ID</description>
<name>oem_product_id</name>
<value>0x0005</value>
</entry>
```

```
<entry ignore="true">
<description>if true, disable APPS debug</description>
<name>apps_debug_disabled</name>
<value>true</value>
```

3. Run below command in WP_SEC_BOOT directory, OEM can also use absolute path accordingly.

```
python.exe “%SECTOOLSPATH%\sectools.py” "fuseblower" -u
“%SECTOOLSPATH%\config\8916\8916_fuseblower_USER.xml” -e
“%SECTOOLSPATH%\config\8916\8916_fuseblower_OEM.xml” -q
“%SECTOOLSPATH%\config\8916\8916_fuseblower_QC.xml” -g -o ".\fuse" -d –a
```



```
C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT>python.exe "%SECTOOLSPATH%\sectools.py" "fuseblower" -u "%SECTOOLSPATH%\config\89
16\8916_fuseblower_USER.xml" -e "%SECTOOLSPATH%\config\8916\8916_fuseblower_OEM.xml" -q "%SECTOOLSPATH%\config\8916\8916
_fuseblower_QC.xml" -g -o ".\fuse" -d -a
Logging to C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\fuse\fuseblower_output\FuseBlower_log.txt
Config paths set to:
    OEM: C:\WP_WOS\APQ8016.WP.2.0.1\apq8016-wp-2-0-1_amss_oem_oem-src\common\tools\sectools\config\8916\8916_fuseblower_
OEM.xml
    QC: C:\WP_WOS\APQ8016.WP.2.0.1\apq8016-wp-2-0-1_amss_oem_oem-src\common\tools\sectools\config\8916\8916_fuseblower_Q
C.xml
    UI: None
    USER: C:\WP_WOS\APQ8016.WP.2.0.1\apq8016-wp-2-0-1_amss_oem_oem-src\common\tools\sectools\config\8916\8916_fuseblower
_USER.xml
Output dir is set to: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\fuse
Dumped debug data model repr at: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\fuse\fuseblower_output\v1\debug\data_model_repr_
fbc.txt, date & time: 05/17/17 10:49:27
Dumped debug secdat repr at: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\fuse\fuseblower_output\v1\debug\secdat_repr.txt, dat
e & time: 05/17/17 10:49:27
Secdat generated at: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\fuse\fuseblower_output\v1\sec.dat
Dumped debug data model repr at: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\fuse\common_output\v1\debug\data_model_repr_fbc.
txt, date & time: 05/17/17 10:49:27
Dumped debug secdat repr at: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\fuse\common_output\v1\debug\secdat_repr.txt, date &
time: 05/17/17 10:49:27
Secdat generated at: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\fuse\common_output\v1\sec.dat
Secdat path is set to: C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\fuse\fuseblower_output\v1\sec.dat


Validating...

Validate Successful!
```

The output file is WP_SEC_BOOT\fuse\fuseblower_output\v1\sec.dat, it will be load to SEC partition in factory floor.

11.4 Sign Boot Loader chain and subsystem images

1. Update flash programmer so that VIP can be disabled. Make below code change and rebuild boot_image to get new prog_emmc_firehose_8916.mbn

   File path: boot_images\core\storage\tools\deviceprogrammer\src\firehose \deviceprogrammer_initialize.c

   ```
   #define SKIP_SECBOOT_CHECK_NOT_RECOMMENDED_BY_QUALCOMM
   #ifndef SKIP_SECBOOT_CHECK_NOT_RECOMMENDED_BY_QUALCOMM
   // This check below is to ensure that only VIP programmer is run on
   secure boot devices
   // In otherwords, signing the non VIP programmer is highly not
   recommended
   if (FALSE == isValidationMode() && TRUE == isAuthenticationEnabled()) {
   strlcat(err_log, "Secure boot detected. VIP not enabled:fail ",
   sizeof(err_log)); }
   #endif
   ```

2. Download apq8016-wp-2-0-1_amss_device_precompile from Chipcode, get below mbn files, copy to WP_SEC_BOOT\firmware, need to rename some files.

   ```
   WP\prebuilt\8016\firmware\uefi.mbn -> uefi_sign_ready.mbn
   WP\prebuilt\8016\firmware\production\uefi.mbn ->
   production\uefi_sign_ready.mbn
   (OEM usually need to rebuild their own UEFI)
   boot_images\build\ms\bin\8916\prog_emmc_firehose_8916.mbn (the one
   disabled VIP, built from above step)
   boot_images\build\ms\bin\8916\sbl1.mbn
   rpm_proc\build\ms\bin\8916\rpm.mbn
   trustzone_images\build\ms\bin\MAUAANAA\tz.mbn
   trustzone_images\build\ms\bin\MAUAANAA\hyp.mbn
   winsecapp_image\build\ms\bin\BATAANAA\uefi_sec.mbn
   winsecapp_image\build\ms\bin\BATAANAA\winsecapp.mbn
   modem_proc\build\ms\bin\EAAAANGZ\mba.mbn
   modem_proc\build\ms\bin\EAAAANGZ\qdsp6sw.mbn
   wcnss_proc\build\ms\bin\8916\wcnss.mbn
   WP\prebuilt\8016\firmware\qcvss8916.mbn -> venus.mbn
   ```

3. Except prog_emmc_firehose_8916.mbn, other images can also be extracted from package files in WP\prebuilt\8016\cabfiles.

   ```
   Qualcomm.QC8916.UEFI.spkg -> 2_UEFI.bin-> uefi_sign_ready.mbn
   Qualcomm.QC8916.UEFI_PRODUCTION.spkg -> 2_UEFI.bin->
   production\uefi_sign_ready.mbn
   prog_emmc_firehose_8916.mbn will not be packaged to cab/pkg, so need to
   copy from boot_images
   Qualcomm.QC8916.SBL1.spkg -> 2_SBL1.bin -> sbl1.mbn
   Qualcomm.QC8916.RPM.spkg -> 2_RPM.bin -> rpm.mbn
   Qualcomm.QC8916.QSEE.spkg -> 2_QSEE.bin -> tz.mbn
   Qualcomm.QC8916.QHEE.spkg -> 2_QHEE.bin -> hyp.mbn
   ```

```
Qualcomm.QC8916.TZAPPS.spkg -> 2_TZAPPS.bin -> TZAPPS.bin, then get
uefi_sec.mbn
(Install freeware "ImDisk Virtual Disk Driver", mount TZAPPS.bin in a
virtual drive and get uefi_sec.mbn)
Qualcomm.QC8916.WINSECAPP.spkg ->2_WINSECA.bin-> winsecapp.mbn
Qualcomm.QC8016.AMSSPeriImage.spkg -> 2_qcdsp1v.mbn -> mba.mbn
Qualcomm.QC8016.AMSSPeriImage.spkg -> 3_qcdsp28.mbn -> qdsp6sw.mbn
Qualcomm.QC8916.WCNSSPeriImage.spkg -> 2_qcwcnss.mbn -> wcnss.mbn
Qualcomm.QC8916.qcvss.spkg -> 2_qcvss89.mbn -> venus.mbn
```

4. Continue use environment variable SECTOOLSPATH, edit %SECTOOLSPATH%\config\8916\8916_secimage.xml, change MSM_part and model_id as below, the model_id needs to be same as oem_product_id in 8916_fuseblower_USER.xml.

