



Qualcomm Linux Boot Guide

80-70018-4 AC

April 30, 2025

Contents

1	Boot overview	3
2	Boot architecture	4
2.1	Boot loader	4
3	APSS cold boot flow	7
4	RTSS cold boot flow	9
4.1	RTSS bootup	9
5	Update and recovery	11
5.1	Update device firmware	11
5.2	UEFI variables involved in capsule update	12
5.3	EFI system resource table for capsule update status from HLOS	13
5.4	Authentication and signing	16
6	Capsule update and trailboot rollback feature in base and advance variants	18
6.1	Base variant	21
6.2	Advance variant	22
6.3	Capsule update triggering	22
7	Interfaces	23
7.1	Platform information	24
7.2	Boot device tree	24
7.3	PMIC	27
7.4	TLMM	46
7.5	Buses	48
7.6	USB	50
7.7	Storage	53
8	Tools to run the boot process	54
9	Develop UEFI applications	56
10	Configure QDTE	57

11 Debug logs	63
11.1 PBL/XBL log	63
11.2 UEFI log	68
11.3 Linux log	74
11.4 Capsule update logs	75
12 Examples on debugging scenarios	94
12.1 Enumeration of EDL device manager	94
12.2 Detection of RAM dump	94
13 References	96

1 Boot overview

A boot process requires the Qualcomm® Linux® system software image to be launched onto a device, following a predefined set of instructions stored in the read-only memory (ROM).

This Linux software is responsible for loading and running the operating system, and configuring the devices after the required software has been initialized. When this process is complete, the file systems are mounted and ready for use.

To initiate an operating system, a boot software image is loaded when the machine is powered on. For information about setting up the Qualcomm Linux boot image, see [Build Boot](#) in the build firmware section.

This guide explains the supported boot features, the architecture of a cold boot (which starts from a power-off state), and the interfaces for the Qualcomm Linux system software.

Additionally, this guide provides instructions on how to develop a unified extensible firmware interface (UEFI) application, configure the device tree (DT) [interfaces](#), debug logs, and verify system bootup. For more information about UEFI, see UEFI.

Note: See [Hardware SoCs](#) that are supported for Qualcomm Linux.

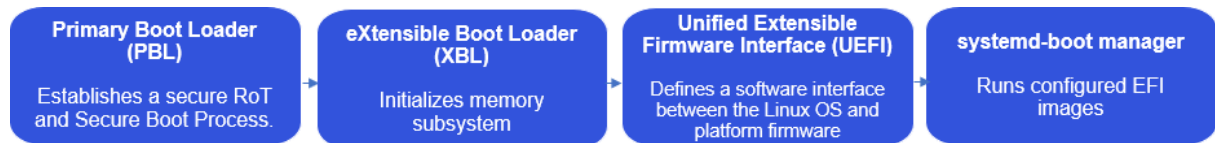
2 Boot architecture

The boot architecture uses boot loaders and board support package (BSP) images at each stage of boot. The boot loaders manage different stages of the boot process until they enable the Linux kernel and applications.

2.1 Boot loader

The system runs the boot loader software when powered on, serving as an interface for loading the operating system and other required applications.

Qualcomm chipsets, including the ones supported by Qualcomm Linux, use a multistage boot process as follows:



Primary boot loader (PBL)

- Establishes a secure root-of-trust (RoT) and secure boot process for applications. For more information, see [Secure boot](#).
- Identifies the primary storage device and loads the secondary boot loader called the eXtensible Boot Loader (XBL).

If the system encounters any recoverable error while loading the XBL image, the PBL enters the emergency download (EDL) mode. For more information about how to detect EDL mode, see [Enumeration of EDL device manager](#).

- Loads XBL segments into the boot SoC internal memory. For more information on how PBL loads the XBL segments, see cold boot architecture.

eXtensible Boot Loader (XBL)

- Initializes the hardware, firmware images, CPU cache, MMU, boot device, PMIC, and DDR.
- Sets up the RAM dump USB driver, USB charging, thermal check, power management integrated circuit (PMIC), and low-power double data rate (LPDDR) clock functions.
- Collects the RAM dump over USB onto the host PC.
- Loads and verifies the Qualcomm® Trusted Execution Environment (Qualcomm TEE), Qualcomm Hypervisor, and UEFI image.
- Provides the XBL configuration (XBL_CFG), which is part of the cold boot flow and includes PMIC and other driver settings.
- Provides XBL with a Qualcomm-signed ELF segment to initialize the external protection units (xPU).
- XBL_CFG is a standalone binary with platform-specific configurations and settings. The XBL uses the XBL_CFG driver to load and configure the required settings. The driver uses the binary to provide on-demand read access to each setting of XBL_CFG.

Unified extensible firmware interface (UEFI)



UEFI is the software interface between an operating system (OS) and the platform firmware.

It includes data tables with platform information, along with the boot and runtime service calls that the OS and its loader can use. Together, they create a standard environment for booting an operating system and running UEFI applications.

Qualcomm uses Tianocore EDK2, an open-source implementation available at <http://www.tianocore.org/edk2/>, which follows the [UEFI specification](#).

The system offers two build options, which are:

- Retail build: Has minimal debug features and an optimal memory footprint, making it ideal for production environments.
- Debug build: Ideal for development environments, with all debug features enabled.

For developing a UEFI application, see [Unified Extensible Firmware Interface \(UEFI\)](#).

systemd-boot OS manager

systemd-boot is a UEFI boot manager which executes configured EFI images. It supports systems with UEFI firmware only and does the following:

- Loads boot entry information from the EFI system partition (ESP), typically mounted at `/efi`, `/boot`, or `/boot/efi` during the Linux OS runtime and from the extended boot loader partition (XBOOTLDR) (mounted to `/boot`).

You must locate the configuration file fragments, kernels, initial RAM disk (initrd), and other EFI images within the ESP or XBOOTLDR.

- Reads simple and generic boot loader configuration files, and selects one file per boot loader entry. All the files are in the ESP partition.

To ensure that the Qualcomm Linux kernel can be directly run as an EFI image, use the `CONFIG_EFI_STUB` compilation option.

For more information about:

- EFI boot stub, see <https://docs.kernel.org/admin-guide/efi-stub.html>.
- For a sample structure of the EFI partition, see [Systemd-boot in Qualcomm Linux Yocto Guide](#).

3 APSS cold boot flow

A cold boot starts the system from the power-off state. This process begins with the PBL.

The following figure shows the cold boot flow.

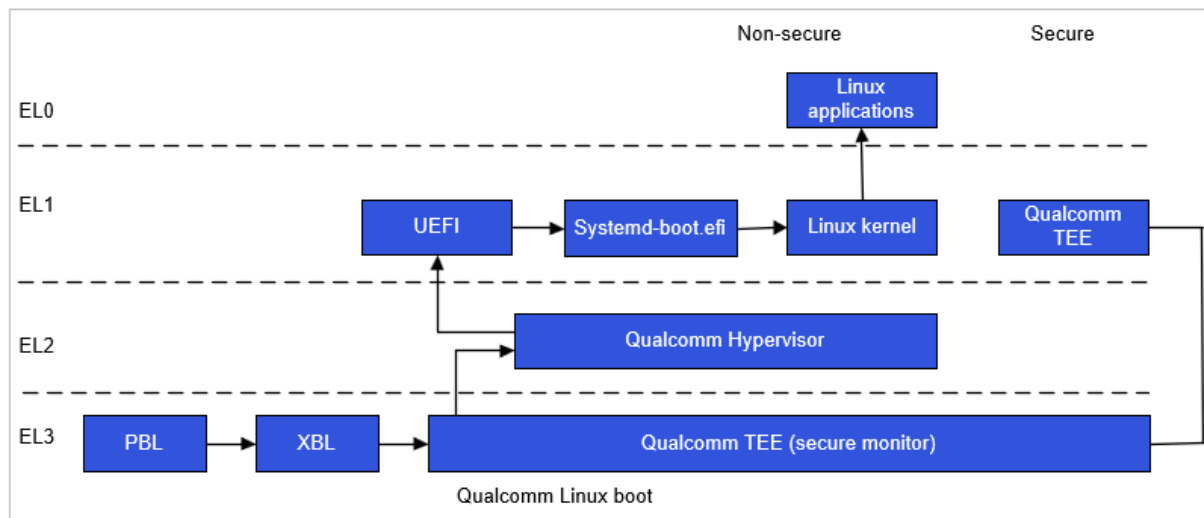


Figure: Cold boot flow

The cold boot process operates in two security modes: Secure and Nonsecure. At the EL0, EL1, and EL2 [exception levels](#), the boot core can operate in either Secure or Nonsecure mode, however at the EL3 it always operates in Secure mode. The Linux kernel runs in the nonsecure EL1 state. After completing the security settings in the secure EL3 state, the XBL may switch to the UEFI nonsecure EL1 state.

For more information about the security modes, see [Software Security Architecture](#).

The cold boot sequence involves the following steps:

1. After a reset, the boot core exits the Reset state and executes the PBL. The PBL initializes the hardware clocks, CPU caches, memory management unit (MMU), and identifies the boot device based on the boot option settings.
2. The PBL loads and authenticates the XBL from the boot device. The XBL runs the security setup in EL3 Secure state and performs the following tasks:
 - a. Initializes the hardware, firmware images, CPU cache, MMU, boot device, PMIC, and

DDR.

- b. Loads and authenticates the Qualcomm TEE image from the boot device.
 - c. Loads and authenticates the Qualcomm Hypervisor image.
 - d. Loads and authenticates the UEFI image.
 - e. Makes a secure channel manager (SCM) call to jump to the Qualcomm TEE image.
SCM is the driver that communicates with Qualcomm TEE using a [secure monitor call \(SMC\)](#).
3. The Qualcomm TEE sets up the secure environment and runs the Qualcomm Hypervisor image.
4. The Qualcomm Hypervisor hands over control to UEFI, which then loads and authenticates the systemd-boot EFI image using the [UEFI secure boot](#).
5. The systemd-boot image loads and authenticates the Qualcomm Linux kernel image, and passes control to the Qualcomm Linux kernel. If you enable kernel-based virtual machine (KVM) mode, the UEFI shuts down the Qualcomm Hypervisor, exits the UEFI boot services, and passes control to Linux KVM in EL2.
6. The Qualcomm Linux kernel launches the Linux application such as the bash shell.

For Real time subsystem (RTSS) cold boot flow, which runs independently and in parallel to the APSS boot flow, see [RTSS cold boot flow](#).

4 RTSS cold boot flow

Note: This boot flow is applicable to QCS9075 and QCS8275.

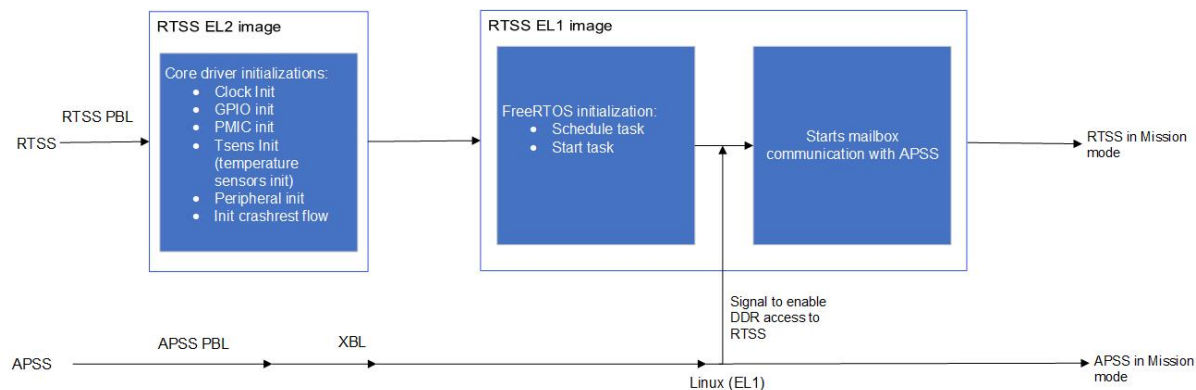
The Real Time Subsystem (RTSS) is designed to deliver high real-time performance through an independent, highly integrated architecture that significantly enhances computing, networking, and safety support. RTSS embodies the “SoC-within-an-SoC” concept, consisting of two domains:

- APSS
- RTSS

In the realm of IoT, RTSS is currently employed for bootup, reset use cases, dump collection, low-power mode, real-time application development, and more.

4.1 RTSS bootup

The RTSS boots up in parallel with the APSS boot flow. The RTSS operates as a separate subsystem and supports the Free Real Time Operating System (FreeRTOS). In the current release, [FreeRTOS](#) is enabled as a binary-only release with functionality limited to booting up the device and initializing clocks and peripherals in IoT applications.



The cold boot sequence in RTSS involves the following steps:

1. Reset state exit: The boot core exits the reset state and begins executing the RTSS PBL.

2. PBL execution:

- Initializes the PLLs and clocks.
- Loads and authenticates the RTSS image (boot code in the EL2 layer and FreeRTOS in the EL1 layer).
- Jumps to the RTSS EL2.

3. EL2 image initialization:

- Initializes core drivers such as clock, GPIO, temperature monitoring, voltage monitoring, UART, I²C, etc.
- Initializes crash dump collection.
- Hands over control to RTSS EL1.

4. EL1 image initialization:

- Completes FreeRTOS initialization to schedule and start tasks.
- Initializes dedicated peripherals like QUP (I²C, SPI, UART).
- Shares DDR memory between the main domain and RTSS. Ensure mailbox software drivers on both APSS and RTSS use the dedicated DDR shared memory space for communication between APSS and RTSS.

For information about flashing RTSS, see: https://docs.qualcomm.com/bundle/publicresource/topics/80-70018-254/flash_images.html#flash-sail.

5 Update and recovery

The firmware on a device is updated using a capsule, through the UEFI. This process involves encapsulating the firmware update payload in a capsule [.cap file], which is then processed by the system firmware to update the device firmware.

Note: The mentioned capsule update is applicable to Qualcomm Linux advanced and custom variants only.

5.1 Update device firmware

Updating device firmware involves installing new software onto a device's firmware, which is the low-level software that controls the hardware. This process can improve the device's performance, fix bugs, add new features, or enhance security. To update the device firmware through a capsule, do the following:

1. **Preparation:** The user creates a capsule containing the firmware update.
2. **Staging:** The capsule is copied to the `/EFI/UpdateCapsule/<capsule>.cap` file in EFI system partition (ESP), before OS triggers capsule update.
3. **Reboot:** The OS triggers the capsule update and restarts the system to enter the UEFI environment.
4. **Update Initiation:** The UEFI firmware detects the capsule and starts the update process.
5. **Verification:** The firmware checks the capsule integrity and authenticity.
6. **Installation:** The firmware is updated using the capsule data.
7. **Completion:** The system reboots again to complete the update.

In this process, the firmware update payload is encapsulated in a .cap file. This capsule (.cap) file is then processed to ensure secure and reliable firmware updates.

To generate a capsule, see capsule generation in uefi.

5.2 UEFI variables involved in capsule update

Communication between the OS and UEFI uses protocols and services to interact with platform firmware. UEFI offers a standard boot environment, data tables with platform information, and boot/runtime service calls for the OS loader and OS.

UEFI variables used in capsule update

Variable name	Description
OsIndications	This variable is owned by the OS and is used to indicate the features the OS wants the firmware to enable or the OS wants the firmware to take. This variable is set by OS and cleared by UEFI.
OsIndicationsSupported	This variable is owned by the firmware and indicates which of the OS indication features and actions the firmware supports.
OsTrialBootStatus - Is a 32 bit UEFI variable.	Bit map: <ul style="list-style-type: none"> • 7:0 - Version - Version number of OsTrialBootStatus • 11:8 - TrialBootMaxCount - This is set by the OS before triggering the capsule update. • 15:12 - TrialBootCount - This is set/incremented by UEFI (if TrialBootEnabled isn't cleared by OS). The OS can set to TrialBootMaxCount, if the OS wants to trigger a firmware rollback. • 16 - TrialBootEnabled - This is set to 1 by UEFI after capsule update and cleared by the OS after successful OTA update. • 31:17 - Unused
EFI system resource table (ESRT)	The ESRT table is set by the UEFI after the capsule update.

For more information about the OsIndications and OsIndicationsSupported variables, see https://uefi.org/specs/UEFI/2.10/08_Services_Runtime_Services.html?highlight=osindications#exchanging-information-between-the-os-and-firmware.

Note:

- If there is more than one capsule in the ESP partition (in EFI/UpdateCapsule), consider all the capsules for the update and update them alphabetically.

- Store all UEFI non-volatile variables in the rollback protection memory block (RPMB) partition, which must be in a Provisioned state.
 - Autoprovision the RPMB on Qualcomm secure-boot enabled devices.
 - On Qualcomm non-secure devices, don't autoprovision the RPMB; instead, provision it with test keys.
 - If you provision the RPMB with test keys on non-secure devices, you can't re-provision it with a device key when enabling Qualcomm secure boot on the device.
 - Use the `rpmbClient` application for RPMB provisioning from Linux. For more information, see [Qualcomm Security Linux Guide](#).

5.3 EFI system resource table for capsule update status from HLOS

The EFI System Resource Table (ESRT) is an optional mechanism for identifying the device and the system firmware resources to target firmware updates. Each ESRT entry describes a firmware resource that can be updated and reports the status of the last attempted update. The entries in ESRT, which are used to report status of the last attempted update are:

Table: ESRT entries

ESRT field	Description
FwClass	The firmware class field contains a GUID that identifies a firmware component that can be updated through <i>UpdateCapsule()</i> .
FwType	Identifies the type of firmware resource.
FwVersion	The firmware version field represents the current version of the firmware resource. The value must always increase as a larger number represents a newer version.
LowestSupportedFwVersion	The lowest firmware resource version to which a firmware resource can be rolled back for the specified system/device.
CapsuleFlags	The capsule flag field contains the <i>CapsuleGuid</i> flags (bits 0-15) as defined in the <i>EFI_CAPSULE_HEADER</i> that's set in the capsule header.
LastAttemptVersion	The field describes the last firmware version for which an update was attempted (uses the same format as the firmware version).
LastAttemptStatus	The field describes the result of the last firmware update attempt for the firmware resource entry.