```
<msm_part>0x007060E1</msm_part>
<oem_id>0x0000</oem_id>
<model_id>0x0005</model_id>
<debug>0x0000000000000002</debug>
…
<image sign_id="modem" name="qdsp6sw.mbn" image_type="elf_has_ht">
<general_properties_overrides>
<sw_id>0x0000000000000002</sw_id>
</general_properties_overrides>
<pil_split>true</pil_split>
<meta_build_location>$(FILE_TYPE:download_file, ATTR:cmm_file_var,
VAR:MPSS_BINARY)</meta_build_location>
</image>
(it removes output_file_name="modem.mbn" in <image…>)
```

5. Create directory "oem_signing" under %SECTOOLSPATH%\resources\data_prov_assets\Signing\Local

   Like below one in my machine:

```
C:\WP_WOS\APQ8016.WP.2.0.1\apq8016-wp-2-0-1_amss_oem_oem-
src\common\tools\sectools\resources\data_prov_assets\Signing\Local\oem_s
igning
Copy all files in WP_SEC_BOOT\OemPKCertificate to oem_signing directory
Copy WP\prebuilt\8016\app\Qwpct\tools\security\config.xml to oem_signing
directory
```

6. Sign images, run below command under WP_SEC_BOOT directory (continue use environment variable SECTOOLSPATH).

```
python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\uefi_sign_ready.mbn" -g "uefi" -o ".\signedmbn" -p "8916" -
s --cfg_selected_cert_config=oem_signing

python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\production\uefi_sign_ready.mbn" -g "uefi" -o
".\signedmbn\production" -p "8916" -s --
cfg_selected_cert_config=oem_signing
```

```
python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\prog_emmc_firehose_8916.mbn" -g "firehose" -o ".\signedmbn"
-p "8916" -s --cfg_selected_cert_config=oem_signing

python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\sbl1.mbn" -g "sbl1" -o ".\signedmbn" -p "8916" -s --
cfg_selected_cert_config=oem_signing

python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\rpm.mbn" -g "rpm" -o ".\signedmbn" -p "8916" -s --
cfg_selected_cert_config=oem_signing

python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\tz.mbn" -g "qsee" -o ".\signedmbn" -p "8916" -s --
cfg_selected_cert_config=oem_signing

python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\hyp.mbn" -g "qhee" -o ".\signedmbn" -p "8916" -s --
cfg_selected_cert_config=oem_signing

python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\uefi_sec.mbn” -g "uefisecapp" -o ".\signedmbn" -p "8916" -s
--cfg_selected_cert_config=oem_signing

python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\winsecapp.mbn” -g "winsecapp" -o ".\signedmbn" -p "8916" -s
--cfg_selected_cert_config=oem_signing

python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\mba.mbn” -g "mba_wp" -o ".\signedmbn" -p "8916" -s --
cfg_selected_cert_config=oem_signing

python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\qdsp6sw.mbn” -g "modem" -o ".\signedmbn" -p "8916" -s --
cfg_selected_cert_config=oem_signing python.exe
"%SECTOOLSPATH%\sectools.py" "secimage" -i ".\firmware\wcnss.mbn” -g
"wcnss" -o ".\signedmbn" -p "8916" -s --
cfg_selected_cert_config=oem_signing

python.exe "%SECTOOLSPATH%\sectools.py" "secimage" -i
".\firmware\venus.mbn” -g "venus" -o ".\signedmbn" -p "8916" -s --
cfg_selected_cert_config=oem_signing
```

7. Copy signed image to WP_SEC_BOOT\signedmbn and rename.

```
WP_SEC_BOOT\signedmbn\8916\uefi\uefi_sign_ready.mbn ->
WP_SEC_BOOT\signedmbn\uefi.mbn

WP_SEC_BOOT\signedmbn\production\8916\uefi\uefi_sign_ready.mbn ->
WP_SEC_BOOT\signedmbn\production\uefi.mbn
```

```
WP_SEC_BOOT\signedmbn\8916\firehose\prog_emmc_firehose_8916.mbn ->
WP_SEC_BOOT\signedmbn\prog_emmc_firehose_8916.mbn

WP_SEC_BOOT\signedmbn\8916\sbl1\sbl1.mbn ->
WP_SEC_BOOT\signedmbn\sbl1.mbn
WP_SEC_BOOT\signedmbn\8916\rpm\rpm.mbn -> WP_SEC_BOOT\signedmbn\rpm.mbn
WP_SEC_BOOT\signedmbn\8916\qsee\tz.mbn -> WP_SEC_BOOT\signedmbn\qsee.mbn
WP_SEC_BOOT\signedmbn\8916\qhee\hyp.mbn ->
WP_SEC_BOOT\signedmbn\qhee.mbn
WP_SEC_BOOT\signedmbn\8916\ uefisecapp\ uefi_sec.mbn ->
WP_SEC_BOOT\signedmbn\uefi_sec.mbn

WP_SEC_BOOT\signedmbn\8916\winsecapp\ winsecapp.mbn ->
WP_SEC_BOOT\signedmbn\winsecapp.mbn

WP_SEC_BOOT\signedmbn\8916\mba_wp\mba.mbn ->
WP_SEC_BOOT\signedmbn\qcdsp1v28916.mbn

WP_SEC_BOOT\signedmbn\8916\modem\qdsp6sw.mbn ->
WP_SEC_BOOT\signedmbn\qcdsp28916.mbn

WP_SEC_BOOT\signedmbn\8916\wcnss\wcnss.mbn ->
WP_SEC_BOOT\signedmbn\qcwcnss8916.mbn

WP_SEC_BOOT\signedmbn\8916\venus\venus.mbn ->
WP_SEC_BOOT\signedmbn\qcvss8916.mbn
```

8. Make TZAPPS.bin

   Create TZAPPS directory under WP_SEC_BOOT\signedmbn,

   Copy below files to WP_SEC_BOOT\signedmbn\TZAPPS

   ```
   WP\prebuilt\8016\app\Qwpct\tools\security\fat_16MB.bin;
   WP\prebuilt\8016\app\Qwpct\tools\security\fatadd.py
   WP_SEC_BOOT\signedmbn\uefi_sec.mbn
   ```

Rename fat_16MB.bin to TZAPPS.bin, run below command in command window

```
fatadd.py --name=tzapps.bin --from=uefi_sec.mbn --dir=TZAPPS
```

```
C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\signedmbn\TZAPPS>fatadd.py --name=tzapps.bin --from=uefi_sec.mbn --dir=TZAPPS
Container size is 0 bytes
adding TZAPPS
Added TZAPPS
adding UEFI_SEC.MBN
Added UEFI_SEC.MBN
```

Copy TZAPPS.bin to WP_SEC_BOOT\signedmbn

If you mount TZAPPS.bin in PC with “ImDisk Virtual Disk Driver”, you will see directory tzapps and it contains uefi_sec.mbn

11.5 Packaging signed

Signed prog_emmc_firehose_8916.mbn will be used in factory process and no need to package to cab/pkg, other signed image files need to build into cab/pkg files.

1. Copy below pkg xml file from WP\prebuilt\8016\app\Qwpct\tools to WP_SEC_BOOT\pkgxml directory.

```
 Directory of C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\pkgxml

05/18/2017  05:18 PM    <DIR>          .
05/18/2017  05:18 PM    <DIR>          ..
03/22/2017  01:50 PM               666 amss.pkg.xml
03/22/2017  01:50 PM               507 qcvss.pkg.xml
03/22/2017  01:50 PM               423 qhee.pkg.xml
03/22/2017  01:50 PM               423 qsee.pkg.xml
03/22/2017  01:50 PM               420 RPM.pkg.xml
03/22/2017  01:50 PM               424 SBL1.pkg.xml
03/22/2017  01:50 PM               426 tzapps.pkg.xml
03/22/2017  01:50 PM               411 uefi.pkg.xml
03/22/2017  01:50 PM               422 uefi_production.pkg.xml
03/22/2017  01:50 PM               523 wcnss.pkg.xml
03/22/2017  01:50 PM               435 winsecapp.pkg.xml
              11 File(s)          5,080 bytes
               2 Dir(s)  615,713,136,640 bytes free
```

Edit uefi.pkg.xml, add file path \uefi.mbn

```
<Components>
<BinaryPartition ImageSource="$(CONTENT_PATH)\uefi.mbn" />
</Components>
```

Edit uefi_production.pkg.xml, add file path \production\uefi.mbn

```
<Components>
<BinaryPartition ImageSource="$(CONTENT_PATH)\production\uefi.mbn" />
</Components>
```

Edit amss.pkg.xml as below

```
<?xml version="1.0" encoding="utf-8"?>
<Package xmlns="urn:Microsoft.WindowsPhone/PackageSchema.v8.00"
Platform="QC$(QCPlatform)" Owner="Qualcomm" Component="QC$(QCChipset)"
SubComponent="AMSSPeriImage" OwnerType="SiliconVendor"
ReleaseType="Production" Partition="MainOS">
```

2. Open command window, go to WP_SEC_BOOT, set below environment and others for WP driver build, like WDKContentRoot.

NOTE: Version 2119.0.0.0 means Version Major="2119" Minor="0" QFE="0" Build="0"

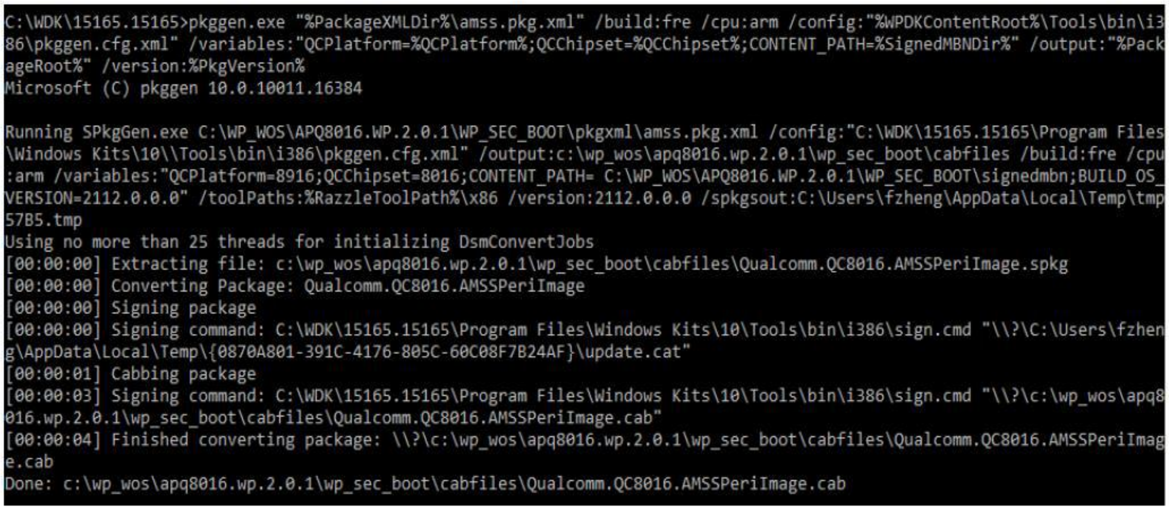
```
Call "C:\WDK\15165.15165\LaunchBuildEnv.cmd"
Set WPDKContentRoot=%WDKContentRoot%
Set SIGN_OEM=1
Set QCPlatform=8916
Set QCChipset=8016
Set PackageXMLDir=C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\pkgxml
Set SignedMBNDir= C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\signedmbn
Set PackageRoot=C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\cabfiles
Set PkgVersion="2119.0.0.0"
```

```
Call "%WPDKContentRoot%\Tools\bin\i386\installoemcerts.cmd"
```

3. Run below commands:

```
pkggen.exe "%PackageXMLDir%\SBL1.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"
/variables:"QCPlatform=%QCPlatform%;CONTENT_PATH=%SignedMBNDir%"
/output:"%PackageRoot%" /version:%PkgVersion%

pkggen.exe "%PackageXMLDir%\rpm.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"
/variables:"QCPlatform=%QCPlatform%;CONTENT_PATH=%SignedMBNDir%"
/output:"%PackageRoot%" /version:%PkgVersion%

pkggen.exe "%PackageXMLDir%\qsee.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"
/variables:"QCPlatform=%QCPlatform%;CONTENT_PATH=%SignedMBNDir%"
/output:"%PackageRoot%" /version:%PkgVersion%

pkggen.exe "%PackageXMLDir%\qhee.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"

/variables:"QCPlatform=%QCPlatform%;CONTENT_PATH=%SignedMBNDir%"
/output:"%PackageRoot%" /version:%PkgVersion%

pkggen.exe "%PackageXMLDir%\tzapps.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"
/variables:"QCPlatform=%QCPlatform%;CONTENT_PATH=%SignedMBNDir%"
/output:"%PackageRoot%" /version:%PkgVersion%

 pkggen.exe "%PackageXMLDir%\winsecapp.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"
/variables:"QCPlatform=%QCPlatform%;QCChipset=%QCChipset%;CONTENT_PATH=%
SignedMBNDir%" /output:"%PackageRoot%" /version:%PkgVersion%

pkggen.exe "%PackageXMLDir%\uefi.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"
/variables:"QCPlatform=%QCPlatform%;QCChipset=%QCChipset%;CONTENT_PATH=%
SignedMBNDir%" /output:"%PackageRoot%" /version:%PkgVersion%

pkggen.exe "%PackageXMLDir%\uefi_production.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"
/variables:"QCPlatform=%QCPlatform%;QCChipset=%QCChipset%;CONTENT_PATH=%
SignedMBNDir%" /output:"%PackageRoot%" /version:%PkgVersion%

pkggen.exe "%PackageXMLDir%\amss.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"
/variables:"QCPlatform=%QCPlatform%;QCChipset=%QCChipset%;CONTENT_PATH=%
SignedMBNDir%" /output:"%PackageRoot%" /version:%PkgVersion%
```

```
pkggen.exe "%PackageXMLDir%\wcnss.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"
/variables:"QCPlatform=%QCPlatform%;QCChipset=%QCChipset%;CONTENT_PATH=%
SignedMBNDir%" /output:"%PackageRoot%" /version:%PkgVersion%

pkggen.exe "%PackageXMLDir%\qcvss.pkg.xml" /build:fre /cpu:arm
/config:"%WPDKContentRoot%\Tools\bin\i386\pkggen.cfg.xml"
/variables:"QCPlatform=%QCPlatform%;QCChipset=%QCChipset%;CONTENT_PATH=%
SignedMBNDir%" /output:"%PackageRoot%" /version:%PkgVersion%
```



```
C:\WDK\15165.15165>pkggen.exe "%PackageXMLDir%\amss.pkg.xml" /build:fre /cpu:arm /config:"%WPDKContentRoot%\Tools\bin\i3
86\pkggen.cfg.xml" /variables:"QCPlatform=%QCPlatform%;QCChipset=%QCChipset%;CONTENT_PATH=%SignedMBNDir%" /output:"%Pack
ageRoot%" /version:%PkgVersion%
Microsoft (C) pkggen 10.0.10011.16384