The ESRT content is exposed by the OS from the sysfs path.

```

/sys/firmware/efi/esrt/entries/entry0/
sh-5.1# cd /sys/firmware/efi/esrt/entries/entry0/
sh-5.1# ls
capsule_flags  fw_type      last_attempt_status  lowest_supported_fw_
version
w_class        fw_version  last_attempt_version

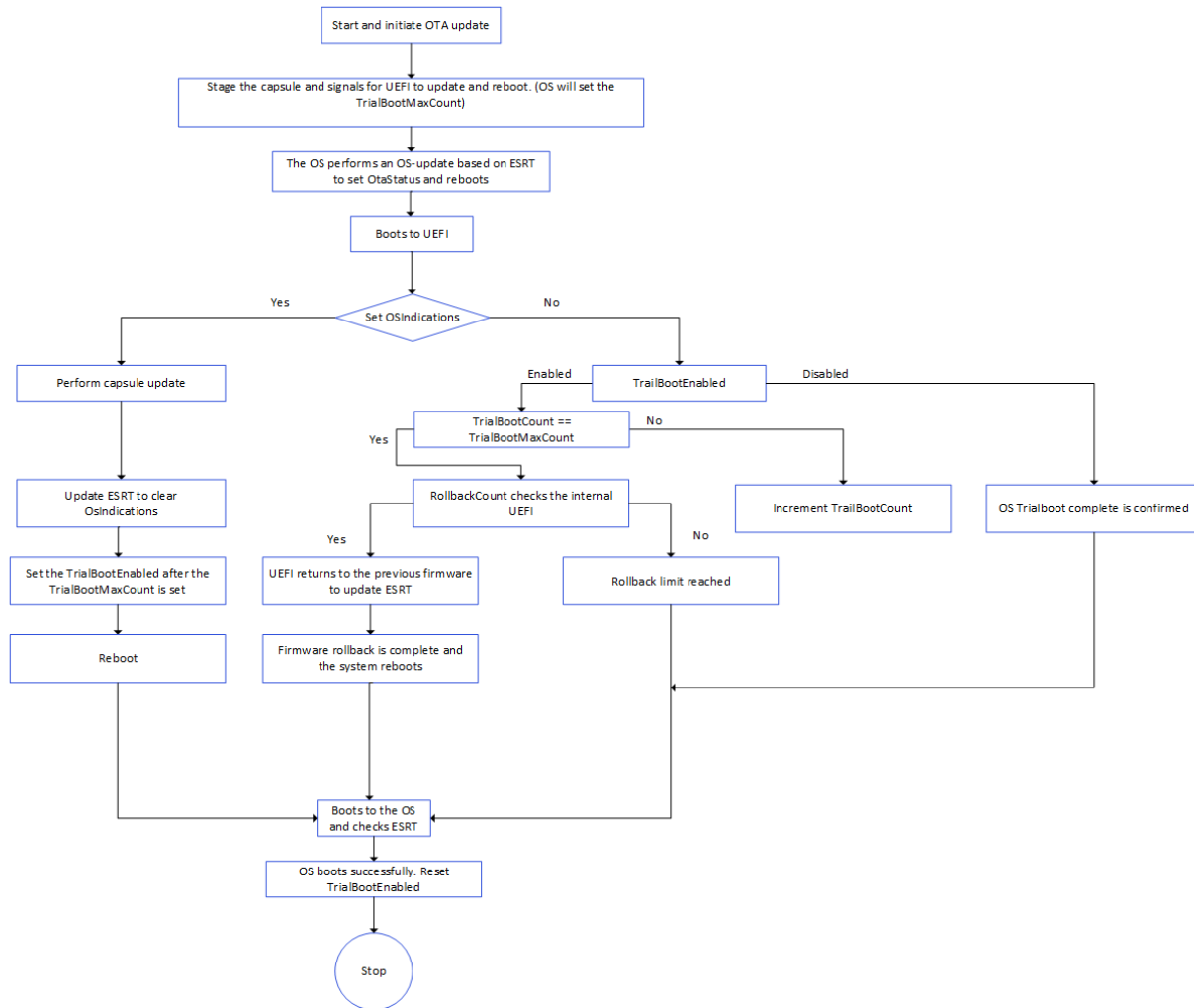
```

ESRT table is set by UEFI after the capsule update. X is the firmware version before the capsule update. Y is the version of the firmware, which is available in the capsule [.cap] file.

Last attempted firmware version	Current firmware version	last_attempt status	Capsule update scenario
Y	Y	0	Update success
Y	X	1	Capsule update failure
0	X	0	Capsule meta-data failure
Y	X	0	Firmware rollback done

For more information about ESRT, see [Firmware Update and Reporting - UEFI Specification 2.10 documentation](#)

The flowchart explains how the device firmware is updated using the UEFI capsule solution/method using different UEFI variables along with the rollback support.



1. **Initiation:** The over the air (OTA) update process starts, and the OS prepares to update the capsule, signaling the UEFI (through setting `OsIndications`) to begin the update. The OS sets a `TrialBootMaxCount` to track the number of trial boots allowed and then reboots the device.
2. **UEFI update:** Upon reboot, the system checks for `OsIndications`. If present, it performs the capsule update, updates the EFI system resource table (ESRT), clears `OsIndications`, and if `TrialBootMaxCount` is set, allows `TrialBootEnabled` and restarts the system again.
3. **Trialboot handling:**
 - If the `OsIndications` aren't set, the system checks if the `TrialBootEnabled` is active.
 - If the `TrialBootEnabled` is active and the `TrialBootCount` equals the `TrialBootMaxCount`, it checks the rollback count.

- If a rollback is allowed, the UEFI reverts to the firmware, updates the ESRT, and reboots.
 - If a rollback isn't allowed, it indicates that the rollback limit is reached.
 - If the `TrialBootCount` is less than the `TrialBootMaxCount`, it increments the `TrialBootCount` and continues the process.
4. **Completion:** The system boots to the OS, checks the ESRT, and if the OS boots successfully, it resets the `TrialBootEnabled` field.
5. **Result:**
- If `TrialBootEnabled` isn't active, the OTA update is confirmed as complete.
 - The system clears the `OtaTrialBootStatus` and sets the `LwSupportedFwVersion` to the current firmware version to disable rollback.
6. **Recovery:** If a firmware capsule update fails during the update process, UEFI has a recovery mechanism to maintain the stability and functionality.
- a. **Rollback:** If a rollback is allowed, UEFI reverts to the previous firmware version. This involves restoring the firmware to the state before the update attempt.
 - b. **ESRT update:** The ESRT is updated to reflect the rollback.
 - c. **Reboot:** After the rollback is complete, the UEFI triggers a restart to ensure it's running the previous, stable firmware version.

This process ensures that the update is applied and allows for rollback if any issues are encountered during the trial period.

5.4 Authentication and signing

Authentication is essential for firmware updates to ensure security and stability. It guarantees that only updates from trusted sources are applied, preventing malicious or unauthorized updates. By verifying the digital signature of the capsule, it ensures the update hasn't been tampered with during transmission and that only verified and tested updates are installed. This reduces the risk of system crashes or malfunctions due to faulty firmware.

To authenticate capsule-based system firmware updates using the instructions from the [Tianocore GitHub](#) - , follow these steps:

1. **Generate signing keys:**
 - Use OpenSSL command line utilities to create a new self-signed X.509 certificate chain. This involves generating a private key and a corresponding public certificate.
 - The private key is used to sign the firmware update capsules, while the public certificate is used by the UEFI firmware to verify the signature.

2. Sign the firmware update capsule:

- Use the signing keys to sign the firmware update capsule. This ensures that the capsule is authenticated and hasn't been tampered with.
- The signing process typically involves creating a hash of the firmware update payload and then encrypting this hash with the private key to create a digital signature.
- Place these certificates in a folder named `Certificates`. Sample files available in this folder might include `QcFMPCert.pem`, `QcFMPCert.pub.pem`, and `QcFMPCertSub.pub.pem`.

3. Verify the capsule:

- The UEFI firmware uses the public certificate to verify the digital signature on the capsule. If the signature is valid and matches the trusted certificate, the firmware update process will proceed.
- `QcFMPCert.pub.pem` (or `NewRoot.cer`) should be provided in the boot DT at `/sw/uefi/uefiplat/QcCapsuleRootCert`.

This verification step ensures that only authorized firmware updates are applied to the system.

6 Capsule update and trailboot rollback feature in base and advance variants

CapsuleUpdate: Seamlessly updates system firmware or software without disrupting functionality.

TrialBoot-Rollback

Capsule found	RPMB Provision	OsIndication variable present	Os TrialBoot variable present	Status of capsule update feature	Availability of TrialBoot and RollBack feature	Applicable variant	Notes
Yes	Yes	No	No	Yes	No	(Standard) + upstream distro	Base

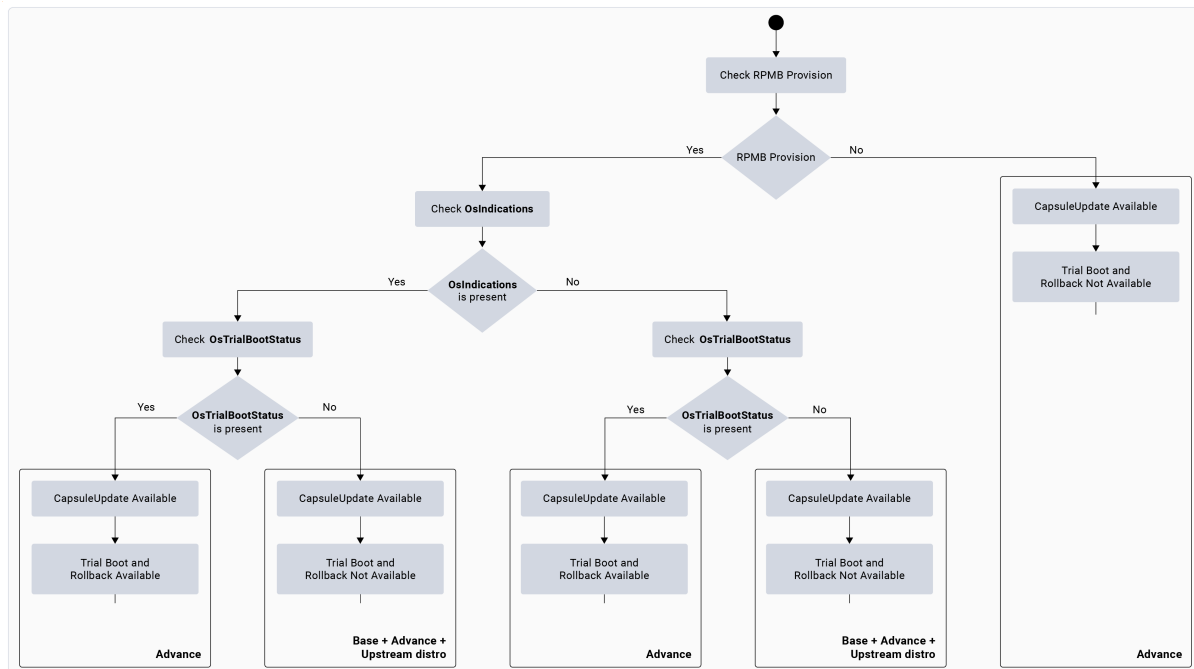
Yes	Yes	No	Yes	No	No	Advance	RPMB partition needs to be erased by OS when switching from Advance to Base Variant on same device. To erase RPMB partition, run the following command.
							<code>fastboot oem rpmb_erase</code>
Yes	Yes	Yes	No	Yes	No	Advance	

Yes	Yes	Yes	Yes	Yes	Yes	Advance	<div>1. uefi_sec linux app is used to explicitly sync the UEFI variables to RPMB.</div> <div>2. RPMB partition needs to be erased by OS when switching from Advance to Base Variant on same device. To erase RPMB partition, run the following command.</div> <div>fastboot oem rpmb_erase</div>
80-70018-4 AC		May contain U.S. and international export controlled information					

Yes	No	No	No	Yes	No	Base + advance + upstream distro	
-----	----	----	----	-----	----	----------------------------------	--

Note:

- Base variants won't have this UEFI variable enabled irrespective of the RPMB provision.
- In the Base variant, Capsule update is triggered if capsule is found in EFI partition. *OsIndications* variable is ignored and TrialBoot & Rollback features are disabled.
- If *OsTrailBootStatus* variable is present, Capsule update won't happen for advance variants until the *OsIndications* is set.



6.1 Base variant

The base variant of RPMB can be either provisioned or unprovisioned. However, UEFI variables such as *OsIndications* and *OsTrailBootStatus* remains disabled. The system triggers a capsule update in this configuration only if it detects a capsule file in the EFI partition, located at */EFI/UpdateCapsule/*. After the system initiates the capsule update process, it updates the ESRT

table, which can be accessed through the kernel shell using the path `/sys/firmware/efi/esrt/entries/entry0/`.

6.2 Advance variant

The advanced variant of RPMB can operate in two modes: provisioned or unprovisioned.

- Unprovisioned RPMB: You can still start the capsule update sometimes if RPMB isn't set up. If the EFI finds a capsule, you can check the update status by looking at the ESRT table from the kernel shell. However, the trialboot and rollback feature works only when RPMB is set up.
- Provisioned RPMB: In contrast, when you provision RPMB, you use UEFI variables to determine whether to trigger the capsule update and enable trial boot and rollback features.

6.3 Capsule update triggering

You need to consider two scenarios:

- *OsIndications* enabled
 - When you enable *OsIndications*, you trigger the capsule update if a capsule is found in the EFI.
 - Additionally, you enable the trial boot and rollback feature based on the value of *OsTrialBootStatus*.
- *OsIndications* disabled
 - When you disable *OsIndications* but enable *OsTrialBootStatus*, you don't trigger the capsule update.

Note:

- Multiple capsules in ESP partition: If the ESP partition contains multiple capsules in EFI/UpdateCapsule, the system considers all capsules for the update and updates them alphabetically one after the other.
 - UEFI non-volatile variables: The system stores all UEFI non-volatile variables in the RPMB partition, which must be in a provisioned state. On a Qualcomm secure-boot-enabled device, the system will automatically provision the RPMB.
 - Qualcomm non-secure devices: On Qualcomm non-secure devices, the system doesn't automatically provision RPMB, but you can provision it using test keys. If you provision RPMB with test keys on a non-secure device, you can't re-provision RPMB with a device key while enabling Qualcomm secure boot on the device.
-

7 Interfaces

The DT nodes provide access to the boot interfaces and properties. The DT provides a data structure that describes the system's hardware information. The image execution environment, which includes firmware such as XBL and UEFI, reads this information. You can modify the DT properties associated with boot to customize the functionality of various device software features.

For information about modifying the interfaces using the QDTE tool and changing the DT properties in the prebuilt device tree blob (DTB) using the QDTE tool, see [Configure QDTE](#).

For more information about DT, see <https://github.com/devicetree-org>.

Platform-info

The Platform-info DT provides the information about CDT which contains platform information version, platform type, major hardware version, minor hardware version, subtype and OEM-id.

Boot device tree

The boot device tree properties provide information on how to enable and disable user-initiated emergency download (EDL) in XBL. The boot DT properties also describe the EDL timeout and the USB cable options that can force the system into the EDL mode. The boot-related functionality configures using the boot DT properties.

PMIC

The power management integrated circuit (PMIC) is a specialized electronic component that manages various power supply requirements within a device.

TLMM

The hardware controls the multiplexing of general-purpose input/output (GPIO) and alternate functions through the top-level mode multiplexer (TLMM). You can configure the GPIOs during the boot process using the DT properties for TLMM.

Buses

A bus facilitates data transfer among various system components. The DT properties allow for the configuration of the Qualcomm Universal Peripheral (QUP) v3 serial interface device nodes.

USB

The DT properties for USB allow you to adjust the USB signal quality.

Storage

To configure DT properties for storage, see [UEFI device tree](#).

7.1 Platform information

The platform information device tree provides details about the platform information version, platform type, major hardware version, minor hardware version, subtype and OEM-id.

If the CDT binary file isn't programmed on the hardware, the platform information from the device tree is used. The file path on the Linux machine is *boot_images/boot/Settings/Soc/<chipset>/Core/SocInfra/PlatformInfo/PlatformInfo.dtsi*.

- **PlatformInfo version** - Specifies the PlatformInfo version of the driver. The supported version numbers are 0x3 and 0x5.
- **Platform type** - Specifies the type of platform which is IOT.
- **Major hardware version** - Specifies the major version information of the platform.
- **Minor hardware version** - Specifies the minor version information of the platform.
- **Subtype** - Specifies the subtype for the platform type.
- **OEM variant ID** - Specifies user defined ID. It's reserved for OEMs and unused by Qualcomm.
 - 0x00 - Qualcomm reference platform.
 - 0x01-0xff - OEM variant
- **Number of key-value pair (KVP)** - Specifies dependent key values associated with platform type.

Note: OEM variant ID is supported from PlatformInfo version 5.

7.2 Boot device tree

The boot device tree properties explain how to enable and disable user-initiated emergency download (EDL) in XBL. They also detail the EDL timeout and the USB cable options that can force the system into EDL mode. You can use these properties to configure boot-related functionality.

The boot DT file path on the Linux build machine is */boot_images/boot/Settings/Soc/<chipset>/Core/Boot/sw_boot.dtsi*.

Table: Boot DT properties

Property name	Property description	Data type	Possible values/value range	Device behavior
vibration	Turn on/off vibration at XBL exit.	UINT32	<ul style="list-style-type: none"> 0x00 0x01 	<ul style="list-style-type: none"> ENABLE (0x01): Turns on the vibration. 0x01 is the default setting. DISABLE (0x00): Turns off the vibration.
forced_download.feature	<ul style="list-style-type: none"> Enable or disable the force emergency download mode (EDL) feature at the XBL stage. Flash images using USB cable along with key combination (Press POWER (PWR), VOL+ and VOL-keys together) 	UINT32	0x00	Force EDL feature (FEDL) DISABLE (0x00): FEDL is disabled.
			0x01	ENABLE (0x01): Enable user-initiated EDL in XBL.
forced_download.check_usb_option	To enter FEDL mode, select the USB cable options	UINT32	0x00	CHECK_USB_D_PLUS_GROUND (0x00): The device enters EDL mode when it is powered on using a special USB cable that grounds the D+ line. This is the default setting.

Property name	Property description	Data type	Possible values/value range	Device behavior
			0x01	CHECK_USB_D_PLUS_HIGH_WITH_TIMEOUT (0x01): Connect the device to USB + ground during boot. Device is forced into EDL mode if the connection is removed withing the boot timeout.
			0x02	<ul style="list-style-type: none"> Time out is provided by the forced_download.check_dp_timeout DT property in milliseconds (ms) CHECK_USB_REGULAR_CABLE (0x02): The device is forced to EDL when booting with a normal USB cable.