Running SPkgGen.exe C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\pkgxml\amss.pkg.xml /config:"C:\WDK\15165.15165\Program Files
\Windows Kits\10\\Tools\bin\i386\pkggen.cfg.xml" /output:c:\wp_wos\apq8016.wp.2.0.1\wp_sec_boot\cabfiles /build:fre /cpu
:arm /variables:"QCPlatform=8916;QCChipset=8016;CONTENT_PATH= C:\WP_WOS\APQ8016.WP.2.0.1\WP_SEC_BOOT\signedmbn;BUILD_OS_
VERSION=2112.0.0.0" /toolPaths:%RazzleToolPath%\x86 /version:2112.0.0.0 /spkgsout:C:\Users\fzheng\AppData\Local\Temp\tmp
57B5.tmp
Using no more than 25 threads for initializing DsmConvertJobs
[00:00:00] Extracting file: c:\wp_wos\apq8016.wp.2.0.1\wp_sec_boot\cabfiles\Qualcomm.QC8016.AMSSPeriImage.spkg
[00:00:00] Converting Package: Qualcomm.QC8016.AMSSPeriImage
[00:00:00] Signing package
[00:00:00] Signing command: C:\WDK\15165.15165\Program Files\Windows Kits\10\Tools\bin\i386\sign.cmd "\\?\C:\Users\fzhen
g\AppData\Local\Temp\{0870A801-391C-4176-805C-60C08F7B24AF}\update.cat"
[00:00:01] Cabbing package
[00:00:03] Signing command: C:\WDK\15165.15165\Program Files\Windows Kits\10\Tools\bin\i386\sign.cmd "\\?\c:\wp_wos\apq8
016.wp.2.0.1\wp_sec_boot\cabfiles\Qualcomm.QC8016.AMSSPeriImage.cab"
[00:00:04] Finished converting package: \\?\c:\wp_wos\apq8016.wp.2.0.1\wp_sec_boot\cabfiles\Qualcomm.QC8016.AMSSPeriImag
e.cab
Done: c:\wp_wos\apq8016.wp.2.0.1\wp_sec_boot\cabfiles\Qualcomm.QC8016.AMSSPeriImage.cab
```

11.6 Creating FFU

1. Copy below files to "...\Packages\QCOM\prebuilt\8016\cabfiles" folder

```
Qualcomm.QC8016.AMSSPeriImage.cab
Qualcomm.QC8016.AMSSPeriImage.spkg
Qualcomm.QC8916.qcvss.cab
Qualcomm.QC8916.qcvss.spkg
Qualcomm.QC8916.QHEE.cab
Qualcomm.QC8916.QHEE.spkg
Qualcomm.QC8916.QSEE.cab
Qualcomm.QC8916.QSEE.spkg
Qualcomm.QC8916.RPM.cab
Qualcomm.QC8916.RPM.spkg
Qualcomm.QC8916.SBL1.cab
Qualcomm.QC8916.SBL1.spkg
Qualcomm.QC8916.TZAPPS.cab
Qualcomm.QC8916.TZAPPS.spkg
Qualcomm.QC8916.UEFI.cab
Qualcomm.QC8916.UEFI.spkg
Qualcomm.QC8916.WCNSSPeriImage.cab
Qualcomm.QC8916.WCNSSPeriImage.spkg
Qualcomm.QC8916.WINSECAPP.cab
Qualcomm.QC8916.WINSECAPP.spkg
```

2. Copy below file to "...\Packages\QCOM\prebuilt\8016\cabfiles\production" folder

   ```
   Qualcomm.QC8916.UEFI_PRODUCTION.cab
   Qualcomm.QC8916.UEFI_PRODUCTION.spkg
   ```

3. Open command prompt and run the 'SetupBuildEnv.cmd' file from "…\ WDK\15165.15165\BuildEnv".

4. Set below environment variables in same command prompt:

   ```
   set QCChipset=8016
   set QCPlatform=8916
   set FFU_TARGETS=SBC_Health < Flavour of FFU image >
   ```

5. After setting above variables, run the 'postbld.bat' from "...\Packages\QCOM\prebuilt\8016" folder.

NOTE: [Note : Delete or Rename the old ffu folder in ...\Packages\QCOM\prebuilt\8016".]

11.7 Enabling the Secure Boot on the Device

1. To enable the secure boot on Dragon Board, We have to blown the fuse on device using "sec.dat" file.

   To blown the fuse, the device should not be RPMB provisioned.

   You can find the "sec.dat" in "WP_SEC_BOOT\fuse\fuseblower_output\v1\sec.dat", where WP_SEC_BOOT is the folder where NHLOS images are signed.

   a. Flash **sec.dat** using mass storage mode

      ```
      emmcdl.exe -b SEC sec.dat
      ```

   b. Flash **sec.dat** file using EDL (9008) mode

      ```
      emmcdl.exe -p COMx -f prog_emmc_firehose_8916.mbn -b SEC sec.dat
      ```

2. Verify secure boot enabled :

   Put device in EDL(9008) mode, use below command to read OEM_PK_HASH, the value should be same to the value in WP_SEC_BOOT\OemPKCertificate\sha256rootcert.txt

   ```
   emmcdl.exe -p COMx -info
   ```

   ```
   emmcdl.exe -p COMx -info
   ```

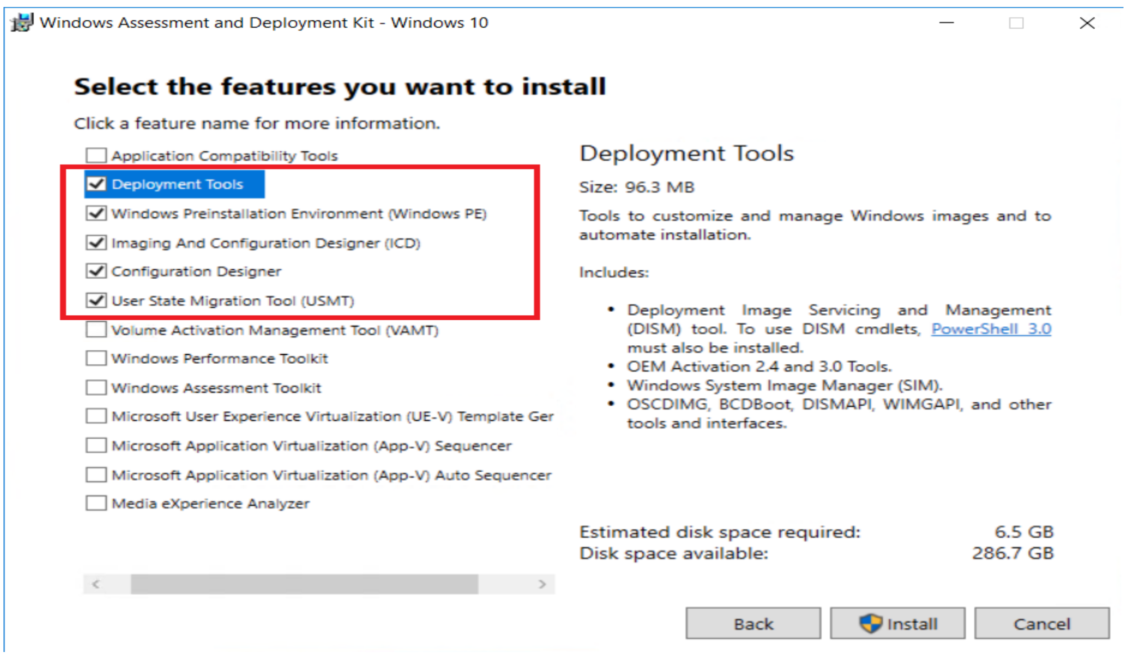
```
C:\Users\c_malik\Desktop\Ravi\Singed_ffu_2118>emmcdl.exe -p COM6 -info
Version 2.22
SerialNumber: 0x0c2b2182
MSM_HW_ID: 0x007060e1
OEM_PK_HASH: 0x1c82379ce800185a379b060f69304a2ffea5892d5b0f52634235edf10fa2d87f
SBL SW Version: 0x00000000
Status: 0 The operation completed successfully.


C:\Users\c_malik\Desktop\Ravi\Singed_ffu_2118>
```

3. Try to flash your secure image on the device.

A Appendix

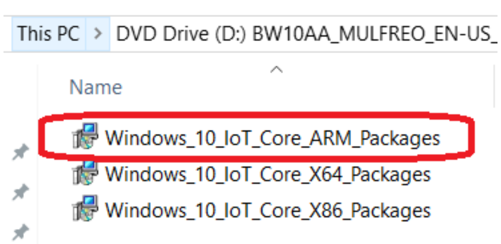
A.1 Installing ADK

The release of the installed ADK should match the Windows 10 IOT Core packages being used.When installing the ADK, install to default path and make sure that the features below are selected at minimum:



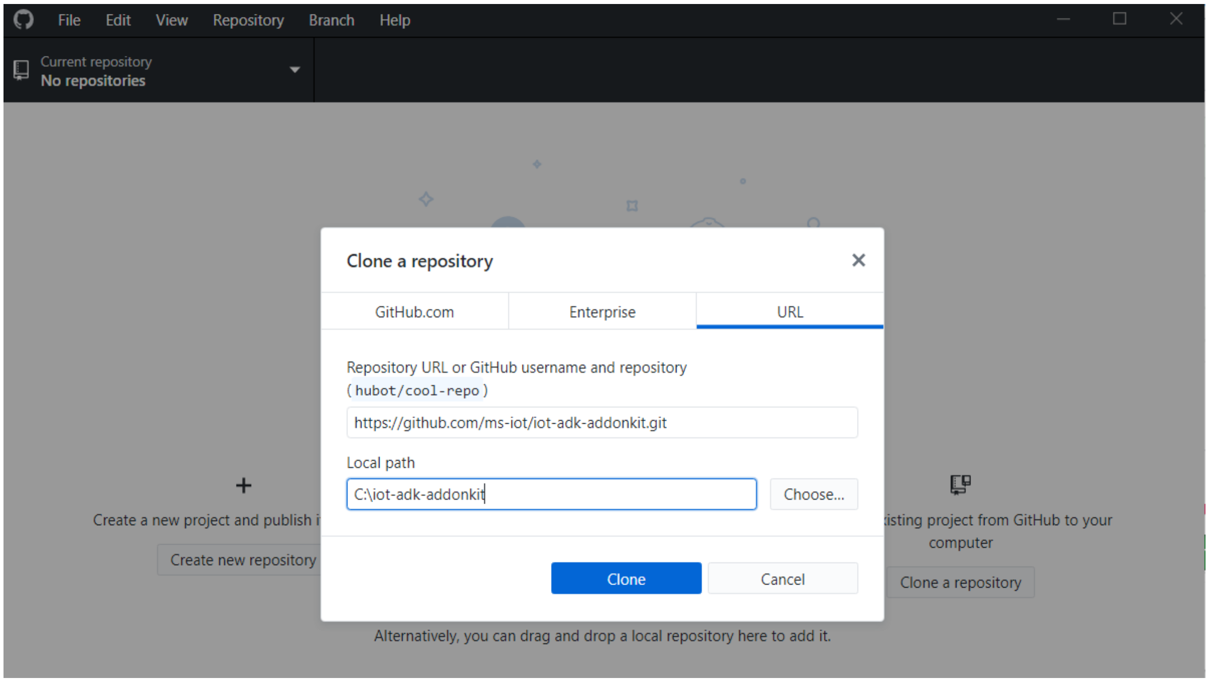
A.2 Installing Windows 10 IoT Core Packages

When installing IoT Core Packages, you only need to install the ARM, as shown below.



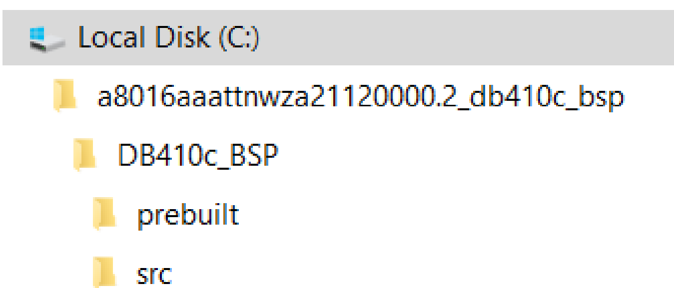
A.3 Cloning IoT ADK Addon Kit from Github

It is recommended that you use a git client (such as this one) to clone the repo at https://github.com/ms-iot/iot-adk-addonkit.



A.4 Windows 10 IoT Core Board Support Package (BSP)

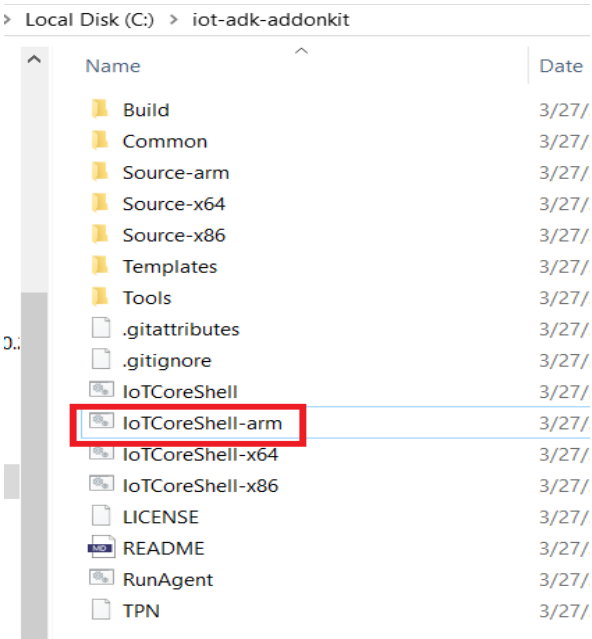
After downloading and extract the BSP zip file, the folder structure should look similar to this:



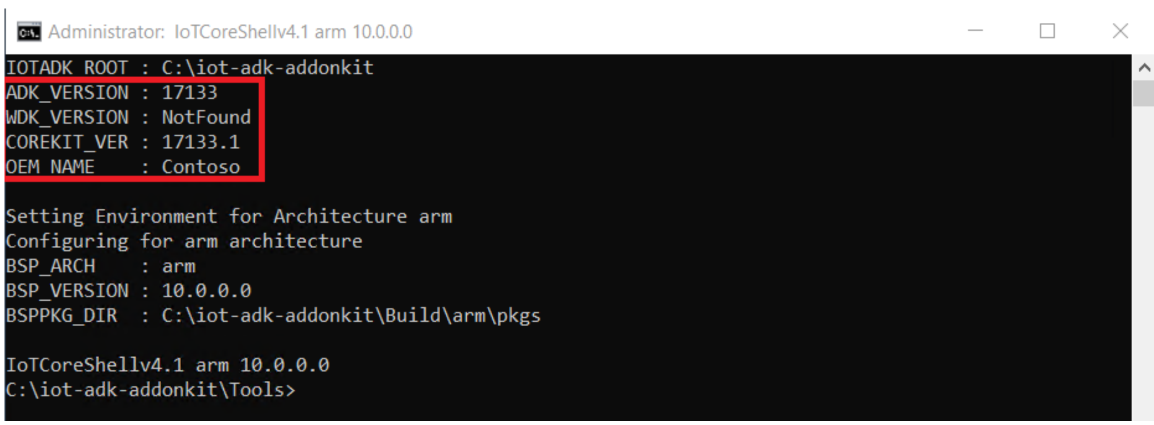
We will be extracting the CAB files from the BSP package into the IoT ADK Addon Kit build tree in the next steps.

A.5 Using IoT ADK Addon Kit to build a Test FFU

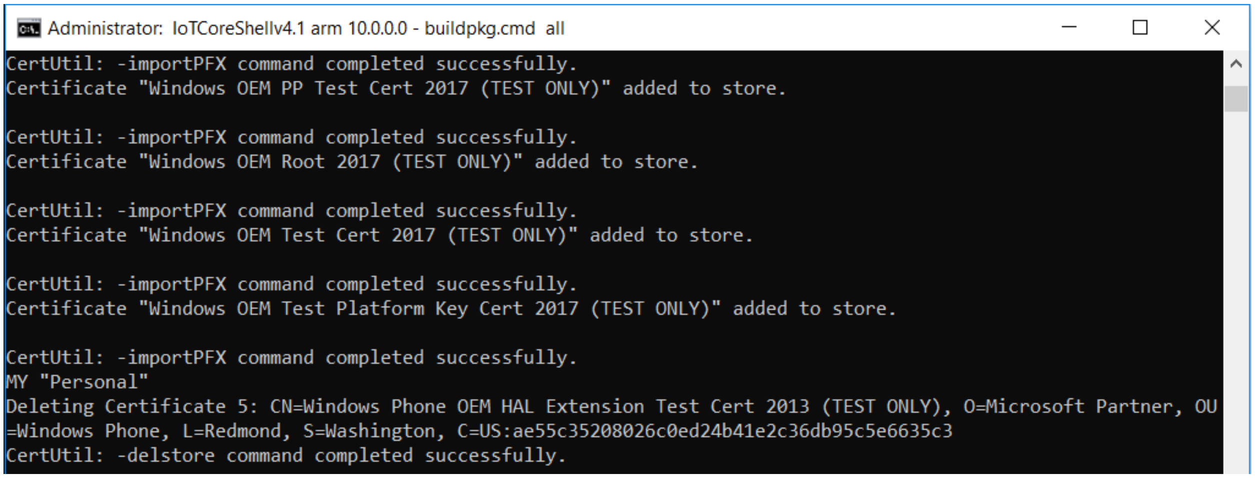
1. Edit C:\iot-adk-addonkit\Tools\setOEM.cmd and change it to the name of your company.
2. Run C:\iot-adk-addonkit\IoTCoreShell-arm.cmd



3. You should see the command window showing the ADK version and the IoT Core Package version and the OEM name you've set.

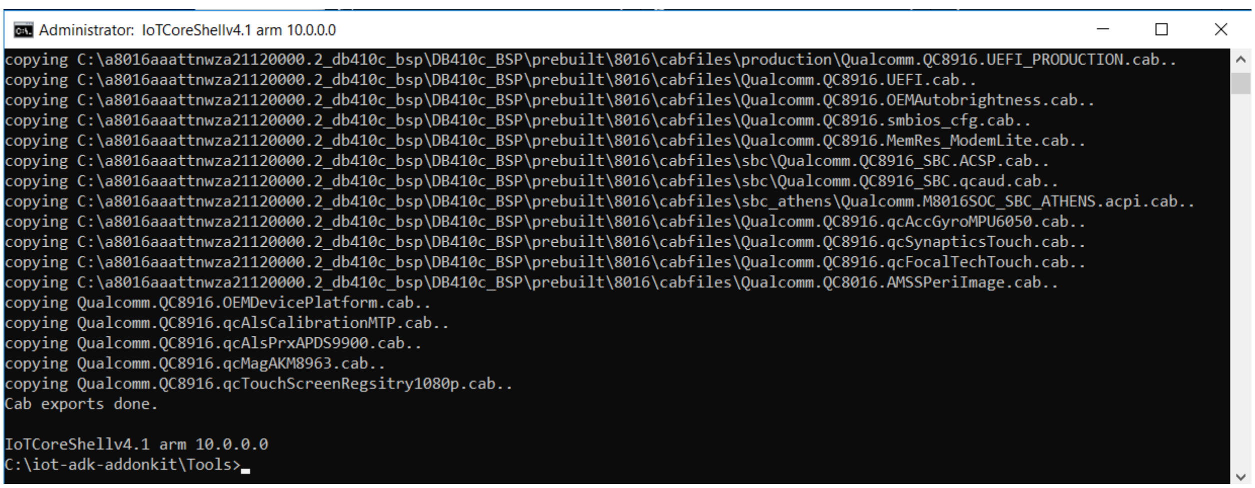


4. Install Test Sign Certificates on the system – **this only need to be run once per system.**
   Run installoemcerts.cmd