UEFI

In the UEFI environment, the device path protocol defines how to connect and identify the devices, with nodes being part of this protocol. The UEFI file path on the Linux build machine is `/boot_images/boot/Settings/Soc/Core/Uefi/cfg/ueficfg.dtsi`.

The following table lists the main types of UEFI device path nodes and property names.

Table: UEFI device path nodes and properties

DT property name	Data type	Possible values/value range	Description
OsConfigTableSelection	UINT32	Range is from 0x1 to 0x2 depending on the Hypervisor.	You can specify which Hypervisor to use. The values are as follows: <ul style="list-style-type: none"> • 0x1 - BOOT_TYPE_LINUX - Gunyah™ Hypervisor Software • 0x2 - BOOT_TYPE_LINUX_WITH_KVM
QcCapsuleRootCert	UINT8 list	Has the OpenSSL generated certificate content.	You use the root certificate for capsule authentication. This node has the entire root certificate content, which you use to authenticate the capsule. You also use this node for firmware updates.

7.3 PMIC

The power management integrated circuit (PMIC) is a specialized electronic component that manages various power supply requirements within a device.

The PMIC oversees various power supply requirements, including:

- Battery management tasks such as charging and gauging.
- User interface components such as flash, display, and RGB color space.
- System-on-chip (SoC) infrastructure elements such as clocks, analog-to-digital converters (ADC), and power-on (PON) functions.

Some PMIC resources can customize in XBL through the XB_CFG image. Configuring the PMIC XBL DT properties ensure optimal performance and integration. You can perform these customizations using a DT framework.

The PMIC DTSI files for the Linux host PC are at `boot_`

images/boot/Settings/Soc/<Chipset>/Core/PMIC/pm.dtsi, access.dtsi

The table lists the PMIC DT properties:

Table : PMIC DT properties

Property name	Property description	Data type	Possible values/value range	Device behavior
s2-kdpwr	Enable: PMIC power key pin reset configuration.	Boolean	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	The power key is configured to reset or shut down the device when it is asserted for a specific duration specified in s1-ms and s2-ms properties. Property to allow the DT setting to take effect. If PM_FALSE, the default hardware settings are programmed.
	reset-type: Select the reset type.	UINT8	<ul style="list-style-type: none"> PM_WARM_RESET PM_HARD_RESET PM_SHUTDOWN 	<ul style="list-style-type: none"> WARM_RESET is used for crash dump collection. HARD_RESET is used for rebooting the device. SHUTDOWN is used for shutting down the device
	s1-ms: Assertion time (to be chosen from the possible values).	UINT32	<ul style="list-style-type: none"> 0 32 56 80 128 184 272 408 608 904 1352 2048 3072 4480 6720 10256 	KDPWR_N_RESET_S1_TIMER in ms.

Property name	Property description	Data type	Possible values/value range	Device behavior
	s2-ms: Assertion time (to be chosen from the possible values).	UINT32	<ul style="list-style-type: none"> • 0 • 10 • 50 • 100 • 250 • 500 • 1000 • 2000 	KDPWR_N_RESET_S2_TIMER in ms.
s2-kdpwr-resin	Enable: Configuration to update the PMIC power and Resin key combination.	Boolean	<ul style="list-style-type: none"> • PM_TRUE • PM_FALSE 	Both power and Resin key combinations for the duration specified in the s1-ms and s2-ms properties. . The property allows the DT setting to take effect. If PM_FALSE, the hardware default settings are programmed.
	reset-type: Select the reset type.	UINT32	<ul style="list-style-type: none"> • PM_WARM_RESET • PM_HARD_RESET • PM_SHUTDOWN 	<ul style="list-style-type: none"> • WARM_RESET is used for crash dump collection. • HARD_RESET is used for rebooting the device. • SHUTDOWN is used for shutting down the device

Property name	Property description	Data type	Possible values/value range	Device behavior
	s1-ms: Assertion time value must choose from possible values.	UINT32	<ul style="list-style-type: none"> • 0 • 32 • 56 • 80 • 128 • 184 • 272 • 408 • 608 • 904 • 1352 • 2048 • 3072 • 4480 • 6720 • 10256 	RESIN_AND_KPDPWR_RESET_S1_TIMER in ms
	s2-ms: Assertion time value must choose from possible values.	UINT32	<ul style="list-style-type: none"> • 0 • 10 • 50 • 100 • 250 • 500 • 1000 • 2000 	RESIN_AND_KPDPWR_RESET_S2_TIMER in ms
s2-resin	Enable: PMIC Resin key pin used as volume down key.	Boolean	<ul style="list-style-type: none"> • PM_TRUE • PM_FALSE 	<p>The resin key can be configured to reset or shut down the device when it is asserted for the duration specified in the s1-ms and s2-ms.</p> <p>The property allows the DT setting to take effect. If the value is PM_FALSE, the hardware default settings are programmed.</p>

Property name	Property description	Data type	Possible values/value range	Device behavior
	reset-type: Select the reset type.	UINT8	<ul style="list-style-type: none"> PM_WARM_RESET PM_HARD_RESET PM_SHUTDOWN 	<ul style="list-style-type: none"> WARM_RESET is used for crash dump collection. HARD_RESET is used for rebooting the device. SHUTDOWN is used for shutting down the device.
	s1-ms: Assertion time value must choose from possible values.	UINT32	<ul style="list-style-type: none"> 0 32 56 80 128 184 272 408 608 904 1352 2048 3072 4480 6720 10256 	RESIN_N_RESET_S1_TIMER in ms
	s2-ms: Assertion time value must choose from possible values.	UINT32	<ul style="list-style-type: none"> 0 10 50 100 250 500 1000 2000 	RESIN_N_RESET_S2_TIMER in ms

Property name	Property description	Data type	Possible values/value range	Device behavior
s3-reset	Enable: Configure the S3 reset.	Boolean	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	<ul style="list-style-type: none"> If the device is stuck and cannot be restored with S2 reset, S3 reset does a fail-safe reset or factory reset. S3 reset is triggered when the key is asserted for the duration specified in the timer value-ms. The property allows the DT setting to take effect. If the value is PM_FALSE, the default hardware settings are programmed.
	s3-src: Select the reset source.	UINT8	<ul style="list-style-type: none"> PM_PON_RESET_SOURCE_KPDPWR PM_PON_RESET_SOURCE_RESIN PM_PON_RESET_SOURCE_RESIN_AND_KPDPWR PM_PON_RESET_SOURCE_RESIN_OR_KPDPWR 	S3 reset source can be the power key or the Resin key, or a combination of both.

Property name	Property description	Data type	Possible values/value range	Device behavior
	timervalue-ms: Assertion time value must choose from possible values.	UINT32	<ul style="list-style-type: none"> • 0 (Immediate) • 8 • 16 • 32 • 63 • 125 • 240 • 500 • 1000 • 2000 • 4000 • 8000 • 16000 • 32000 • 64000 • 128000 	RESET_S3_TIMER_n in ms
uvlo-config	PMIC_INDEX: Configure the under voltage lockout threshold of various PMICs.	UINT8	<ul style="list-style-type: none"> • PMIC_A (PMK7325) • PMIC_B (PM7325) • PMIC_C (PM7350C) • PMIC_D (PM7325B) • PMIC_E (PMR735A) • PMIC_I (PM7250B) • PMIC_K (PMG1110) 	When the input voltage of PMIC reaches the under voltage lockout (UVLO) threshold, the device shuts down abruptly. For each PMIC present in the Qualcomm reference device, configure the UVLO threshold. The pmic_index is used to select the PMIC for which the threshold should be changed.
	uvlo_thresh	UINT32	mV	UVLO threshold in millivolts
	uvlo_hyst	UINT32	mV	UVLO hysteresis in millivolts
	uvlo_enable	Boolean	PM_ENABLE PM_DISABLE	The property allows the DT setting to take into effect. If PM_DISABLE, the hardware default settings are programmed.

Property name	Property description	Data type	Possible values/value range	Device behavior
ovlo-config	PMIC_INDEX: Configure the over voltage lockout threshold of various PMICs.	PMIC_INDEX	<ul style="list-style-type: none"> • PMIC_A (PMK7325) • PMIC_B (PM7325) • PMIC_C (PM7350C) • PMIC_D (PM7325B) • PMIC_E (PMR735A) • PMIC_I (PM7250B) • PMIC_K (PMG1110) 	When the input voltage of PMIC reaches the over voltage lockout (OVLO) threshold, the device shuts down abruptly. For each PMIC present in the Qualcomm reference device, configure the OVLO threshold. The <code>pmic_index</code> is used to select the PMIC for which the threshold has to be changed.
	<code>ovlo_thresh</code>	–	mV	OVLO threshold in millivolts
	<code>ovlo_hyst</code>	–	mV	OVLO hysteresis in millivolts
	<code>ovlo_enable</code>	–	<ul style="list-style-type: none"> • PM_ ENABLE • PM_ DISABLE 	The property allows the DT setting to take into effect. If PM_ DISABLE, the hardware default settings are programmed.
Long-pwrkey-dbnc-chk	<code>dbnc-time-ms</code> : The power key debounces time to check for a valid keypress to boot up the device.	UINT32	msec	If the power key is not pressed for the configured <code>dbnc-time-ms</code> duration, the device shuts down in XBL or UEFI based on the config <code>chk-at</code> during boot up.
	<code>chk-at</code> : Check for keypress at the XBL or UEFI stage.	UINT8	0, 1, 2	<ul style="list-style-type: none"> • 0 = Do not check • 1 = Check-in XBL Loader stage • 2 = Check-in XBL core (UEFI)

Property name	Property description	Data type	Possible values/value range	Device behavior
sw-config To modify PMIC software configuration during boot up time for various LDOs, SMPSSs, Clocks, GPIOs	Verbose: Select PON log level.	UINT32	<ul style="list-style-type: none"> PM_EVENT_LOG_LEVEL_MIN PM_EVENT_LOG_LEVEL_VERBOSE PM_EVENT_LOG_LEVEL_RAWDATA PM_EVENT_LOG_LEVEL_MAX 	PON log indicates the various reasons for device power-on, OFF, and faults. <ul style="list-style-type: none"> Minimal version of PON logs printed in UART logs Verbose PON logs printed in UART logs Verbose PON logs along with raw SDAM register data printed in UART logs
	PM_LDO_SET_ENABLE: Enable the LDO.	UINT32	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	Configures the LDOs. See SMPS/LDO/CLK sample code
	PM_LDO_SET_VOLT: Set LDO for the voltage.	UINT32	mV	
	PM_LDO_SET_MODE: Set the LDO mode.	UINT32	<ul style="list-style-type: none"> 0: Low power 1: Bypass 3: Normal power mode 4: Retention mode 	
	PM_LDO_SET_PD_CTRL: Set the pull down on the LDO.	UINT32	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	
	PM_LDO_SET_PIN_CTRL: Enable pin control for the LDO.	UINT32	<ul style="list-style-type: none"> 0: HWEN0 1: HWEN1 2: HWEN2 	
	PM_LDO_SET_OCP_BROADCAST: Enable LDO OCP broadcast so that the device can shut down during OCP.	UINT32	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	

Property name	Property description	Data type	Possible values/value range	Device behavior
	PM_LDO_SET_AHC: Enable LDO automatic head room control.	UINT32	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	
	PM_LDO_SET_ULS: Set the upper limit for the LDO voltage.	UINT32	mV	
	PM_SMPS_SET_AHC_HR: Configure the SMPS AHC reserved head room.	UINT32	<ul style="list-style-type: none"> 0x0: 0mV 0x1: 8mV 0x1: 8mV 0x1: 8mV 0x1: 8mV 0x2: 16mV 0x3: 24mV 0x4: 32mV (DEFAULT) 0x5: 40mV 0x6: 48mV 0x7: 56mV 	
	PM_SMPS_SET_ENABLE: Enable the SMPS.	UINT32	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	
	PM_SMPS_SET_VOLT: Set the SMPS for the voltage.	UINT32	mV	
	PM_SMPS_SET_MODE: Set the SMPS mode.	UINT32	<ul style="list-style-type: none"> 0: Low power 2: Auto mode 3: Normal power mode 4: Retention mode 	
	PM_SMPS_SET_PD_CTRL: Set the pull down on the SMPS.	UINT32	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	

Property name	Property description	Data type	Possible values/value range	Device behavior
	PM_SMPS_SET_PIN_CTRL: Enable Pin control for the SMPS.	UINT32	<ul style="list-style-type: none"> • 0: HWEN0 • 1: HWEN1 • 2: HWEN2 	
	PM_SMPS_SET_OCP_BROADCAST: Enable SMPS OCP broadcast so that the device can shut down during OCP.	UINT32	<ul style="list-style-type: none"> • PM_TRUE • PM_FALSE 	
	T PM_SMPS_SET_AHC: Enable SMPS automatic head room control.	UINT32	<ul style="list-style-type: none"> • PM_TRUE • PM_FALSE 	
	PM_SMPS_SET_ULS: Set the upper limit to the SMPS voltage.	UINT32	mV	
	PM_GPIO_SET_ENABLE: Enable GPIO.	UINT32	<ul style="list-style-type: none"> • PM_TRUE • PM_FALSE 	
	PM_GPIO_SET_CFG_MODE: Configure GPIO mode.	UINT32	<ul style="list-style-type: none"> • 0:PM_GPIO_DIG_IN • 1:PM_GPIO_DIG_OUT • 2:PM_GPIO_DIG_IN_OUT • 3:PM_GPIO_ANA_PASS_THRU 	
	PM_GPIO_SET_OUTPUT_LVL: Set the GPIO output level.	UINT32	<ul style="list-style-type: none"> • 0: Level of pin is low • 1: Level of pin is high • 2 - Level of pin is Hi-Z 	

Property name	Property description	Data type	Possible values/value range	Device behavior
	PM_GPIO_SET_VOLT_SRC: Select GPIO voltage source as 1.8V or 3.6V.	UINT32	<ul style="list-style-type: none"> 0:PM_GPIO_VIN0 1:PM_GPIO_VIN1 	
	PM_GPIO_SET_OUT_BUFF_CONFIG: Select GPIO output buffer configuration.	UINT32	<ul style="list-style-type: none"> 0: CMOS output. 1:Open drain NMOS output 2: Open drain PMOS output 	
	PM_GPIO_SET_OUT_SRC_CFG: Select GPIO output source configuration.	UINT32	<ul style="list-style-type: none"> 0: Ground 1: Paired GPIO 2: Special function 1 3: Special function 2 4: Special function 3 5: Special function 4 6: D-test 1 7: D-test 2 8: D-test 3 9:L D-test 4 	
	PM_GPIO_SET_OUT_DRV_STR: Select GPIO pin drive strength.	UINT32	<ul style="list-style-type: none"> 0: Output buffer strength low. 1: Output buffer strength medium. 2: Output buffer strength high. 	

Property name	Property description	Data type	Possible values/value range	Device behavior
	PM_GPIO_SET_PULL_SEL: Select GPIO pull settings.	UINT32	<ul style="list-style-type: none"> • 0:30μA constant pulls up • 1:31.5μA constant pulls up • 3:1.5μA constant pulls up combined with 30μA boost • 4:10μA constant pull down • 5: No pull 	
	PM_SET_DELAY: Add delay between any 2 API calls.	UINT32	Delay in μ S.	
	PM_CLK_ENABLE: Enable the clocks such as In_bb_clk, rf_clk.	UINT32	<ul style="list-style-type: none"> • PM_TRUE • PM_FALSE 	
	PM_CLK_DRV_STR: Select the drive strength of the clock.	UINT32	0-63	
	PM_SPMI_CLK_DATA_CFG: Select the SPMI drive strength.	UINT32	<ul style="list-style-type: none"> • 0x0: LOW10PF • 0x1: MID20PF • 0x2: HIGH40PF • 0x3: VERYHIGH50PF 	
	PM_BUSID: Select SPMI bus.	UINT32	PM_BUSID_0	Must be PM_BUSID_0

Property name	Property description	Data type	Possible values/value range	Device behavior
	PMIC_Index: Select the PMIC.	UINT32	<ul style="list-style-type: none"> • PMIC_A (PMK7325) • PMIC_B (PM7325) • PMIC_C (PM7350C) • PMIC_D (PM7325B) • PMIC_E (PMR735A) • PMIC_I (PM7250B) • PMIC_K (PMG1110) 	See /boot_images/boot/Settings/ Include/pm_defines.h
	PMIC_Peripheral: Select the LDO, SMPS, clock, GPIO number.	UINT32	<ul style="list-style-type: none"> • PM_LDO_x • PM_SMPS_x • PM_CLK_x • PM_GPIO_x 	See /boot_images/boot/Settings/Include/ pm_defines.h and /boot_images/boot/QcomPkg/. Include /api/pmic/pm/pm_gpio.h (for GPIO enums).
	API parameter argument.	UINT32	PM_TRUE, PM_FALSE OR Integer OR Enum	See /boot_images/boot/Settings/Include/ pm_defines.h /boot_images/boot/QcomPkg/ Include /api/pmic/pm/pm_gpio.h (for GPIO enums)
Access (access.dtsi) To modify/enable the write access to PMIC peripheral for different subsystems such as AOP, APSS, UEFI, TZ, MSS BUSID: Select SPMI bus	UINT32	<ul style="list-style-type: none"> • PM_BUSID_0 • PM_BUSID_1 	<ul style="list-style-type: none"> • PM_BUSID_0– For SPMI0 bus secondary devices • PM_BUSID_1– For SPMI1 bus secondary devices 	–

Property name	Property description	Data type	Possible values/value range	Device behavior
	SecondaryID: Select the secondary ID.	UINT32	<ul style="list-style-type: none"> • 0: PMIC_A (PMK7325) • 1: PMIC_B (PM7325) • 2: PMIC_C (PM7350C) • 3: PMIC_D (PM7325B) • 4: PMIC_E (PMR735A) • 8: PMIC_I (PM7250B) • 9: PMIC_I (PM7250B) 	—
	PERIPH (PPID): Select the peripheral within PMIC.	UINT32	PMIC Peripheral Register base 0x1 to 0xFF	PMIC peripheral register base. Example: <ul style="list-style-type: none"> • For PM7325, GPIO_01 PPID is 0x88 • For PMK7325, PON_PBS PPID is 0x08

Property name	Property description	Data type	Possible values/value range	Device behavior
	OPERATION: Add or remove the channel with permission.	UINT32	<ul style="list-style-type: none"> • APPEND • REMOVE • APPEND_WITH_IRQ 	<ul style="list-style-type: none"> • APPEND: A channel is added for the PMIC peripheral with the defined DRV having the write permission. The default channel setting is retained. • REMOVE: The channel is removed for the PMIC peripheral with the defined DRV. • APPEND_WITH_IRQ: A channel is added for the PMIC peripheral with the defined DRV, which overrides the default IRQ owner DRV. In this case, the write access DRV is retained from the default setting. <p>Note: There can only be one IRQ DRV owner, while multiple write-access DRVs are allowed.</p>
Charger	pmic-index-charger: Select the charger PMIC index.	UINT32	PMIC_D or PMIC_I	Index of the primary charger PMIC
	Dead-battery-threshold: Set the dead battery threshold.	UINT32	2800 -3300	dBc threshold in millivolts. XBL allows you to boot to UEFI post the event.
	ichg-fs-cfg-enable: Increase the charge current range up to 20A.	UINT8	<ul style="list-style-type: none"> • PM_TRUE • PM_FALSE 	Flag to check if the full scale charging current setting should be applied.
	ichg-fs-cfg-value	UINT32	<ul style="list-style-type: none"> • 0: 10A • 1: 20A 	Supported full-scale charging current - 0: 10A; 1: 20A.
	pm-chg-batt-cfg-enable: Select the battery type as 1 S or 2 S.	UINT8	<ul style="list-style-type: none"> • PM_TRUE • PM_FALSE 	Flag to check if the 1S or 2S battery configuration should be applied.