```
Administrator:  IoTCoreShellv4.1 arm 10.0.0.0 - buildpkg.cmd  all
CertUtil: -importPFX command completed successfully.
Certificate "Windows OEM PP Test Cert 2017 (TEST ONLY)" added to store.

CertUtil: -importPFX command completed successfully.
Certificate "Windows OEM Root 2017 (TEST ONLY)" added to store.

CertUtil: -importPFX command completed successfully.
Certificate "Windows OEM Test Cert 2017 (TEST ONLY)" added to store.

CertUtil: -importPFX command completed successfully.
Certificate "Windows OEM Test Platform Key Cert 2017 (TEST ONLY)" added to store.

CertUtil: -importPFX command completed successfully.
MY "Personal"
Deleting Certificate 5: CN=Windows Phone OEM HAL Extension Test Cert 2013 (TEST ONLY), O=Microsoft Partner, OU
=Windows Phone, L=Redmond, S=Washington, C=US:ae55c35208026c0ed24b41e2c36db95c5e6635c3
CertUtil: -delstore command completed successfully.
```

5. Assuming C:\a8016aaattnwza21120000.2_db410c_bsp is where the BSP was extracted to as in the screen capture above
   Run (on one line)
   C:\iot-adk-addonkit\Tools\bsptools\QCDB410C\export.cmd
   C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP
6. You should see the CAB file successfully exported from the BSP.



```
Administrator:  IoTCoreShellv4.1 arm 10.0.0.0
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\production\Qualcomm.QC8916.UEFI_PRODUCTION.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\Qualcomm.QC8916.UEFI.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\Qualcomm.QC8916.OEMAutobrightness.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\Qualcomm.QC8916.smbios_cfg.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\Qualcomm.QC8916.MemRes_ModemLite.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\sbc\Qualcomm.QC8916_SBC.ACSP.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\sbc\Qualcomm.QC8916_SBC.qcaud.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\sbc_athens\Qualcomm.M8016SOC_SBC_ATHENS.acpi.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\Qualcomm.QC8916.qcAccGyroMPU6050.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\Qualcomm.QC8916.qcSynapticsTouch.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\Qualcomm.QC8916.qcFocalTechTouch.cab..
copying C:\a8016aaattnwza21120000.2_db410c_bsp\DB410c_BSP\prebuilt\8016\cabfiles\Qualcomm.QC8016.AMSSPeriImage.cab..
copying Qualcomm.QC8916.OEMDevicePlatform.cab..
copying Qualcomm.QC8916.qcAlsCalibrationMTP.cab..
copying Qualcomm.QC8916.qcAlsPrxAPDS9900.cab..
copying Qualcomm.QC8916.qcMagAKM8963.cab..
copying Qualcomm.QC8916.qcTouchScreenRegsitry1080p.cab..
Cab exports done.

IoTCoreShellv4.1 arm 10.0.0.0
C:\iot-adk-addonkit\Tools>_
```

7. Run
   newproduct.cmd **DemoDevice** QCDB410C
   and choose to enter SMBIOS data to uniquely identify your device in SMBIOS.
   You should change **DemoDevice** to your product name.
   Example:

```
Administrator: IoTCoreShellv4.1 arm 10.0.0.0
IoTCoreShellv4.1 arm 10.0.0.0
C:\iot-adk-addonkit\Tools>newproduct.cmd DemoDevice QCDB410C
Creating DemoDevice Product with BSP QCDB410C
 Running QCDB410C specifics
 Provide SMBIOS fields
Do you want to provide SMBIOS data? [Y,N]?Y
Enter System SKU:DemoSku
Enter System Family:DeviceFamily
Enter Baseboard Product:DBClone
 SMBIOS Fields set are
 System Manufacturer: Contoso
 System Product Name: DemoDevice
 System SKU          : DemoSku
 System Family       : DeviceFamily
 Baseboard Product   : DBClone
 You can edit these fields later by changing C:\iot-adk-addonkit\Source-arm\Products\DemoDevice\SMBIOS.cfg.
DemoDevice product directories ready