Property name	Property description	Data type	Possible values/value range	Device behavior
	pm-chg-batt-cfg-is_1s_battery	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	To indicate if a 1S or 2S battery is present.
	dbg-board-id-cfg-min-id-ohms: Minimum batt_id in ohms for debug board detection.	UINT32	2000	Battery less platform with debug board, the batt_id should be within this min and max range to skip the battery charging.
	dbg-board-id-cfg-max-id-ohms: Maximum batt_id in ohms for debug board detection.	UINT32	14000	
	afp-cfg-too-hot-threshold: Set the AFP threshold temperature for a hot temperature.	UINT32	Temperature in degree C	If the battery temperature crosses any of the AFP thresholds, the device shuts down automatically. Example: 75 C
	afp-cfg-too-hot-enable: Enable/Disable AFP for hot temperature threshold config.	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	Enable/Disable AFP when the hot temperature threshold is reached. For PM_FALSE, the hardware default settings are programmed.
	afp-cfg-too-cold-threshold: AFP threshold temp for cold temperatures.	UINT32	Temperature in degree C	Example: -20 C
	afp-cfg-too-cold-enable: Enable/Disable AFP for cold temperature threshold config	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	<ul style="list-style-type: none"> Enable/Disable AFP when the cold temperature threshold is reached. For ``PM_FALSE`` value, the hardware default settings are programmed.
	no-batt-cfg-enable: Enable the battery-less configuration.	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	For PM_TRUE value, the device boots up automatically if the battery is not detected.
	no-batt-cfg-boot-without-batt: Continue to boot when the battery is not detected.	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	For ``PM_FALSE`` value, the device shuts down automatically if the battery is not detected.

Property name	Property description	Data type	Possible values/value range	Device behavior
	no-batt-cfg-icl-value-ma: Set the input current limit for a battery-less platform.	UINT32	0-5000	Input current limit value in mA
	float-voltage-cfg-enable: Configure the maximum battery voltage.	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	For PM_FALSE value, the hardware default settings are programmed.
	float-voltage-cfg-value-mv	UINT8	0-4500	Maximum battery voltage for charging
	pre-charging-current-cfg-enable: Configure the precharge current.	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	For PM_FALSE value, the hardware default settings are programmed.
	pre-charging-current-cfg-value-ma	UINT32	0-600	Precharge current in mA when battery voltage is < Vsys_min Ex: 3V
	fast-charging-current-enable: Configure the fast-charge current.	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	For PM_FALSE value, the hardware default settings are programmed.
	fast-charging-current-value-ma	UINT32	0-10000	–
	usbin-input-current-enable: Configure the USB input current limit.	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	For PM_FALSE value, the hardware default settings are programmed.
	usbin-input-current-value-ma	UINT32	0-5000	USB input current configuration
	dam-cable-chg-enable: Enable charging for DAM cable.	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	For PM_FALSE value, the hardware default settings are programmed. By default, the hardware disables the DAM cable charging.
	dam-cable-aicl-enable: Enable AICL for DAM cable.	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	For PM_FALSE value, the hardware default settings are programmed. By default, the hardware disables the DAM cable AICL.
	Secondary-charger-present: Indicate the Qualcomm secondary charger presence.	UINT8	<ul style="list-style-type: none"> PM_TRUE PM_FALSE 	Secondary charger settings

SMPS/LDO/CLK sample code

```

driver-post-init = <
    // PM_SMPS_SET_AHC_HR          PM_BUSID_0    PMIC_C    PM_SMPS_1
0x57
    // PM_SMPS_SET_ULS             PM_BUSID_0    PMIC_F    PM_SMPS_6
1375000    /*The Voltage is in microVolts*/
    // PM_LDO_SET_ULS             PM_BUSID_0    PMIC_F    PM_SMPS_6
1375000    /*The Voltage is in microVolts*/
    // PM_CLK_ENABLE              PM_BUSID_0    PMIC_A    PM_CLK_RF_1
PM_TRUE
    // PM_SMPS_SET_VOLT           PM_BUSID_0    PMIC_B    PM_SMPS_1
1200
    // PM_SMPS_SET_AHC            PM_BUSID_0    PMIC_B    PM_SMPS_1
PM_FALSE
    // PM_LDO_SET_AHC             PM_BUSID_0    PMIC_B    PM_LDO_1
PM_FALSE
    // PM_LDO_SET_VOLT           PM_BUSID_0    PMIC_B    PM_LDO_1
1200
    // PM_LDO_SET_MODE            PM_BUSID_0    PMIC_B    PM_LDO_1
PM_SW_MODE_NPM
    // PM_SMPS_SET_MODE           PM_BUSID_0    PMIC_B    PM_SMPS_1
PM_SW_MODE_NPM
    // PM_SMPS_SET_PD_CTRL        PM_BUSID_0    PMIC_B    PM_SMPS_1
PM_TRUE
    // PM_SMPS_SET_PIN_CTRL       PM_BUSID_0    PMIC_B    PM_SMPS_1
PM_TRUE
    // PM_SMPS_SET_OCP_BROADCAST  PM_BUSID_0    PMIC_B    PM_SMPS_1
PM_TRUE
    // PM_LDO_SET_PD_CTRL         PM_BUSID_0    PMIC_B    PM_LDO_1
PM_TRUE
    // PM_LDO_SET_PIN_CTRL        PM_BUSID_0    PMIC_B    PM_LDO_1
PM_TRUE
    // PM_LDO_SET_OCP_BROADCAST  PM_BUSID_0    PMIC_B    PM_LDO_1
PM_TRUE
    // PM_CLK_DRV_STR             PM_BUSID_0    PMIC_A    PM_CLK_RF_1
3
    // PM_SMPS_SET_ENABLE         PM_BUSID_0    PMIC_B    PM_SMPS_1
PM_TRUE
    // PM_LDO_SET_ENABLE          PM_BUSID_0    PMIC_B    PM_LDO_1
PM_TRUE
    // PM_SPMI_CLK_DATA_CFG       PM_BUSID_0    PMIC_A    0x2
0x2    /*BUS_ID, PMIC_ID, CLK buff Config, DATA buff
Config*/

```

```
PM_DELAY(10)
> ;
```

7.4 TLMM

The hardware controls the multiplexing of general-purpose input/output (GPIO) and alternate functions through the top-level mode multiplexer (TLMM). You can configure the GPIOs during the boot process using the DT properties for TLMM.

You can edit the following file paths to update the properties on the Linux host machine:

- /boot_images/boot/Settings/Soc/<Chipset>/Core/SocInfra/TLMM/<Chipset>-pinctrl.dtsi
- /boot_images/boot/Settings/Soc/<Chipset>/Core/SocInfra/TLMM/tlmm.dtsi

The table lists the properties that describe how to configure the default value of a GPIO based on the requirements of the Qualcomm reference device.

Table : TLMM DT properties

Property name	Property description	Data type	Possible values/value range	Device behavior
QCOM, sleep-config	GPIO configuration settings as defined in <code><Chipset>-pinctrl.dtsi</code> . The configurations are applied on device boot up.	UINT32	<p>Pin direction:</p> <ul style="list-style-type: none"> GPIO_INPUT – 0x1 GPIO_OUTPUT – 0x2 <p>Pin pull configuration:</p> <ul style="list-style-type: none"> GPIO_PULL_DOWN – 0x4 GPIO_PULL_UP – 0x8 GPIO_NO_PULL – 0x10 GPIO_KEEPER – 0x20 <p>Pin output:</p> <ul style="list-style-type: none"> GPIO_OUT_LOW – 0x40 GPIO_OUT_HIGH – 0x80 GPIO_PRG_YES – 0x100 GPIO_PRG_NO – 0x000 	<ul style="list-style-type: none"> The default value is GPIO_INPUT <ul style="list-style-type: none"> GPIO_INPUT: Allows the state of an input pin to be read GPIO_OUTPUT: Controls the state of an output pin The default value is GPIO_PULL_DOWN. <ul style="list-style-type: none"> GPIO_PULL_DOWN: Logic 0, consider as connected to Ground GPIO_PULL_UP: Logic 1, connected to Vdd supply GPIO_NO_PULL: Floating/High Impedance state GPIO_KEEPER: Maintain the previous state of the GPIO. When an SoC enters the deepest power-saving mode, this configuration is applied. The default value is GPIO_OUT_LOW <ul style="list-style-type: none"> GPIO_OUT_HIGH: Logic high, consider as connected to Vdd. The default value is GPIO_PRG_NO. <ul style="list-style-type: none"> GPIO_PRG_YES: Ensures that any unused GPIOs remain in a low-power state after bootup <p>Example:</p> <pre>(GPIO_INPUT GPIO_PULL_DOWN GPIO_OUT_LOW GPIO_PRG_NO) /* PIN 10 */</pre>
Compatible	Compatible property contains a read-only string that points to the compatible chipset.	String	–	<p>Example:</p> <pre>compatible = "qcom,<chipset>-pinctrl"</pre>

Property name	Property description	Data type	Possible values/value range	Device behavior
reg	Represents the read only GPIO base address and size.	UINT32	–	In a <code>reg</code> property tuple, the first index contains the base address, and the second index contains the size. Example: <pre>reg = <0xf100000 0x100000>;</pre>
ngpios	Number of read only GPIO pins in chipset.	UINT32	–	Example: <pre>ngpios = <175>;</pre>
width	Each GPIO pin has its own set of read only control registers. Width indicates pin to pin register offset.	–	–	Example: <pre>width = <0x1000>;</pre>
id	Hardware instance of read only GPIO pad ID.	UINT32	–	Example: <pre>id = <0x0>;</pre>
version	GPIO read only driver version. Driver 1.0 refers as 0x1.	UINT32	–	Example: <pre>version = <0x1>;</pre>
gpio-controller	Identifier to represent the read only connected device as a GPIO device.	String	–	–
phandles for pin configurations	Mux configuration that is used to configure the alternative functionality of GPIOs. For more information, see Pin descriptions .	–	<pre>sdc4_data_1: sdc4_data_1 { mux = <13 3>; };</pre>	On bootup, the GPIO configures to alternate functionality of GPIO. Example: <pre>sdc4_data_1: sdc4_data_1 { mux = <13 3> };</pre> In this example, 13 refers to the GPIO number and 3 indicates the alternate function selection.

7.5 Buses

A bus facilitates data transfer among various system components. The DT properties allow for the configuration of the Qualcomm Universal Peripheral (QUP) v3 serial interface device nodes.

QUP v3 supports interintegrated circuit (I²C), serial peripheral interface (SPI), and universal asynchronous receiver-transmitter (UART) serial engines. The DT properties include the active and sleep settings for these serial engines.

For more information about QUP v3 and its peripheral interfaces, see [Overview of peripheral interfaces](#).

You can apply the serial engine protocol and active configurations when the firmware loads onto

the serial engine.

You can configure the following files at the Linux host PC:

- `boot_images/boot/Settings/Soc/<chipset>/Core/Buses/qup_common/<chipset>-qupv3-pinctrl.dtsi`
- `boot_images/boot/Settings/Soc/<chipset>/Core/Buses/qup_common/<chipset>-qupv3.dtsi`

In `&top_qup k _se n`, **k** represents the QUP number and **n** represents the serial engine number. The table lists the DT properties:

Table : Bus DT properties

Property name	Property description	Data type	Possible values/value range	Device behavior
<code>status = "disabled";</code>	Status control to enable/disable serial interfaces supported for QUP v3 serial engine (I2C/SPI/UART) protocols.	String	<ul style="list-style-type: none"> • Okay • Disabled 	Enables or disables the serial engine.
<code>&top_qup**k**_se**n**_i2c_active</code>	GPIOs active settings for I2C serial engines.	UNIT32-array	Reference to a node label.	Sets active GPIO configurations for the I2C protocol.
<code>&top_qup**k**_se**n**_i2c_sleep</code>	GPIOs sleep settings for I2C.	UNIT32-array	Reference to a node label.	Sets sleep GPIO configurations for the I2C protocol.
<code>&top_qup**k**_se**n**_spi_active</code>	GPIOs active settings for SPI.	UNIT32-array	Reference to a node label.	Sets active GPIO configurations for the SPI protocol.
<code>&top_qupk**k**_se**n**_spi_sleep</code>	GPIOs sleep settings for SPI.	UNIT32-array	Reference to a node label.	Sets sleep GPIO configurations for the SPI protocol.
<code>&top_qup**k**_se**n**_uart_active</code>	GPIOs active settings for UART.	UNIT32-array	Reference to a node label.	Sets active GPIO configurations for the UART protocol.

Property name	Property description	Data type	Possible values/value range	Device behavior
&top_qup\ **k**\ se\ **n**\ uart_sleep	GPIOs sleep settings for UART.	UNIT32-array	Reference to a node label.	Sets sleep GPIO configurations for the UART protocol.

Sample configuration

```
TOP_QUP_0{}
/*TOP_QUP_0_SE_0 Instance */
TOP_QUP_0_SE_0{
status = "disabled"; /* status to enable/disable SE0 Node */
pinctrl-0 = <&top_qup0_se0_i2c_active>;
pinctrl-1 = <&top_qup0_se0_i2c_sleep>;
pinctrl-2 = <&top_qup0_se0_spi_active>;
pinctrl-3 = <&top_qup0_se0_spi_sleep>;
pinctrl-4 = <&top_qup0_se0_uart_active>;
pinctrl-5 = <&top_qup0_se0_uart_sleep>;
```

7.6 USB

The DT properties for USB allow you to adjust the USB signal quality.

For information about USB features and architecture, see [USB](#).

The DT file path at the Linux host PC is: /boot_images/boot/Settings/Soc/
<Chipset>/Core/WiredConnectivity/USB/usb.dtsi.

Table : USB DT properties

Property name	Property description	Data type	Possible values/value range	Device behavior
path=/soc/usb0/hs_phy_cfg	Applies to the primary USB controller high-speed PHY (HSPHY) for configuring the USB signal quality.	UINT32-array	<ul style="list-style-type: none"> An array of address, value pairs <addr, val>. addr is 4 bytes in length. It can have 1 of the 4 values: [0x88E306C, 0x88E3070, 0x88E3074, 0x88E3078]. val is of 1byte in length, range: 0x00 to 0xFF. 	By tuning this property, the USB signal quality of the primary USB HSPHY is improved.

Property name	Property description	Data type	Possible values/value range	Device behavior
<code>path=/soc/usb0/ss_phy_cfg</code>	Applies to the primary USB controller SuperSpeed PHY (SSPHY) for configuring the signal quality.	UINT32-array	<ul style="list-style-type: none"> • Array of address, value pairs <addr, val>. • addr is 4bytes in length: [0x088E8000 to 0x088EB000]. • val is of 1byte in length, range: 0x00 to 0xFF. 	By tuning this property, the USB signal quality for primary USB SSPHY is improved.

Property name	Property description	Data type	Possible values/value range	Device behavior
<code>path=/soc/usb1/hs_phy_cfg</code>	Applies to the secondary USB controller high-speed PHY (HSPHY). Used for configuring the USB signal quality.	UINT32-array	<ul style="list-style-type: none"> • Array of address, value pairs <addr, val>. • addr 4bytes in length. It can have 1 of the 4 values: [0x88E406C, 0x88E4070, 0x88E4074, 0x88E4078]. • val is of 1byte in length, range: 0x00 to 0xFF. 	By tuning this property, the USB signal quality for the secondary USB HSPHY is improved.

7.7 Storage

To configure DT properties for storage, see the [Storage guide](#). The UFS device tree specifies the UFS host configuration parameters such as number of gears and lanes, rate value, and timeout values.

8 Tools to run the boot process

The tools for the boot process include Qualcomm device tree editor (QDTE), Fastboot, and capsule generation in UEFI. You can access QDTE, a GUI editor tool, through the [Qualcomm Software Center](#) (QSC), whereas Fastboot is a diagnostic tool that offers a Recovery mode known as the Fastboot mode. The tools for the boot process include

- QDTE
- Fastboot
- Capsule generation

QDTE

QDTE is a GUI tool that's used to configure the DTB files. It can also be used to change and configure Interfaces inside the `xb1_config.elf` file. You can download QDTE from QSC and launch the UI using the command `./qdte`.

- In Windows by clicking the QDTE app.
- In Linux run the command `./qdte`.

Note: In Linux, check the QDTE installation path before running `./qdte`.

Fastboot

Fastboot allows you to flash software images such as the boot loader and helps in system recovery.

capsule generation in uefi

- Capsule generation tools are specialized utilities designed to create capsule files, which are essential for performing firmware updates on various devices.
- These tools streamline the process of packaging firmware updates into a format that can be deployed and applied to the device hardware.
- The tools can be run on both Linux and Windows environments,

providing flexibility for different development setups.

- By using OpenSSL for certificate generation and signing, the tools ensure that the firmware updates are secure and authenticated. To set up the capsule see the capsule generation in uefi.

For more information about QSC, see [Qualcomm Software Center User Guide](#).

9 Develop UEFI applications

Licensed users with authorized access can develop applications based on UEFI and configure hardware IDs for the Qualcomm reference device using the CDT. For information about how to synchronize the code for CDT and UEFI development, see [Develop in Qualcomm Linux Boot Guide - Addendum](#).

10 Configure QDTE

The QDTE graphical user interface (GUI) tool configures and modifies Device Tree Binary (DTB) files. For more information, see [Interfaces](#).

For more details about QDTE features, see QDTE.

This section explains how to use QDTE tool to do the following:

- Load, parse, edit and save the DTB file
- Read and write the DTB file from device or on host machine
- Set up the configuration on your host device, whether it's Linux or Windows.

Set up the configuration on your host device as follows:

Table : Configuration for Linux/Windows

Property	File path for Linux/Windows host machine
SecTool path: Installation path to SecToolsv2, a tool used to perform secure operations on software images.	<ul style="list-style-type: none">• Linux: <code>SECTOOLS="META_PATH"/<chipset>.LE.X.x/common/sectoolsv2/ext/Linux/sectools</code>• Windows: <code>SECTOOLS="META_PATH"\<chipset>.LE.X.x\common\sectoolsv2\ext\windows\sectools</code>. For more information on SecToolsv2 for Windows, see SecTools V2: Metabuild Secure Image User Guide
Profile file: Used for signing the security profile XML files.	<ul style="list-style-type: none">• Linux: <code>"META_PATH"/<chipset>.LE.X.x/common/sectoolsv2/kodiak_security_profile.xml</code>• Windows: <code>"META_PATH"\<chipset>.LE.X.x\common\sectoolsv2\kodiak_security_profile.xml</code>
Sign cmd json: Used for signing the JSON file from the QDTE installation directory. The test signing mode is used for working on a nonsecure device. For secure devices, see enable secure boot .	<ul style="list-style-type: none">• Linux: <code>/local/mnt/qcom/QDTE/test_signing_mode.json</code>• Windows: <code>C:\Program Files (x86)\Qualcomm\QDTE\test_signing_mode.json</code>

Property	File path for Linux/Windows host machine
Devprg file: Device programmer file.	<ul style="list-style-type: none"> Linux: "META_PATH"/<chipset>.LE.X.x/BOOT.MXF.1.0.c1-00134-KODIAKLA-2/boot_images/boot/QcomPkg/SocPkg/Kodiak/Bin/LAA/DEBUG/prog_firehose_ddd.elf Windows: "META_PATH"/<chipset>.LE.X.x\BOOT.MXF.1.0.c1\boot_images\boot\QcomPkg\SocPkg\Kodiak\Bin\LAA\DEBUG\prog_firehose_ddd.elf
Flash type: Type of storage, which in this case is UFS.	–

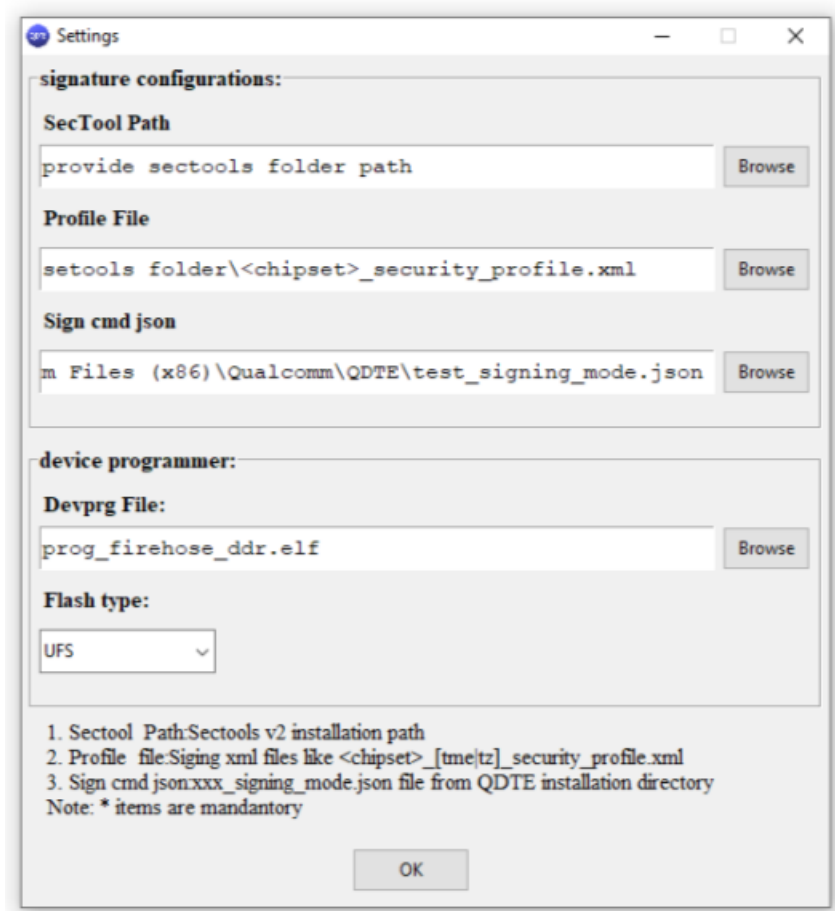


Figure : Settings for Linux signature configuration



Figure : Settings for Windows signature configuration

Use the following procedure to configure the DTB files using QDTE. This procedure is applicable only to XBL and UEFI DTB.

1. To open the XBLConfig ELF file, go to *File > Open DTB Elf > From Build*.
 - If the file is invalid or corrupted, the system generates an error. To prevent the error, ensure that you download the valid boot binaries, including `xbl_config.elf`, and use them with the XBL configuration. The binary releases the boot component, which is part of the firmware binary.
 - If the file loads successfully, a tree view appears, populated with the DTB elements.
2. Double-click on the XBLConfig ELF file to edit it.

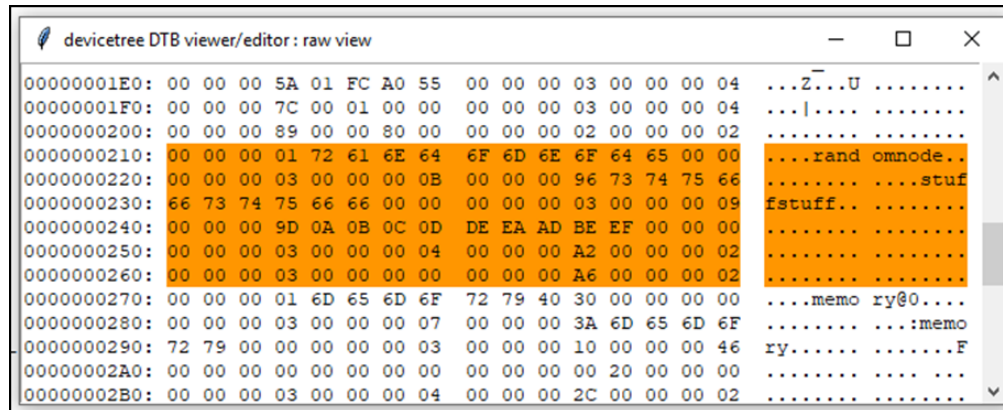
Table : DT items

Tree item	Description
Add New Child	Adds a child in the tree.
Node	Asks for the name of the new node and adds it. If the node name is invalid, an error occurs.
Property of Type	Opens a menu that allows you to choose the type of property to add. Adding a property displays the <i>Edit</i> dialog box corresponding to the type of the new property. On confirmation, a new property of the specified type is added.

You can remove all items in the DT, except for the root node. To delete an item, right-click on it and select *Delete*. Deleting a node also removes its child nodes, if any.

- To edit a property, you can either right-click and select *Edit* or double-click to open a dialog box. This dialog box allows you to edit both the property name and value.
- To view the DTB file as a binary hexadecimal dump, go to *View > Raw*, which displays the exported DTB file.

The following example of the raw view shows the offset, hexadecimal value, and corresponding character columns, with a node highlighted in orange.

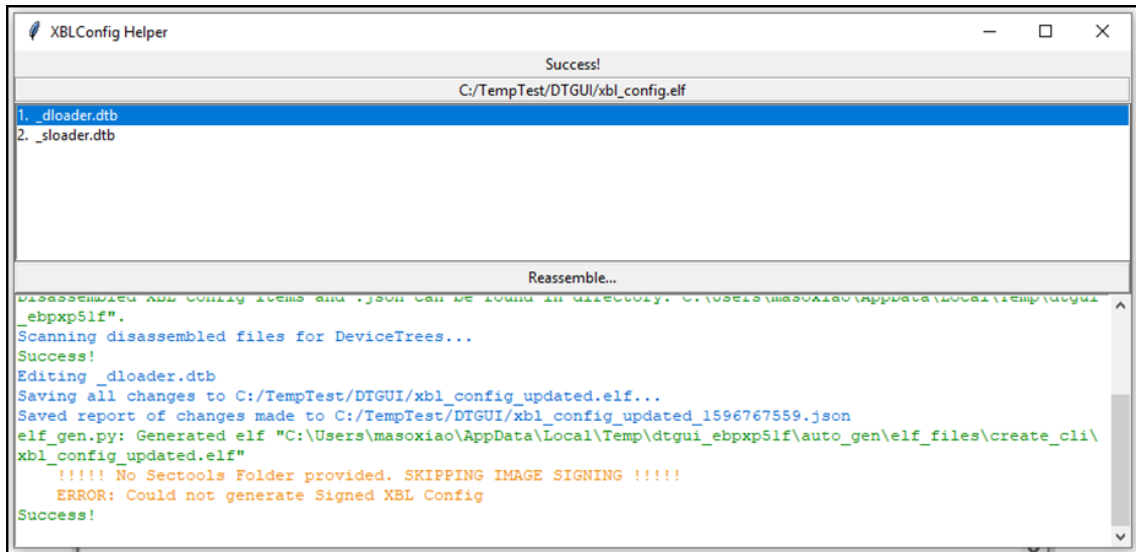


To remove all highlights, go to *View > Clear Highlights*.

- To save the modified DTBs, you can either go to *File > Save* (or press **CTRL + S**) to overwrite the existing file or go to *File > Save Copy As* (or press **CTRL + SHIFT + S**) to save a copy of the DTB to a new file.
- To generate a summary of the changes, select *File > Export Change Report*.
Creates a JSON file listing all changes made to the DT in the session, along with the result of the DTB after the operation.
- To save the modified DTB files within the main QDTE, go to the XBLConfig helper window, and click *Reassemble and Sign*. Saves your modified XBLConfig file.

10. Next, choose a name for the output file.

The toolchain re-assembles the ELF using XBLConfig. If an error message indicates that a signed XBLConfig wasn't generated, it downgrades the error to a warning and shows a success message instead.



11 Debug logs

The universal asynchronous receiver/transmitter (UART) logs enable you to monitor and debug data exchanged between the Qualcomm Linux software and a device using a UART interface.

The sample UART logs help you identify the various stages of system bootup. In the event of a crash, you can use these logs to do the following:

- Determine the stage at which the crash occurred
- Gain a high-level understanding of the execution process

11.1 PBL/XBL log

The sample primary boot loader/eXtensible boot loader (PBL/XBL) log indicates the start and end of the PBL and XBL.

For more information about PBL and XBL, see Boot loader.

The following table lists log entries that can help you verify the start and end of the boot loaders.

PBL/XBL log entries

Log entry	Description
S - Boot Interface: UFS	Indicates the booting of Qualcomm Linux software images from UFS as the boot image is stored in UFS.
S - PBL Patch Ver: 1	Indicates the start of PBL.
S - 69043 - PBL, End	Indicates the end of PBL.
B - 79574 - SBL1, Start	Indicates the start of XBL. Both SBL1 and XBL refer to the same secondary boot loader.
B - 1394338 - SBL1, End	Indicates the end of XBL.
D - 1318210 - SBL1, Delta	Indicates the time difference in microseconds between start and end of XBL.

Note: The numbers in the log entry column of the table show the timestamp in microseconds.

The following is a sample log.


```

Format: Log Type - Time(microsec) - Message - Optional Info
Log Type: B - Since Boot(Power On Reset), D - Delta, S - Statistic
S - QC_IMAGE_VERSION_STRING=BOOT.<XX.x.x.xx-xxxxx-XXXXX-x>
S - IMAGE_VARIANT_STRING=Soc<XXXXXXX>
S - OEM_IMAGE_VERSION_STRING=ip-<xx-xxx-xxx-xx>**
S - Boot Interface: UFS**
S - Secure Boot: Off
S - Boot Config @ 0x00786070 = 0x000000c1
S - JTAG ID @ 0x00786130 = 0x001980e1
S - OEM ID @ 0x00786138 = 0x00000000
S - Serial Number @ 0x00786134 = 0xf744c1e9
S - OEM Config Row 0 @ 0x007841c0 = 0x0000000000000000
S - OEM Config Row 1 @ 0x007841c8 = 0x0000000000000000
S - Feature Config Row 0 @ 0x00784148 = 0x0000000000000000
S - Feature Config Row 1 @ 0x00784150 = 0x0000000000000000
S - Core 0 Frequency, 1516 MHz**
S - PBL Patch Ver: 1**
D -      6626 - pbl_apps_init_timestamp
D -     31574 - bootable_media_detect_timestamp
D -      853 - bl_elf_metadata_loading_timestamp
D -      704 - bl_hash_seg_auth_timestamp
D -     6621 - bl_elf_loadable_segment_loading_timestamp
D -     4638 - bl_elf_segs_hash_verify_timestamp
D -    17185 - bl_sec_hash_seg_auth_timestamp
D -      811 - bl_sec_segs_hash_verify_timestamp
D -      31 - pbl_populate_shared_data_and_exit_timestamp**
S -     69043 - PBL, End
B -     79574 - SBL1, Start**
B -    210297 - SBL1 BUILD @ <XXXXXXX>
B -    214445 - usb: usb_shared_xbl_dtb_node_init dtb status , 0x3
B -    223717 - usb: HS: device tree Not available , 0x3
B -    224114 - usb: eud_serial_upd , 0xf744c1e9
D -    229421 - sbl1_hw_init
D -      519 - media_init:1
D -      0 - smss_load_cancel
B -    239455 - SMSS - Image Load, Start
D -     3111 - SMSS - Image Loaded, Delta - (0 Bytes)
D -     1037 - Auth Metadata
D -     5825 - sbl1_xblconfig_init
B -    254522 - XBL Config - Image Load, Start
B -    258640 - DTB Found: [pre-ddr][7601f201000000][0]
D -     4453 - boot_pre_ddr_dtb_load
B -    267851 - UFS Device Tree Settings

```

```
B - 271206 - UFS Init Speed: HS Enabled 1, Gear 4 2 Lane Rate 2
B - 280874 - UFS Perf Speed: HS Enabled 1, Gear 4 2 Lane Rate 2
B - 287096 - UFS Timeouts(us): fDeviceInit 2500000, UTRD Poll
30000000
D - 45323 - media_init:2
B - 387868 - UFS INQUIRY ID: SAMSUNG KM2L9001CM-B518 0700
B - 389485 - UFS Boot LUN: 1
B - 400953 - UFS GEAR: 3
D - 95099 - media_init:3
D - 30 - shrm_load_cancel
B - 410835 - SHRM - Image Load, Start
D - 702 - Auth Metadata
D - 1403 - Segments hash check
D - 12230 - SHRM - Image Loaded, Delta - (35752 Bytes)
D - 0 - boot_default_cdt_init
D - 488 - boot_cdt_init
B - 435540 - CDT - Image Load, Start
B - 438407 - CDT Version:3,Platform ID:32,Major ID:1,Minor ID:0,
Subtype:2
D - 16928 - sbll_hw_platform_pre_ddr
D - 0 - devcfg init
B - 464942 - PMIC A:2.0 B:1.0 C:2.2 I:1.0
B - 466772 - PM: PM8 0x4
B - 469181 - PM: xVdd reset
B - 471835 - PM: PON by SYSOK
B - 724497 - PM: SET_VAL:Skip
B - 724527 - PM: Verifying PON-Trigger specific configurations &
current PON-Trigger
B - 733281 - PM: All PON-Trigger specific configs verified.
Proceeding to BOOT
B - 742736 - PM: PSI: b0x06_v0x3c
B - 745450 - PM: Device Init # SPMI Transn: 15422
D - 291031 - pm_device_init, Delta
B - 751154 - pm_driver_init, Start
B - 762286 - PM: Driver Init # SPMI Transn: 500
D - 7655 - pm_driver_init, Delta
B - 766983 - PM: CHG Type in CHG init : 0
B - 771375 - PM: Battery ID: 76120hm
B - 774547 - PM: debug board connected
B - 778207 - vsense_init, Start
D - 0 - vsense_init, Delta
D - 339312 - sbll_hw_pre_ddr_init
D - 0 - boot_dload_handle_forced_dload_timeout
D - 2928 - sbll_load_ddr_training_data
```

```

B -      803034 - Pre_DDR_clock_init, Start
D -          61 - Pre_DDR_clock_init, Delta
D -      12902 - sbll_ddr_set_params
B -      814746 - sbll_ddr_init, Start
B -      818162 - LP4 DDR detected
B -      832772 - eCDT MRR - Data Starting Address: 0x09066D00
D -      15250 - sbll_ddr_init, Delta
B -      838445 - DSF version = 262.0.46
B -      841830 - Manufacturer ID = 1, Device Type = 7
B -      845399 - Rank 0 size = 2048 MB, Rank 1 size = 4096 MB
B -      850248 - Row Hamming DDR
B -      855769 - Row Hammer Check : DRAM supports unlimited MAC Value
: MR_RH[OP2:0 = 0] & MR_RH[OP3 = 1] for CH0 & CS0
B -      864431 - Row Hammer Check : DRAM supports unlimited MAC Value
: MR_RH[OP2:0 = 0] & MR_RH[OP3 = 1] for CH0 & CS1
B -      875136 - Row Hammer Check : DRAM supports unlimited MAC Value
: MR_RH[OP2:0 = 0] & MR_RH[OP3 = 1] for CH1 & CS0
B -      885842 - Row Hammer Check : DRAM supports unlimited MAC Value
: MR_RH[OP2:0 = 0] & MR_RH[OP3 = 1] for CH1 & CS1
D -      81893 - sbll_ddr_init
D -          31 - boot_pre_ddi_entry
B -      904325 - do_ddr_training, Start
B -      945530 - DDR: Start of DDR Training Restore
B -      949068 - Current DDR Freq = 1709 MHz
B -      950166 - Max enabled DDR Freq = 2092 MHz
B -      954192 - DDR: End of DDR Training Restore
D -      51057 - do_ddr_training, Delta
D -      58834 - sbll_do_ddr_training
D -          518 - boot_ddi_entry
B -      970022 - Pimem init cmd, entry
D -          9211 - Pimem init cmd, exit
B -      982466 - External heap init, Start
B -      985546 - External heap init, End
D -      22112 - sbll_post_ddr_init
D -          30 - sbll_hw_init_secondary
B -      996252 - DDR - Image Load, Start
B -      1001559 - DTB Found:
[post-ddr][7601f201000000][2001010000000000]
D -          9486 - boot_post_ddr_dtb_load
D -          762 - boot_fedl_check
B -      1013789 - APDP - Image Load, Start
D -          2959 - APDP - Image Loaded, Delta - (0 Bytes)
D -          0 - boot_dload_dump_security_regions
D -          0 - ramdump_load_cancel