IoTCoreShellv4.1 arm 10.0.0.0
C:\iot-adk-addonkit\Tools>_
```

8. Edit
   C:\iot-adk-addonkit\Source-arm\Products**DemoDevice**\oemcustomization.cmd

to configure boot time scripts, eg, change default password etc.

oemcustomization.cmd is run at every boot, and contains a section that is only run on first boot.

Interesting command line utilities

https://docs.microsoft.com/en-us/windows/iot-core/manage-your-device/commandlineutils

9. Edit

C:\iot-adk-addonkit\Source-arm\Products**DemoDevice**\prov\customizations.xml

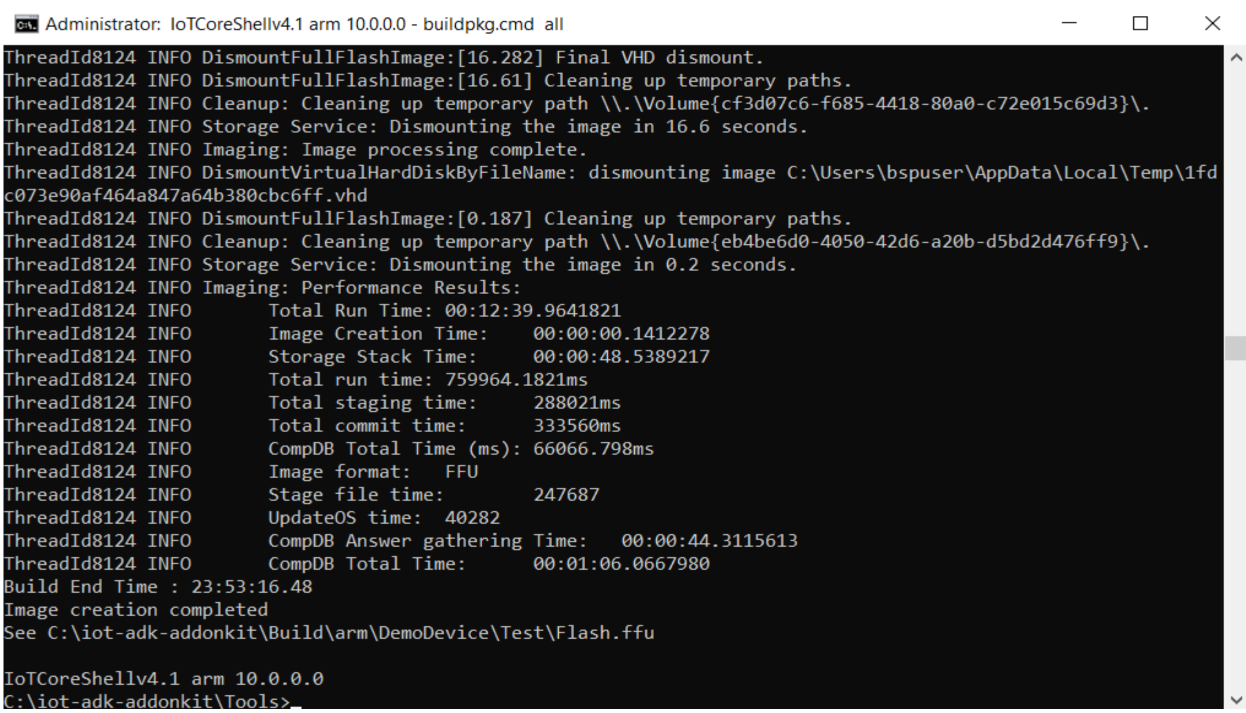
To configure policies and computer name.

10. Run
    buildpkg.cmd all

```
Administrator: IoTCoreShellv4.1 arm 10.0.0.0 - buildpkg.cmd all

IoTCoreShellv4.1 arm 10.0.0.0
C:\iot-adk-addonkit\Tools>buildpkg.cmd all
Building all provisioning packages
Processing all provisioning packages
 Processing Provisioning.Enroll
 Processing Appx.DigitalSign
 Processing Appx.IoTCoreDefaultApp
 Processing Appx.IoTOnboardingTask
 Processing DemoDevice
 Processing DigitalSign
 Processing MultiLangSample
 Processing RecoverySample
 Processing SampleA
 Processing SampleB
 Processing SecureSample
 Processing SingleLangSample
Building all packages under C:\iot-adk-addonkit\Common\Packages
 Processing Custom.BCD.wm.xml
_
```

11. Edit
C:\iot-adk-addonkit\Source-arm\Products**DemoDevice**\TestOEMInput.xml
To change the features to be included in the Test Image
see https://docs.microsoft.com/en-us/windows-hardware/manufacture/iot/iot-core-feature-list

12. To start creating a Test FFU, **unplug all USB storage device** and run
createimage.cmd **DemoDevice** Test
It may take between 10 to 30 min to build an FFU depend on your build system speed.



13. Navigate to
C:\iot-adk-addonkit\Build\arm\DemoDevice\Test
to find the Flash.ffu

See IoT Core Manufacturing Guide for additional customizations, security lock downs and building retail images.
https://docs.microsoft.com/en-us/windows-hardware/manufacture/iot/iot-core-manufacturing-guide

B Sample ACPI code

```
//
//
// SSDT (Secondary System Description Table) sample file
//  This file shows an example of writing ACPI Device entries for
//  user developed driver for user device
//


DefinitionBlock("SSDT.AML", "SSDT", 0x02, "USERS ", "MSM8016 ", 1)
{
// Should be scoped System Bus
Scope(\_SB_)
{
// An User Sample Device
Device (USRS)
{
Name (_HID, "USRS0000") Name (_UID, 1)

// define your Current Resource Settings
Method (_CRS, 0x0, NotSerialized)
{
Name (RBUF, ResourceTemplate()
{
// An end user device can consume bus resources as depicted below
// A sample entry to access I2C bus
I2CSerialBus(     // I2C0_SCL - GPIO 7 - Pin 15
// I2C0_SDA - GPIO 6 - Pin 17
0xFFFF, // SlaveAddress: placeholder
,  // SlaveMode: default to ControllerInitiated
0, // ConnectionSpeed: placeholder
,  // Addressing Mode: default to 7 bit
"\\_SB.I2C2",     // ResourceSource: I2C bus controller name
,
,
,  // Descriptor Name: creates name for offset of resource descriptor
)  // VendorData
```

```
// A sample entry to access SPI bus
SPISerialBus(     // SPI0_SCLK - GPIO 19 - Pin 8
// SPI0_MOSI - GPIO 16 - Pin 14
// SPI0_MISO - GPIO 17 - Pin 10
// SPI0_CS   - GPIO 18 - Pin 12
0, // Device selection (CS) PolarityLow,      // Device selection polarity
FourWireMode,         // wiremode
8, // databit len
ControllerInitiated,  // slave mode
4000000,                // connection speed ClockPolarityLow,       // clock
polarity ClockPhaseFirst,        // clock phase
"\\_SB.SPI5",     // ResourceSource: SPI bus controller name
,  // ResourceSourceIndex
,
,
RawDataBuffer()   // VendorData
{
0x00,  // Reserved
0x00,  // DeassertWait
0x00,  // ClockAlwaysOn
0x00,  // MXCSMode
0x00,  // HSMode
0x00,  // LoopBackMode
}
)

// A sample entry to access UART bus
UARTSerialBus(
115200, // ConnectionSpeed
,  // BitsPerByte (defaults to DataBitsEight)
,  // StopBits    (defaults to StopBitsOne)
0xC0,   // LinesInUse
,  // IsBigEndian (defaults to LittleEndian)
,  // Parity      (defaults to ParityTypeNone) FlowControlHardware,   //
FlowControl  (defaults to FlowControlNone)
0, // ReceiveBufferSize
0, // TransmitBufferSize
"\\_SB.UAR1",     // ResourceSource
,  // ResourceSourceIndex (defaults to 0)
,  // ResourceUsage   (defaults to ResourceConsumer)
,  // DescriptorName
)

// A sample entry to access Gpio PIN for general purpose data, control
operation
GpioIO(Shared, PullNone, 0, 0, IoRestrictionNone, "\\_SB.GIO0", 0,
ResourceConsumer, , ) { 36 }
```