```

```
B - 1030137 - RamDump - Image Load, Start
D - 3325 - RamDump - Image Loaded, Delta - (0 Bytes)
D - 0 - boot_update_abnormal_reset_status
D - 0 - boot_cache_set_memory_barrier
D - 0 - boot_smem_debug_init
D - 458 - boot_smem_init
D - 0 - boot_smem_alloc_for_minidump
D - 61 - boot_smem_store_pon_status
D - 30 - sbll_hw_platform_smem
D - 61 - boot_ddr_share_data_to_aop
D - 488 - boot_clock_init_rpm
D - 0 - boot_vsense_copy_to_smem
D - 0 - boot_populate_ram_partition_table
D - 30 - boot_populate_ddr_details_shared_table
D - 0 - sbll_tlmm_init
D - 0 - sbll_efs_handle_cookies
B - 1092906 - OEM_MISC - Image Load, Start
D - 762 - Auth Metadata
D - 274 - Segments hash check
D - 10645 - OEM_MISC - Image Loaded, Delta - (5048 Bytes)
B - 1106875 - QTI_MISC - Image Load, Start
D - 5704 - QTI_MISC - Image Loaded, Delta - (0 Bytes)
B - 1122308 - PM: PM Total Mem Allocated: 2340
D - 5520 - sbll_pm_aop_pre_init_wrapper
B - 1126883 - AOP - Image Load, Start
D - 763 - Auth Metadata
D - 1677 - Segments hash check
D - 13085 - AOP - Image Loaded, Delta - (200732 Bytes)
B - 1143292 - QSEE Dev Config - Image Load, Start
D - 854 - Auth Metadata
D - 610 - Segments hash check
D - 13024 - QSEE Dev Config - Image Loaded, Delta - (53248
Bytes)
B - 1165344 - QSEE - Image Load, Start
D - 17873 - Auth Metadata
D - 21594 - Segments hash check
D - 82441 - QSEE - Image Loaded, Delta - (3900768 Bytes)
B - 1251811 - DTB: invalid vibration prop value
B - 1256722 - DTB: Vibration Enabled
D - 10217 - sbll_hw_play_vibr
B - 1264865 - SEC - Image Load, Start
D - 3142 - SEC - Image Loaded, Delta - (0 Bytes)
B - 1271758 - CPUCFW - Image Load, Start
D - 17598 - Auth Metadata
```

```

D -      17385 - Segments hash check
D -      48007 - CPUCPFW - Image Loaded, Delta - (171180 Bytes)
B -     1328793 - QHEE - Image Load, Start
D -      17843 - Auth Metadata
D -       6954 - Segments hash check
D -      27664 - QHEE - Image Loaded, Delta - (1478736 Bytes)
B -     1359781 - APPSBL - Image Load, Start
D -       671 - Auth Metadata
D -      10889 - Segments hash check
D -      25315 - APPSBL - Image Loaded, Delta - (2564048 Bytes)
D -          0 - sbl1_save_appsbl_index**
B -     1394338 - SBL1, End
D -     1318210 - SBL1, Delta**

```

11.2 UEFI log

The UEFI log provides information about the start and end of UEFI.

The table lists the log entries that can help you verify the start and end of UEFI.

Table : Checking UEFI log

Log entry	Description
UEFI Start [1568]	Indicates the start of UEFI
[3086] UEFI End	Indicates the end of UEFI

Note: The numbers in the log entry column of the table show the timestamp in milliseconds.

The next example log uses the debug build flavor.

To use the build flavor, replace the 'RELEASE' string with 'DEBUG' in the boot build command. For more information, see [Build BOOT](#).

```

- UEFI Start      [ 1568]
- 0x09FC01000 [ 1571] Sec.efi
  ASLR           : ON
  DEP            : ON (RTB)
  Timer Delta    : +11 mS
  RAM Entry 0   : Base 0x0080000000 Size 0x003A800000
  RAM Entry 1   : Base 0x00C0000000 Size 0x0040000000
  RAM Entry 2   : Base 0x0100000000 Size 0x0100000000

```

```
Total Available RAM : 6056 MB (0x017A800000)
Total Installed RAM : 6144 MB (0x0180000000)
Multithread      : ON (Lib ver 1.2)
CPU Cores       : 8 (init 2)
Init 1 aux cores of 7
Init CPU core 1
CONF File      : uefiplatLA.cfg
UEFI Ver      : 6.0.231205.BOOT.<XXX.x.x.Xx-xxxxx-XXXXXXX-x>
Build Info    : 64b <XXXX>
Boot Device   : UFS
PROD Mode     : FALSE
Retail        : FALSE
Scheduler up on Core 1
DTB config    : client[0]..trace[0]..verbose[0]
- 0x09F0CF000 [ 1630] DxeCore.efi
Loading DxeCore at 0x009F0CF000 EntryPoint=0x009F0D0000
HOBLIST address in DXE = 0x9EEC0018
FV Hob        0x9FC00000 - 0x9FE70FFF
FV Hob        0x9F10E000 - 0x9F47AFFF
FV2 Hob       0x9F10E000 - 0x9F47AFFF
- 631008B0-B2D1-410A-8B49-2C5C4D8ECC7E - 00000000-0000-0000-0000-
000000000000
- 0x09F069000 [ 1633] EnvDxe.efi
- 0x09F05F000 [ 1634] RscRtDxe.efi
- 0x09F057000 [ 1634] SCHandlerRtDxe.efi
- 0x09EB94000 [ 1634] RuntimeDxe.efi
- 0x09F049000 [ 1635] ArmCpuDxe.efi
- 0x09F040000 [ 1636] ArmGicDxe.efi
- 0x09F039000 [ 1636] MetronomeDxe.efi
- 0x09F031000 [ 1636] ArmTimerDxe.efi
- 0x09F027000 [ 1636] SmemDxe.efi

- 0x09EFCD000 [ 1637] DALSys.efi
- 0x09EFC4000 [ 1637] HWIODxeDriver.efi
- 0x09EFB9000 [ 1637] ChipInfo.efi
- 0x09EFB1000 [ 1638] PlatformInfoDxeDriver.efi
- 0x09EF8D000 [ 1638] HALIOMMU.efi
- 0x09EF79000 [ 1687] ULogDxe.efi
- 0x09EF71000 [ 1687] CmdDbDxe.efi
- 0x09EF69000 [ 1688] PwrUtilsDxe.efi
- 0x09EF56000 [ 1688] NpaDxe.efi
- 0x09EF46000 [ 1689] RpmhDxe.efi
- 0x09EF3A000 [ 1689] VcsDxe.efi
- 0x09EF0C000 [ 1690] ClockDxe.efi
```

```
Cluster 0:          0 Hz
Cluster 2:          0 Hz
- 0x09EEFF000 [ 1697] ICBdxe.efi
- 0x09EEF6000 [ 1699] ShmBridgeDxeLA.efi
- 0x09EEE7000 [ 1700] ScmDxeLA.efi
- 0x09EEDE000 [ 1702] DALTLMM.efi
- 0x09EED4000 [ 1702] SPMI.efi
- 0x09EECA000 [ 1707] I2C.efi
- 0x09EB8B000 [ 1707] ResetRuntimeDxe.efi
- 0x09EB66000 [ 1708] PmicDxe.efi
PM0: 47, PM1: 63, PM2: 49, PM8: 46,
Module cannot re-initialize DAL module environments
- 0x09EB5B000 [ 1726] DiskIoDxe.efi
- 0x09EB4E000 [ 1726] PartitionDxe.efi
- 0x09EB46000 [ 1726] EnglishDxe.efi
- 0x09EB3B000 [ 1727] FvSimpleFileSystem.efi
- 0x09EB1F000 [ 1727] SdcccDxe.efi
- 0x09EAFE000 [ 1728] UFSDxe.efi
UFS INQUIRY ID: SAMSUNG KM2L9001CM-B518 0700
UFS Boot LUN: 1
Protective MBR validation might be needed.
Protective MBR validation might be needed.
- 0x09E80F000 [ 1757] Fat.efi
- 0x09D9B0000 [ 1757] TzDxeLA.efi
- 0x09D90F000 [ 1759] VariableDxe.efi
ConfigStorPartitions: Status:Success
<VariablePolicyInitialize:179> variable policy engine disabled
- 0x09E807000 [ 1877] QcomWDogDxe.efi
HW Wdog Setting from PCD : Disabled
- 0x09D927000 [ 1878] WatchdogTimer.efi
- 0x09D6D4000 [ 1878] SecurityStubDxe.efi
- 0x09E821000 [ 1878] EmbeddedMonotonicCounter.efi
- 0x09D91E000 [ 1880] RealTimeClock.efi
- 0x09D6AD000 [ 1881] DevicePathDxe.efi
- 0x09D6CC000 [ 1881] CapsuleRuntimeDxe.efi
- 0x09D666000 [ 1882] HiiDatabase.efi
- 0x09D6C3000 [ 1882] FontDxe.efi
- 0x09D58E000 [ 1887] QcomBds.efi
- 0x09D63F000 [ 1895] VerifiedBootDxe.efi
- 0x09D65A000 [ 1896] DDRInfoDxe.efi
- 0x09D626000 [ 1897] FeatureEnablerDxe.efi
QseeResponse->result = 0xFFFFFFFF
Status = 0x7
QseeResponse->result = 0xFFFFFFFF
```

```
Status = 0x7
Image partition label not found
EFI_QseecomStartApp: Load from partition (featenabler_a)Failed:
Status(0x800000000000000E), appId(0)
QseeResponse->result = 0xFFFFFFFF
Status = 0x7
QseeResponse->result = 0xFFFFFFFF
Status = 0x7
Image partition label not found
EFI_QseecomStartApp: Load from partition (featenabler)Failed:
Status(0x800000000000000E), appId(0)
Image partition label not found
LoadImageFromPartitionUsingGuid Failed: 14
EFI_QseecomStartApp: Load Failed: Status(0x800000000000000E)
Failed to start featenabler_a TA, status = 14
- 0x09D4EB000 [ 1901] DisplayDxe.efi
DisplayDxe: SW renderer mode enabled!
DisplayDxe: Panel ID:0x00000000 [LCD]
DisplayDxe: Resolution 640x480 (1 intf)
- 0x09D5DA000 [ 1912] FvDxe.efi
- 0x09D60B000 [ 1912] PILProxyDxe.efi
- 0x09D555000 [ 1913] PILDxe.efi
- 0x09D5CF000 [ 1934] IPCCDxe.efi
- 0x09D571000 [ 1934] GlinkDxe.efi
smem_alloc_ex: SMEM alloc_ex failed with err=-3! smem_type=478,
remote=3, size=32, flags=0x40000000. - 0x09D4DB000 [ 1935] CPRDxe.efi
- 0x09D586000 [ 1935] PrintDxe.efi
- 0x09D540000 [ 1936] SPI.efi
- 0x09D4C2000 [ 1936] PmicGlinkDxe.efi
- 0x09D4D1000 [ 1937] UsbPwrCtrlDxe.efi
- 0x09D483000 [ 1937] QcomChargerDxeLA.efi
- 0x09D54D000 [ 1947] ChargerExDxe.efi
- 0x09D46A000 [ 1947] UsbfnDwc3Dxe.efi
- 0x09D4AB000 [ 1948] XhciPciEmulation.efi
- 0x09D43B000 [ 1948] XhciDxe.efi
- 0x09D429000 [ 1949] UsbBusDxe.efi
- 0x09D41C000 [ 1949] UsbKbDxe.efi
- 0x09D4B6000 [ 1950] UsbMassStorageDxe.efi
- 0x09D406000 [ 1950] UsbConfigDxe.efi
UsbConfigInit: USB RUMI 0, ver 65536, sub 1
UsbConfigLibOpenProtocols: PMI version (0x0)
UsbConfigLibOpenProtocols: gPmicNpaClientHS2 cannot be created
UsbConfigInit: Dual Role Enabled on Port Number: 0
UsbConfigInit: after setting role
```



```

    UsbConfigInit: UsbConfigInit, not start on port: 0, mode 0
    UsbConfigInit: Dual Role Enabled on Port Number: 1
    UsbConfigInit: after setting role
    UsbConfigInit: UsbConfigInit, not start on port: 1, mode 0
    UsbConfigInit: UsbPwrCtrl No. of Ports = pinctrl-1

UsbConfigPortsQueryConnectionChange: usbport->connectstate: ATT
ConnectSts : Attach, Data Role : UFP (DEVICE Mode), Lane : CC1,
CoreNum : 0
HandlePortPartnerXtach: Cable Attach core 0, portmode 1, dual mode 1
- 0x09D455000 [ 1954] ButtonsDxe.efi
ButtonsDxeTest: Keypress SDAM data payload 0
- 0x09D462000 [ 1956] TsensDxe.efi
- 0x09D3FE000 [ 1958] LimitsDxe.efi
- 0x09D3D7000 [ 1962] GpiDxe.efi
- 0x09D390000 [ 2010] UCDxe.efi
- 0x09D3C6000 [ 2014] RngDxe.efi
- 0x09D3BE000 [ 2015] SimpleTextInOutSerial.efi
- 0x09D386000 [ 2016] ConPlatformDxe.efi
- 0x09D369000 [ 2016] ConSplitterDxe.efi
- 0x09D35D000 [ 2017] GraphicsConsoleDxe.efi
- 0x09D37C000 [ 2018] ASN1X509Dxe.efi
- 0x09D3B3000 [ 2019] SecRSADxe.efi
- 0x09D348000 [ 2019] DtPlatformDxe.efi
    DtPlatformLoadDtb XXXxxxxx-rb3.dtb is loaded
    DtPlatformDxeEntryPoint: no DT/ACPI preference found, defaulting to
DTB
- 0x09D2A6000 [ 2030] SoftSKUDxe.efi
    SoftSKUDxeInitialize: SoftSKU not supported for this chip
    Error: Image at 0009D2A6000 start failed: Unsupported
- 0x09D2A6000 [ 2033] MinidumpTADxe.efi
    Image partition label not found
    LoadImageFromPartitionUsingGuid Failed: 14
    EFI_QseecomStartApp: Load Failed: Status(0x8000000000000000E)
    MinidumpTALib:LoadImageFromPartition(mdcompress) failed: 0xNot
Found
    MinidumpTADxe: Minidump TA loading failed.
    Error: Image at 0009D2A6000 start failed: Not found
- 0x09D294000 [ 2037] UsbMsDxe.efi
- 0x09D287000 [ 2038] UsbDeviceDxe.efi
- 0x09D2A7000 [ 2038] UsbInitDxe.efi
- 0x09D27E000 [ 2039] ParserDxe.efi
- 0x09D277000 [ 2040] SerialPortDxe.efi
BDS Entry      [ 2040]
```

```
Disp init wait [ 2040]
-----
Platform Init [ 2057] BDS
Boot Cycle : 1
Run Cycle : 1
UEFI Ver : 6.0.231205.BOOT.<XXX.x.x.Xx-xxxxx-XXXXXXX-x>
Platform : IOT
Subtype : 1
Boot Device : UFS
Chip Name : <XXXxxxx>
Chip Ver : 1.0
Chip Serial Number : <0xXxxxxxxx>
Boot Core : 0 AppsProcClkMHz
-----
- 0x09D21C000 [ 2114] QcomChargerApp.efi
Protective MBR validation might be needed.
Protective MBR validation might be needed.
ERROR: A bit not set !
AppsProcClkMHz is zero
Firmware update failed - Status: Unsupported
Firmware provisioning failed
Platform Init End : 2282
-----
[QcomBds] BootOrder not found
Protective MBR validation might be needed.
Protective MBR validation might be needed.
[QcomBds] Enumerating non-removable boot options
[QcomBds] Adding new boot/driver option Boot0000, Description: Non-removable Media
Boot option 0:(Boot0000) "Non-removable Media"
UEFI Total : 886 ms
POST Time [ 2454] OS Loader
- 0x1C22E0000 [ 2456]
- 0x1D1F80000 [ 2655]
- 0x1FDD00000 [ 2688]
EFI stub: Booting Linux Kernel...
EFI stub: Loaded initrd from LINUX_EFI_INITRD_MEDIA_GUID device path
EFI stub: Using DTB from configuration table
EFI stub: Exiting boot services...
Start EBS [ 2717]
MDPUpdateDynamicClocks: Clock management not supported!
MDPUpdateCoreClockAndBandwidth: MDPUpdateDynamicClocks failed!
MDPLib: MDPUpdateCoreClockAndBandwidth failed!
```

```

MDPUpdateDynamicClocks: Clock management not supported!
MDPUpdateCoreClockAndBandwidth: MDPUpdateDynamicClocks failed!
MDPLib: MDPUpdateCoreClockAndBandwidth failed!
MDPLib: SW Render enabled, no Clocks disabled need
MDPLib: SW Render enabled, no Clocks disabled need
MDPLib: SW Render enabled, no Clocks disabled need
MDPLib: SW Render enabled, no Clocks disabled need
Successfully synced all UEFI tables
Sync Duration = 40 ms
Warning: Clearing A-bit !
App Log Flush : 315 ms
ScmArmV8ExitBootServicesHandler, Status = 0x0.
Exit EBS
``[ 3086] UEFI End``

```

11.3 Linux log

The sample log signifies the start of the Linux boot process.

Use this log entry to verify if the Linux boot process has finished successfully.

```

[ 0.000000] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
[ 0.000000] Linux version 6.6.0 (oe-user@oe-host) (aarch64-qcom-
linux-gcc (GCC) 11.4.0, GNU ld (GNU Binutils) 2.38.20220708) #1 SMP
PREEMPT <dayXX> <monthXX>
[ 0.000000] KASLR enabled
[ 0.000000] Machine model: Qualcomm Technologies, Inc. XXXxxxx-addons
XXx platform
[ 0.000000] efi: EFI v2.7 by Qualcomm Technologies, Inc.

```

The following log is a sample Linux bootup log:

```

[ 0.000000] Booting Linux on physical CPU 0x0000000000
[0x412fd050]
[ 0.000000] Linux version 6.6.0 (oe-user@oe-host) (aarch64-qcom-
linux-gcc (GCC) 11.4.0, GNU ld (GNU Binutils) 2.38.20220708) \#1 SMP
PREEMPT <dayXX> <monthXX> <timexxx> UTC <yearxxxx>\>****
[ 0.000000] KASLR enabled
[ 0.000000] Machine model: Qualcomm Technologies, Inc. XXXxxxx-
addons XXx platform
[ 0.000000] efi: EFI v2.7 by Qualcomm Technologies, Inc.``
[ 0.000000] efi: MEMATTR=0x9d233018 INITRD=0x9d228718
RNG=0x9d222018 MEMRESERVE=0x9d228218

```

```

[ 0.000000] random: crng init done
[ 0.000000] Reserved memory: created CMA memory pool at
0x00000000ff000000, size 12 MiB
[ 0.000000] OF: reserved mem: initialized node adsp-heap,
compatible id shared-dma-pool
[ 0.000000] OF: reserved mem: 0x00000000ff000000..
0x00000000fffbffff (12288 KiB) map reusable adsp-heap
[ 23.653629] qnoc-sc7280 1580000.interconnect: sync_state() pending
due to 3d00000.qcom,kgs1-3d0
[ 23.662613] qnoc-sc7280 1580000.interconnect: sync_state() pending
due to 3d00000.gpu
[ 23.670698] qnoc-sc7280 9100000.interconnect: sync_state() pending
due to 3d00000.gpu
[ 23.678776] qnoc-sc7280 9100000.interconnect: sync_state() pending
due to 3d00000.qcom,kgs1-3d0
[ 23.687743] qnoc-sc7280 1580000.interconnect: sync_state() pending
due to aa00000.video-codec
[ 23.696524] qnoc-sc7280 1740000.interconnect: sync_state() pending
due to aa00000.video-codec
[ 23.705308] qnoc-sc7280 1500000.interconnect: sync_state() pending
due to aa00000.video-codec
[ 23.714081] qnoc-sc7280 9100000.interconnect: sync_state() pending
due to aa00000.video-codec
[ 23.722895] qcom-rpmhpd 18200000.rsc:power-controller: sync_
state() pending due to aa00000.video-codec
[ 33.874665] refgen: disabling

```

11.4 Capsule update logs

The capsule update log provides information about the start and end of the capsule update.

The UEFI firmware processes the capsule, which contains the firmware update payload along with the metadata, during the system boot sequence.

The documentation specifies three example logs for the different stages of the capsule update. See the following logs for more details:

UEFI stage: Start capsule update

- **Log entry:** At the UEFI stage, the capsule loads the mass-storage capsule file capsule1.fv'! and starts to process mass-storage capsule file capsule1.fv'!.
- **Description:** The log indicates that the system has detected a capsule. The system triggers and processes the capsule update.
- **Log file:**

```
UEFI Start      [ 1570]
  - 0x09FC01000 [ 1573] Sec.efi
ASLR            : ON
DEP             : ON (RTB)
Timer Delta    : +1 mS
RAM Entry 0    : Base 0x0080000000 Size 0x003A800000
RAM Entry 1    : Base 0x00C0000000 Size 0x0001800000
RAM Entry 2    : Base 0x00C3400000 Size 0x003CC00000
RAM Entry 3    : Base 0x0100000000 Size 0x0100000000
Total Available RAM : 6028 MB (0x0178C00000)
Total Installed RAM : 6144 MB (0x0180000000)
Multithread     : ON (Lib ver 1.2)
Init Cores      : 2
Init CPU core 1
  > Scheduler up on Core 1
CONF File       : uefiplatLA.cfg
UEFI Ver        : 6.0.xxx.xxx.xxx.1.0.c1-xxxx.1-KODIAKLA-
1
Build Info      : 64b Nov 27 2024 15:04:28
Boot Device     : UFS
PROD Mode       : FALSE
Retail          : FALSE

DTB config      : client[0]..trace[0]..verbose[0]
  - 0x09EDD0000 [ 1628] DxeCore.efi
Loading DxeCore at 0x009EDD0000
EntryPoint=0x009EDD1000
HOBList address in DXE = 0x9EBC1018
FV Hob          0x9FC00000 - 0x9FFFFFFFFF
FV Hob          0x9EE0F000 - 0x9F1DBFFF
FV2 Hob         0x9EE0F000 - 0x9F1DBFFF
                631008B0-B2D1-410A-8B49-
2C5C4D8ECC7E - 00000000-0000-0000-0000-000000000000

>>> Cluster 0: 1804800000 Hz
```

```
>>> Cluster L3: 1516800000 Hz
PM0: 47, PM1: 63, PM2: 49, PM8: 46,
Module cannot re-initialize DAL module environment
UFS INQUIRY ID: SAMSUNG KM2L9001CM-B518 0700
UFS Boot LUN: 1
tz_armv8_smc_call failed, TzStatus = 0xFFFFFFFF,
SmcId = 0x32000105
Status = 0x3
APP_REGION_NOTIFICATION_CMD: Syscall Not Supported,
status 3
QseeResponse->result = 0xE
Status = 0x7
QseeResponse->result = 0xE
Status = 0x7
Image partition label not found
EFI_QseecomStartApp: Load from partition
(featenabler_a)Failed: Status(0x800000000000000E),
appId(0)
QseeResponse->result = 0xE
Status = 0x7
QseeResponse->result = 0xE
Status = 0x7
Image partition label not found
EFI_QseecomStartApp: Load from partition
(featenabler)Failed: Status(0x800000000000000E),
appId(0)
Image partition label not found
LoadImageFromPartitionUsingGuid Failed: 14
EFI_QseecomStartApp: Load Failed:
Status(0x800000000000000E)
Failed to start featenabler_a TA, status = 14
- 0x09CA2C000 [ 1836] DisplayDxe.efi
DisplayUtils: mount logfs system failed with status
14!
DisplayUtils: mount logfs system failed with status
14!
DisplayUtils: mount logfs system failed with status
14!
DisplayDxe: Panel ID:0x00000000 [LCD]
DisplayDxe: Resolution 1920x1080 (1 intf)
pil-imagefv Done t=10ms s=2ms a=3ms f=3ms

fast_core_num val: 0
UsbConfigLibOpenProtocols: PMI version (0x0)
```

```
UsbConfigLibOpenProtocols: gPmicNpaClientHS2 cannot
be created
UsbConfigInit: Dual Role Enabled on Port Number: 0
UsbConfigInit: after setting role
UsbConfigInit: UsbConfigInit, not start on port: 0,
mode 0
UsbConfigInit: Dual Role Enabled on Port Number: 1
UsbConfigInit: after setting role
UsbConfigInit: UsbConfigInit, not start on port: 1,
mode 0
UsbConfigInit: UsbPwrCtrl No. of Ports = 1

UsbConfigPortsQueryConnectionChange: usbport->
connectstate: ATT
ConnectSts : Attach, Data Role : UFP (DEVICE Mode),
Lane : CC1, CoreNum : 0
HandlePortPartnerXtach: Cable Attach core 0, portmode
1, dualmode 1
ButtonsDxeTest: Keypress SDAM data payload 0
    Current SysFwVersion Entry info:
    =====
    Signature                : 5245565746535953
    Revision                  : 0x10000
    VersionDataCrc32          : 0xD6AE50A5
    FwVersion                  : 0x10000
    LowestSupportedFwVersion  : 0x0
    =====
Platform: QCS6490 IOT
Selected FW GUID =: 6F25BFD2-A165-468B-980F-
AC51A0A45C52
    Platform = 20
    - 0x09C885000 [ 1981] SoftSKUDxe.efi
SoftSKUDxeInitialize: SoftSKU not supported for this
chip
Error: Image at 0009C885000 start failed: Unsupported
    - 0x09C885000 [ 1984] MinidumpTADxe.efi
MinidumpTADxe: Minidump TA loading not enabled.
BDS Entry          [ 1993]
Disp init wait [ 1993]
DisplayUtils: mount logfs system failed with status
14!
MDP_Display_LoadFileEx: Failed to mount imagefv_a
partition, eStatus=Not Found
MDP_Display_LoadFileEx: load file from imagefv_a
```

```
failed with status(5), So loading default image!
-----
Platform Init [ 2065] BDS
Boot Cycle : 34
Run Cycle : 34
UEFI Ver : 6.0.xxxxx.xxxx.xxx.1.0.c1-00xxx-xxxx-1
Platform : IOT
Subtype : 2
Boot Device : UFS
Chip Name : xxxxx
Chip Ver : 1.0
Chip Serial Number : 0xDF4B18A2
Boot Core : 1804 MHz
-----
- 0x09C70F000 [ 2083] QcomChargerApp.efi
EFI partition block size: 4096
No pending capsules found in EFI Raw file
Starting to flush any pending persisted capsules from
media
Could not find 'BootOrder' variable to find EFI
partition with mass-storage capsules!
Looking for local EFI partition variable to find EFI
partition with mass-storage capsules
Loading mass-storage capsule file 'capsule1.fv'!
Starting to process mass-storage capsule file
'capsule1.fv'!
FmpDxe(Qualcomm System Firmware Update Driver):
CheckTheImage() - No dependency associated in image.
    V3 payload found
    V3 payload found
    NewImage Version
- 0x30004
    NewImage LowestSupportedImageVersion
- 0x0
    Current Version (partition)
- 0x10000
    Current LowestSupportedImageVersion (partition)
- 0x0
```


Firmware backup, update and post update

- **Log entry:**

- Phase 1: The firmware entry backup starts and phase 1 is complete.
- Phase 2: The firmware entry update starts and phase 2 is complete.
- Phase 3: The firmware entry post-update starts and phase 3 is complete.

The firmware update is completed successfully.

- **Description:** The markers for successful firmware update are provided in the log file. The phases described in the firmware update are as indicated in the log entries.

- **Log file:**

```
Attempting to start: Firmware update

Current SysFwVersion Entry info:
=====
Signature                : 5245565746535953
Revision                  : 0x10000
VersionDataCrc32          : 0xD6AE50A5
FwVersion                  : 0x10000
LowestSupportedFwVersion  : 0x0
=====

Current FW update attempt count is: 1
Current FW update state is: FW_STATE_NOT_IN_PROGRESS
Successfully synced all UEFI tables
Rendering Progress: 1

Parsing payload...
V3 payload found
Payload info:
=====
Revision                  : 3
Header size (Byte)        : 24
FwVer                     : 0x30004
Lowest supported version  : 0x0
Breaking change number    : 0
Entry count                : 10
=====

Current breaking change number is 0
Payload breaking change number is 0
Double update not required
```

```
ACTIVE LUN: 1

1 FwEntry found...
FwEntry:
=====
    Operation              = 0x1, {UPDATE}

    UpdateType              = 0x0, {Partition}
    DiskPartitionType       = 0xC, {LUN4}
    PartitionName           = imagefv_a
    FileName                =

    BackupType              = 0x0, {Partition}
    DiskPartitionType       = 0xC, {LUN4}
    PartitionName           = imagefv_b
    FileName                =

    FwEntry Encrypted = False
=====

2 FwEntry found...
FwEntry:
=====
    Operation              = 0x1, {UPDATE}

    UpdateType              = 0x0, {Partition}
    DiskPartitionType       = 0xC, {LUN4}
    PartitionName           = uefi_a
    FileName                =

    BackupType              = 0x0, {Partition}
    DiskPartitionType       = 0xC, {LUN4}
    PartitionName           = uefi_b
    FileName                =

    FwEntry Encrypted = False
=====

3 FwEntry found...
FwEntry:
=====
    Operation              = 0x1, {UPDATE}
```

```
UpdateType           = 0x0, {Partition}
  DiskPartitionType   = 0xC, {LUN4}
  PartitionName       = aop_a
  FileName            =
```

```
BackupType           = 0x0, {Partition}
  DiskPartitionType   = 0xC, {LUN4}
  PartitionName       = aop_b
  FileName            =
```

```
FwEntry Encrypted = False
```

```
=====
```

```
4 FwEntry found...
```

```
FwEntry:
```

```
=====
```

```
Operation            = 0x1, {UPDATE}
```

```
UpdateType           = 0x0, {Partition}
  DiskPartitionType   = 0xC, {LUN4}
  PartitionName       = hyp_a
  FileName            =
```

```
BackupType           = 0x0, {Partition}
  DiskPartitionType   = 0xC, {LUN4}
  PartitionName       = hyp_b
  FileName            =
```

```
FwEntry Encrypted = False
```

```
=====
```

```
5 FwEntry found...
```

```
FwEntry:
```

```
=====
```

```
Operation            = 0x1, {UPDATE}
```

```
UpdateType           = 0x0, {Partition}
  DiskPartitionType   = 0xC, {LUN4}
  PartitionName       = tz_a
  FileName            =
```

```
BackupType           = 0x0, {Partition}
  DiskPartitionType   = 0xC, {LUN4}
  PartitionName       = tz_b
```

```
        FileName          =

        FwEntry Encrypted = False
=====

6 FwEntry found...
FwEntry:
=====
        Operation          = 0x1, {UPDATE}

        UpdateType         = 0x0, {Partition}
        DiskPartitionType   = 0xC, {LUN4}
        PartitionName       = cpucp_a
        FileName           =

        BackupType         = 0x0, {Partition}
        DiskPartitionType   = 0xC, {LUN4}
        PartitionName       = cpucp_b
        FileName           =

        FwEntry Encrypted = False
=====

7 FwEntry found...
FwEntry:
=====
        Operation          = 0x1, {UPDATE}

        UpdateType         = 0x0, {Partition}
        DiskPartitionType   = 0xC, {LUN4}
        PartitionName       = shrm_a
        FileName           =

        BackupType         = 0x0, {Partition}
        DiskPartitionType   = 0xC, {LUN4}
        PartitionName       = shrm_b
        FileName           =

        FwEntry Encrypted = False
=====

8 FwEntry found...
FwEntry:
=====
```

```

Operation                = 0x1,{UPDATE}

UpdateType               = 0x0,{Partition}
  DiskPartitionType      = 0xC,{LUN4}
  PartitionName          = multiimgoem_a
  FileName               =

BackupType               = 0x0,{Partition}
  DiskPartitionType      = 0xC,{LUN4}
  PartitionName          = multiimgoem_b
  FileName               =

```

FwEntry Encrypted = False

=====

9 FwEntry found...

FwEntry:

=====

```

Operation                = 0x1,{UPDATE}

UpdateType               = 0x0,{Partition}
  DiskPartitionType      = 0xC,{LUN4}
  PartitionName          = xbl_ramdump_a
  FileName               =

BackupType               = 0x0,{Partition}
  DiskPartitionType      = 0xC,{LUN4}
  PartitionName          = xbl_ramdump_b
  FileName               =

```

FwEntry Encrypted = False

=====

10 FwEntry found...

FwEntry:

=====

```

Operation                = 0x1,{UPDATE}

UpdateType               = 0x0,{Partition}
  DiskPartitionType      = 0xC,{LUN4}
  PartitionName          = uefisecapp_a
  FileName               =

BackupType               = 0x0,{Partition}

```

```
DiskPartitionType = 0xC, {LUN4}
PartitionName      = uefisecapp_b
FileName           =

FwEntry Encrypted = False
=====

Validating FwEntry list...

FwEntry list validated. Totally 10 FwEntries found
Rendering Progress: 5
Successfully synced all UEFI tables
: Failed to get ResetPhase test hook data with error: Not
Found

Phase 1: FwEntry Backup start. Time (ms): 2631

Rendering Progress: 6
    GPT-Data Compression Statistics:: OriginalSize: 5900,
CompressedSize: 594

Successfully synced all UEFI tables
    Backing up 1/10 FwEntry...
        Backup Success

Progress span of current phase: [6, 50]. Progress value
within this span in (0-100 scale): 1
Rendering Progress: 6
    Backing up 2/10 FwEntry...
        Backup Success

Progress span of current phase: [6, 50]. Progress value
within this span in (0-100 scale): 3
Rendering Progress: 7
    Backing up 3/10 FwEntry...
        Backup Success

Progress span of current phase: [6, 50]. Progress value
within this span in (0-100 scale): 3
Rendering Progress: 7
    Backing up 4/10 FwEntry...
        Backup Success

Progress span of current phase: [6, 50]. Progress value
```

```
within this span in (0-100 scale): 4
Rendering Progress: 7
    Backing up 5/10 FwEntry...
        Backup Success

Progress span of current phase: [6, 50]. Progress value
within this span in (0-100 scale): 5
Rendering Progress: 8
    Backing up 6/10 FwEntry...
        Backup Success

Progress span of current phase: [6, 50]. Progress value
within this span in (0-100 scale): 45
Rendering Progress: 25
    Backing up 7/10 FwEntry...
        Backup Success

Progress span of current phase: [6, 50]. Progress value
within this span in (0-100 scale): 59
Rendering Progress: 31
    Backing up 8/10 FwEntry...
        Backup Success

Progress span of current phase: [6, 50]. Progress value
within this span in (0-100 scale): 61
Rendering Progress: 32
    Backing up 9/10 FwEntry...
        Backup Success

Progress span of current phase: [6, 50]. Progress value
within this span in (0-100 scale): 99
Rendering Progress: 49
    Backing up 10/10 FwEntry...
        Backup Success

Progress span of current phase: [6, 50]. Progress value
within this span in (0-100 scale): 100
Rendering Progress: 50
Successfully synced all UEFI tables
Rendering Progress: 50

    Phase 1 Done. Time (ms): 3048
: Failed to get ResetPhase test hook data with error: Not
Found
```

```
Phase 2: FwEntry Update start. Time (ms): 3050

Rendering Progress: 51
    Updating 1/10 FwEntry...
Progress span of current phase: [51, 90]. Progress value
within this span in (0-100 scale): 1
Rendering Progress: 51
    Update Success

    Updating 2/10 FwEntry...
Progress span of current phase: [51, 90]. Progress value
within this span in (0-100 scale): 3
Rendering Progress: 52
    Update Success

    Updating 3/10 FwEntry...
Progress span of current phase: [51, 90]. Progress value
within this span in (0-100 scale): 3
Rendering Progress: 52
    Update Success

    Updating 4/10 FwEntry...
Progress span of current phase: [51, 90]. Progress value
within this span in (0-100 scale): 4
Rendering Progress: 52
    Update Success

    Updating 5/10 FwEntry...
Progress span of current phase: [51, 90]. Progress value
within this span in (0-100 scale): 5
Rendering Progress: 52
    Update Success

    Updating 6/10 FwEntry...
Progress span of current phase: [51, 90]. Progress value
within this span in (0-100 scale): 45
Rendering Progress: 68
    Update Success

    Updating 7/10 FwEntry...
Progress span of current phase: [51, 90]. Progress value
within this span in (0-100 scale): 59
Rendering Progress: 74
```



```
Update Success

Updating 8/10 FwEntry...
Progress span of current phase: [51, 90]. Progress value
within this span in (0-100 scale): 61
Rendering Progress: 74
Update Success

Updating 9/10 FwEntry...
Progress span of current phase: [51, 90]. Progress value
within this span in (0-100 scale): 99
Rendering Progress: 89
Update Success

Updating 10/10 FwEntry...
Progress span of current phase: [51, 90]. Progress value
within this span in (0-100 scale): 100
Rendering Progress: 90
Update Success

Successfully synced all UEFI tables
Rendering Progress: 90

Phase 2 Done. Time (ms): 3190
: Failed to get ResetPhase test hook data with error: Not
Found

Phase 3: FwEntry Post Update start. Time (ms): 3192

Rendering Progress: 91
Deleting 1/10 FwEntry from backup device path...
Delete skipped as delete operation not applicable
for this FwEntry...Continue

Deleting 2/10 FwEntry from backup device path...
Delete skipped as delete operation not applicable
for this FwEntry...Continue

Deleting 3/10 FwEntry from backup device path...
Delete skipped as delete operation not applicable
for this FwEntry...Continue

Deleting 4/10 FwEntry from backup device path...
Delete skipped as delete operation not applicable
```

```
for this FwEntry...Continue

    Deleting 5/10 FwEntry from backup device path...
    Delete skipped as delete operation not applicable
for this FwEntry...Continue

    Deleting 6/10 FwEntry from backup device path...
    Delete skipped as delete operation not applicable
for this FwEntry...Continue

    Deleting 7/10 FwEntry from backup device path...
    Delete skipped as delete operation not applicable
for this FwEntry...Continue

    Deleting 8/10 FwEntry from backup device path...
    Delete skipped as delete operation not applicable
for this FwEntry...Continue

    Deleting 9/10 FwEntry from backup device path...
    Delete skipped as delete operation not applicable
for this FwEntry...Continue

    Deleting 10/10 FwEntry from backup device path...
    Delete skipped as delete operation not applicable
for this FwEntry...Continue

Successfully synced all UEFI tables
Successfully synced all UEFI tables
Successfully synced all UEFI tables
Rendering Progress: 95

    Phase 3 Done. Time (ms): 3230
Successfully synced all UEFI tables
Successfully synced all UEFI tables
Firmware update completed successfully
Attempting to update SysFw Partition Entry
```