```
// A sample entry to access Gpio PIN for configuring as an interrupt
GpioInt(Edge, ActiveBoth, Shared, PullNone, 0, "\\_SB.GIO0",){ 36 }
})
}
}
}
```

EXHIBIT 1

PLEASE READ THIS LICENSE AGREEMENT (“AGREEMENT”) CAREFULLY. THIS AGREEMENT IS A BINDING LEGAL AGREEMENT ENTERED INTO BY AND BETWEEN YOU (OR IF YOU ARE ENTERING INTO THIS AGREEMENT ON BEHALF OF AN ENTITY, THEN THE ENTITY THAT YOU REPRESENT) AND Qualcomm Technologies, Inc. (“QTI” “WE” “OUR” OR “US”). THIS IS THE AGREEMENT THAT APPLIES TO YOUR USE OF THE DESIGNATED AND/OR ATTACHED DOCUMENTATION AND ANY UPDATES OR IMPROVEMENTS THEREOF (COLLECTIVELY, “MATERIALS”). BY USING OR COMPLETING THE INSTALLATION OF THE MATERIALS, YOU ARE ACCEPTING THIS AGREEMENT AND YOU AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THESE TERMS, QTI IS UNWILLING TO AND DOES NOT LICENSE THE MATERIALS TO YOU. IF YOU DO NOT AGREE TO THESE TERMS YOU MUST DISCONTINUE AND YOU MAY NOT USE THE MATERIALS OR RETAIN ANY COPIES OF THE MATERIALS. ANY USE OR POSSESSION OF THE MATERIALS BY YOU IS SUBJECT TO THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT.

1.1 **License.** Subject to the terms and conditions of this Agreement, including, without limitation, the restrictions, conditions, limitations and exclusions set forth in this Agreement, Qualcomm Technologies, Inc. (“QTI”) hereby grants to you a nonexclusive, limited license under QTI’s copyrights to use the attached Materials; and to reproduce and redistribute a reasonable number of copies of the Materials. You may not use Qualcomm Technologies or its affiliates or subsidiaries name, logo or trademarks; and copyright, trademark, patent and any other notices that appear on the Materials may not be removed or obscured. QTI shall be free to use suggestions, feedback or other information received from You, without obligation of any kind to You. QTI may immediately terminate this Agreement upon your breach. Upon termination of this Agreement, Sections 1.2-4 shall survive.

1.2 **Indemnification.** You agree to indemnify and hold harmless QTI and its officers, directors, employees and successors and assigns against any and all third party claims, demands, causes of action, losses, liabilities, damages, costs and expenses, incurred by QTI (including but not limited to costs of defense, investigation and reasonable attorney’s fees) arising out of, resulting from or related to: (i) any breach of this Agreement by You; and (ii) your acts, omissions, products and services. If requested by QTI, You agree to defend QTI in connection with any third party claims, demands, or causes of action resulting from, arising out of or in connection with any of the foregoing.

1.3 **Ownership.** QTI (or its licensors) shall retain title and all ownership rights in and to the Materials and all copies thereof, and nothing herein shall be deemed to grant any right to You under any of QTI's or its affiliates’ patents. You shall not subject the Materials to any third party license terms (e.g., open source license terms). You shall not use the Materials for the purpose of identifying or providing evidence to support any potential patent infringement claim against QTI, its affiliates, or any of QTI’s or QTI’s affiliates’ suppliers and/or direct or indirect customers. QTI hereby reserves all rights not expressly granted herein.

1.4 **WARRANTY DISCLAIMER.** YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT THE USE OF THE MATERIALS IS AT YOUR SOLE RISK. THE MATERIALS AND TECHNICAL SUPPORT, IF ANY, ARE PROVIDED "AS IS" AND WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED. QTI ITS LICENSORS AND AFFILIATES MAKE NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE MATERIALS OR ANY OTHER INFORMATION OR DOCUMENTATION PROVIDED UNDER THIS AGREEMENT, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR AGAINST INFRINGEMENT, OR ANY EXPRESS OR IMPLIED WARRANTY ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. NOTHING CONTAINED IN THIS AGREEMENT SHALL BE CONSTRUED AS (I) A WARRANTY OR REPRESENTATION BY QTI, ITS LICENSORS OR AFFILIATES AS TO THE VALIDITY OR SCOPE OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT OR (II) A WARRANTY OR REPRESENTATION BY QTI THAT ANY MANUFACTURE OR USE WILL BE FREE FROM INFRINGEMENT OF PATENTS, COPYRIGHTS OR OTHER INTELLECTUAL PROPERTY RIGHTS OF OTHERS, AND IT SHALL BE THE SOLE RESPONSIBILITY OF YOU TO MAKE SUCH DETERMINATION AS IS NECESSARY WITH RESPECT TO THE ACQUISITION OF LICENSES UNDER PATENTS AND OTHER INTELLECTUAL PROPERTY OF THIRD PARTIES.

1.5 **LIMITATION OF LIABILITY.** IN NO EVENT SHALL QTI, QTI’S AFFILIATES OR ITS LICENSORS BE LIABLE TO YOU FOR ANY INCIDENTAL, CONSEQUENTIAL OR SPECIAL DAMAGES, INCLUDING BUT NOT LIMITED TO ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL DAMAGES, ARISING OUT OF THE USE OR INABILITY TO USE, OR THE DELIVERY OR FAILURE TO DELIVER, ANY OF THE MATERIALS, OR ANY BREACH OF ANY OBLIGATION UNDER THIS AGREEMENT, EVEN IF QTI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE FOREGOING LIMITATION OF LIABILITY SHALL REMAIN IN FULL FORCE AND EFFECT REGARDLESS OF WHETHER YOUR REMEDIES HEREUNDER ARE DETERMINED TO HAVE FAILED OF THEIR ESSENTIAL PURPOSE. THE ENTIRE LIABILITY OF QTI, QTI’s AFFILIATES AND ITS LICENSORS, AND THE SOLE AND EXCLUSIVE REMEDY OF YOU, FOR ANY CLAIM OR CAUSE OF ACTION ARISING HEREUNDER (WHETHER IN CONTRACT, TORT, OR OTHERWISE) SHALL NOT EXCEED US$10.

2. **COMPLIANCE WITH LAWS; APPLICABLE LAW.** You agree to comply with all applicable local, international and national laws and regulations and with U.S. Export Administration Regulations, as they apply to the subject matter of this Agreement. This Agreement is governed by the laws of the State of California, excluding California’s choice of law rules.

3. **CONTRACTING PARTIES.** If the Materials are downloaded on any computer owned by a corporation or other legal entity, then this Agreement is formed by and between QTI and such entity. The individual accepting the terms of this Agreement represents and warrants to QTI that they have the authority to bind such entity to the terms and conditions of this Agreement.

4. **MISCELLANEOUS PROVISIONS.** This Agreement, together with all exhibits attached hereto, which are incorporated herein by this reference, constitutes the entire agreement between QTI and You and supersedes all prior negotiations, representations and agreements between the parties with respect to the subject matter hereof. No addition or modification of this Agreement shall be effective unless made in writing and signed by the respective representatives of QTI and You. The restrictions, limitations, exclusions and conditions set forth in this Agreement shall apply even if QTI or any of its affiliates becomes aware of or fails to act in a manner to address any violation or failure to comply therewith. You hereby acknowledge and agree that the restrictions, limitations, conditions and exclusions imposed in this Agreement on the rights granted in this Agreement are not a derogation of the benefits of such rights. You further acknowledges that, in the absence of such restrictions, limitations, conditions and exclusions, QTI would not have entered into this Agreement with You. Each party shall be responsible for and shall bear its own expenses in connection with this Agreement. If any of the provisions of this Agreement are determined to be invalid, illegal, or otherwise unenforceable, the remaining provisions shall remain in full force and effect. This Agreement is entered into solely in the English language, and if for any reason any other language version is prepared by any party, it shall be solely for convenience and the English version shall govern and control all aspects. If You are located in the province of Quebec, Canada, the following applies: The Parties hereby confirm they have requested this Agreement and all related documents be prepared in English.