Delete mass storage capsule file

- **Log entry:** Deleting mass-storage in the capsule file: capsule1.fv'!.
- **Description:** The system deletes the capsule after successfully processing the firmware update.
- **Log file:**

```

Writing updated SysFwVer details into SysFwVer partition...

Current SysFwVersion Entry info:
=====
Signature                : 5245565746535953
Revision                  : 0x10000
VersionDataCrc32          : 0x1F2CFF54
FwVersion                  : 0x30004
LowestSupportedFwVersion  : 0x0
=====
Successfully synced all UEFI tables
Updated SysFw Partition Entry
Platform: QCS6490 IOT
Selected FW GUID =: 6F25BFD2-A165-468B-980F-AC51A0A45C52
Platform = 20
Successfully synced all UEFI tables
Deleting mass-storage capsule file 'capsule1.fv'!
Starting to flush any pending persisted capsules from media
!!Warning!! Could not clear 'BootNext' variable after
processing mass-storage capsules. Status - Not Found!
Write Log Buffer to FAT partition after capsule update
failed, Status = (Unsupported)

INFO: Received ResetSystem request.
INFO: Type      :0x0
INFO: DataSize  :0x0

Start EBS          [ 4674]
MDPLib: Halt vbif axi failed!
BDS: LogFs sync skipped, Unsupported
Successfully synced all UEFI tables
Sync Duration = 16 ms
Warning: Clearing A-bit !
App Log Flush : 0 ms
ScmArmV8ExitBootServicesHandler, Status = 0x0.
Gunyah based bootup
Exit EBS          [ 4719] UEFI End

```

```
UEFI Start      [ 1587]
- 0x09FC01000 [ 1590] Sec.efi
ASLR           : ON
DEP            : ON (RTB)
Timer Delta    : +0 mS
RAM Entry 0    : Base 0x0080000000 Size 0x003A800000
RAM Entry 1    : Base 0x00C0000000 Size 0x0001800000
RAM Entry 2    : Base 0x00C3400000 Size 0x003CC00000
RAM Entry 3    : Base 0x0100000000 Size 0x0100000000
Total Available RAM : 6028 MB (0x0178C00000)
Total Installed RAM : 6144 MB (0x0180000000)
Init CPU core 1
  > Scheduler up on Core 1
UEFI Ver       : 6.0.xxxx.xx.xxx.1.0.c1-xxx-xxx-1
Build Info     : 64b Nov  5 2024 10:08:42
Boot Device    : UFS
PROD Mode      : TRUE
Retail         : TRUE
After Create Event Status (0x0)
ShmBridgeInitialize: enable status 0
PM0: 47, PM1: 63, PM2: 49, PM8: 46,
Module cannot re-initialize DAL module environment
UFS INQUIRY ID: SAMSUNG KM2L9001CM-B518 0700
UFS Boot LUN: 1
tz_armv8_smc_call failed, TzStatus = 0xFFFFFFFF, SmcId =
0x32000105
Status = 0x3
APP_REGION_NOTIFICATION_CMD: Syscall Not Supported, status 3
HW Wdog Setting from PCD : Disabled
QseeResponse->result = 0xE
Status = 0x7
QseeResponse->result = 0xE
Status = 0x7
QseeResponse->result = 0xE
Status = 0x7
QseeResponse->result = 0xE
Status = 0x7
LoadImageFromPartitionUsingGuid Failed: 14
Failed to start featenabler_a TA, status = 14
DisplayDxe: Resolution 1920x1080 (1 intf)
smem_alloc_ex: SMEM alloc_ex failed with err=-3! smem_
type=478, remote=3, size=32, flags=0x40000000.smem_alloc_ex:
```

```
SMEM alloc_ex failed with err=-3! smem_type=478, remote=12,
size=32, flags=0x40000000.smem_alloc_ex: SMEM alloc_ex failed
with err=-3! smem_type=478, remote=17, size=32,
flags=0x40000000.smem_alloc_ex: SMEM alloc_ex failed with err=-
3! smem_type=478, remote=18, size=32, flags=0x40000000.fast_
core_num val: 0
```

```
    UsbConfigLibOpenProtocols: PMI version (0x0)
```

```
    UsbConfigLibOpenProtocols: gPmicNpaClientHS2 cannot be
created
```

```
    UsbConfigInit: after setting role
```

```
    UsbConfigInit: UsbConfigInit, not start on port: 0, mode 0
```

```
    UsbConfigInit: after setting role
```

```
    UsbConfigInit: UsbConfigInit, not start on port: 1, mode 0
```

```
    UsbConfigPortsQueryConnectionChange: usbport->connectstate:
```

```
ATT
```

```
    ButtonsDxeTest: Keypress SDAM data payload 0
```

```
    HypDtFixupEntryPoint : Reading of OsConfigTableSelection
failed,checking DT setting
```

```
    Selected FW GUID =: 6F25BFD2-A165-468B-980F-AC51A0A45C52
```

```
    Platform = 20
```

```
    SoftSKUDxeInitialize: SoftSKU not supported for this chip
```

```
    MinidumpTADxe: Minidump TA loading not enabled.
```

```
    Disp init wait [ 1998]
```

```
-----
Platform Init [ 2063] BDS
```

```
UEFI Ver : 6.0.xxx.xxx.xxx.1.0.c1-xxxx-xxxx-1
```

```
Platform : IOT
```

```
Subtype : 2
```

```
Boot Device : UFS
```

```
Chip Name : xxxxx
```

```
Chip Ver : 1.0
```

```
Chip Serial Number : 0xDF4B18A2
```

```
-----
No pending capsules found in EFI Raw file
```

```
No pending capsules found in EFI\UpdateCapsule folder
```

```
LocateProtocol(DPP) returned Status:Not Found
```

```
Locate pMorPpiProtocol failed 0xE 00000010
```

```
Platform Init End : 2219
```

```
-----
Compatible OS DTB found. Model = Qualcomm Technologies, Inc.
Robotics RB3gen2 addons vision mezz platform
```

```
    Reading of OsConfigTableSelection failed,checking DT
settings
```

```
    ERROR: DisableDisplay disabling,Status=0
```

```
SdcccDxeFixupEventNotifyFunc: non-removable node not found
Added qcom,platform-parts-info successfully in BoardInfoDxe
Failed to get SKU information. Unsupported
UEFI Total : 854 ms
POST Time      [ 2441] OS Loader
systemd-boot@0x1c22e0000 254.4
systemd-stub@0x1d1f80000 254.4
EFI stub: Booting Linux Kernel...
EFI stub: Loaded initrd from LINUX_EFI_INITRD_MEDIA_GUID
device path
EFI stub: Using DTB from configuration table
EFI stub: Exiting boot services...
Start EBS      [ 2705]
BDS: LogFs sync skipped, Unsupported
App Log Flush : 0 ms
ScmArmV8ExitBootServicesHandler, Status = 0x0.
Gunyah based bootup
```

12 Examples on debugging scenarios

The examples show the debugging scenarios encountered during system bootup.

For information about Fastboot detection and commands, see [Fastboot](#).

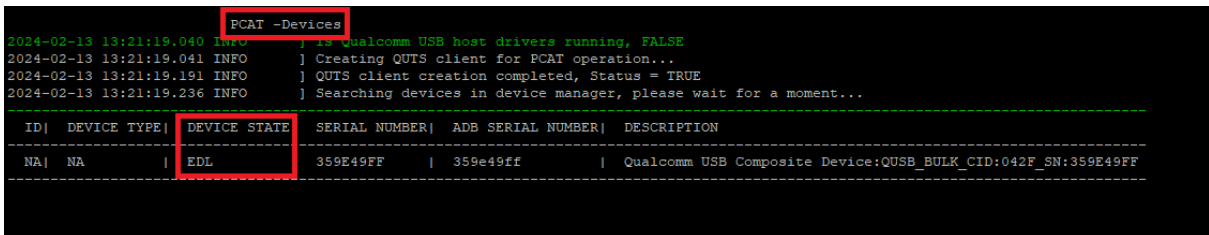
12.1 Enumeration of EDL device manager

If the storage is either empty or corrupted, the device switches to the EDL mode upon power-on.

To verify if the device is in EDL mode, run the following command:

```
PCAT -Devices
```

The figure shows the device state as EDL.



```
2024-02-13 13:21:19.040 INFO    ] Qualcomm USB host drivers running, FALSE
2024-02-13 13:21:19.041 INFO    ] Creating QUTS client for PCAT operation...
2024-02-13 13:21:19.191 INFO    ] QUTS client creation completed, Status = TRUE
2024-02-13 13:21:19.236 INFO    ] Searching devices in device manager, please wait for a moment...

-----
ID| DEVICE TYPE| DEVICE STATE| SERIAL NUMBER| ADB SERIAL NUMBER| DESCRIPTION
-----
NA| NA        | EDL        | 359E49FF    | 359e49ff        | Qualcomm USB Composite Device:QUSB_BULK_CID:042F_SN:359E49FF
-----
```

When EDL is detected, load the build using [Qualcomm Product Configuration Assistant Tool \(PCAT\)](#).

12.2 Detection of RAM dump

If a crash occurs, PCAT gathers the RAM dump and shows the logs for debugging.

For information about collecting and parsing the RAM dump, see [Debug Linux Kernel space > Collect and Parse RAM dump collection](#).

The figure shows the driver identifying the crash and displaying the USB logs:

```
PCAT --PLUGIN CC -DEVICE 359E49FF -DUMPPATH "\\local\mnt\workspace\dump" -RESET TRUE -SKIP8K FALSE -SKIP9K TRUE -UNIQUETS TRUE
2024-02-13 13:16:36.542 INFO      ] Creating QUTS client for PCAT operation...
2024-02-13 13:16:36.689 INFO      ] QUTS client creation completed, Status = TRUE
2024-02-13 13:16:36.727 INFO      ] Searching device "359E49FF" in device manager
2024-02-13 13:16:36.728 INFO      ] Searching for the device "359E49FF" in QUTS device manager list
2024-02-13 13:16:37.811 INFO      ] Device "359E49FF" found in QUTS device manager list
2024-02-13 13:16:37.816 INFO      ] Downloading memory dump from the device - Qualcomm USB Composite Device:QUSB_BULK_SN:359E49FF
2024-02-13 13:16:37.819 INFO      ] Device - Qualcomm USB Composite Device:QUSB_BULK_SN:359E49FF not connected to other PCAT instance
2024-02-13 13:16:37.860 INFO      ] Memory dump Path - \\local\mnt\workspace\dump\359E49FF\20240213-131637
2024-02-13 13:16:37.860 INFO      ] Reset device - TRUE
2024-02-13 13:16:37.860 INFO      ] Skip 8K memory dump collection - FALSE
2024-02-13 13:16:37.860 INFO      ] Skip 9K memory dump collection - TRUE
2024-02-13 13:16:37.860 INFO      ] Unique timestamped folder - TRUE
2024-02-13 13:16:37.860 INFO      ] Is Fusion Device - FALSE
2024-02-13 13:16:37.860 INFO      ] Downloading 8K memory dump from the device - Qualcomm USB Composite Device:QUSB_BULK_SN:359E49FF is in progress..., Please wait for completion
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 4% ... /
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 8% ... /
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 12% ... /
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 16% ... /
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 20% ... /
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 24% ... /
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 28% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 32% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 36% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 40% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 44% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 48% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 52% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 56% ... \
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 60% ... \
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 64% ... \
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 68% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 72% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 76% ... /
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 80% ... |
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 84% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 88% ... |
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 92% ... -
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 96% ... |
2024-02-13 13:16:37.860 INFO      ] The memory dump is in progress 99% ... |
2024-02-13 13:17:21.155 INFO      ] Downloaded 8K memory dump from the device - Qualcomm USB Composite Device:QUSB_BULK_SN:359E49FF, resetting device now
2024-02-13 13:17:23.167 ERROR      ] Status - FALSE
2024-02-13 13:17:23.198 ERROR      ] Response - Failed to reset the device - Qualcomm USB Composite Device:QUSB_BULK_SN:359E49FF, Status - DEVICE_UNKNOWN_ERROR, Response -
$ [ ]
```


13 References

Related documents:

Title	Document number
Document	
Qualcomm Linux Build Guide	80-70018-254
Dev Kit User Guide	80-70018-251
Qualcomm Linux Security Guide	80-70018-11
Resources	
TianocoreEDK2 open-source implementation	https://github.com/tianocore/tianocore.github.io/wiki/EDK-II/
systemd-boot	https://www.freedesktop.org/software/systemd/man/latest/systemd-boot.html
EFI boot stub	https://docs.kernel.org/admin-guide/efi-stub.html
UEFI specification – DXE phase	https://uefi.org/specs/PI/1.8/V2_Overview.html
Qualcomm package manager (QPM)	https://qpm.qualcomm.com/#/main/tools/details/QPM3

Table : Acronyms

Acronym	Definition
AOP	Always-on processor
APPS	Applications
cDSP	Compute DSP
CDT	Configuration data table
CLK	Clock
CPUCP	CPU subsystem control processor
DDR	Double data rate
DEVCFG	Device configuration
DSC	Description
DT	Device Tree
DXE	Driver execution environment

Acronym	Definition
EDL	Emergency download
EFI	Extensible firmware interface
ESP	EFI system partition
FDF	Forms data format
GPIO	General-purpose input/output
IMEM	Internal memory
Initrd	Initial RAM disk
ISP	Image signal processor
KVM	Kernel-Based Virtual Machine
LPASS	Low-power audio subsystem
MMU	Memory management unit
NPU	Neural processing unit
NSP	Neural signal processor
PBL	Primary boot loader
PCAT	Product configuration assistant tool
PCD	Platform configuration database
PMIC	Power management integrated circuit
QDTE	Qualcomm Device Tree editor
QSC	Qualcomm Software Center
Qualcomm TEE	Qualcomm® Trusted Execution Environment (TEE)
RoT	Root of trust
UART	Universal asynchronous receiver/transmitter
UEFI	Unified Extensible Firmware interface
UFS	Universal flash storage
XBL	eXtensible Boot Loader, referred as secondary boot loader (SBL) and SBL1 occasionally in this guide
XBL_CFG	XBL configuration
XBOOTLDR	Extended Boot Loader partition
xPU	eXternal protection unit

LEGAL INFORMATION

Your access to and use of this material, along with any documents, software, specifications, reference board files, drawings, diagnostics and other information contained herein (collectively this “Material”), is subject to your (including the corporation or other legal entity you represent, collectively “You” or “Your”) acceptance of the terms and conditions (“Terms of Use”) set forth below. If You do not agree to these Terms of Use, you may not use this Material and shall immediately destroy any copy thereof.

1) Legal Notice.

This Material is being made available to You solely for Your internal use with those products and service offerings of Qualcomm Technologies, Inc. (“Qualcomm Technologies”), its affiliates and/or licensors described in this Material, and shall not be used for any other purposes. If this Material is marked as “Qualcomm Internal Use Only”, no license is granted to You herein, and You must immediately (a) destroy or return this Material to Qualcomm Technologies, and (b) report Your receipt of this Material to qualcomm.support@qti.qualcomm.com. This Material may not be altered, edited, or modified in any way without Qualcomm Technologies’ prior written approval, nor may it be used for any machine learning or artificial intelligence development purpose which results, whether directly or indirectly, in the creation or development of an automated device, program, tool, algorithm, process, methodology, product and/or other output. Unauthorized use or disclosure of this Material or the information contained herein is strictly prohibited, and You agree to indemnify Qualcomm Technologies, its affiliates and licensors for any damages or losses suffered by Qualcomm Technologies, its affiliates and/or licensors for any such unauthorized uses or disclosures of this Material, in whole or part.

Qualcomm Technologies, its affiliates and/or licensors retain all rights and ownership in and to this Material. No license to any trademark, patent, copyright, mask work protection right or any other intellectual property right is either granted or implied by this Material or any information disclosed herein, including, but not limited to, any license to make, use, import or sell any product, service or technology offering embodying any of the information in this Material.

THIS MATERIAL IS BEING PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESSED, IMPLIED, STATUTORY OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUALCOMM TECHNOLOGIES, ITS AFFILIATES AND/OR LICENSORS SPECIFICALLY DISCLAIM ALL WARRANTIES OF TITLE, MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, COMPLETENESS OR ACCURACY, AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MOREOVER, NEITHER QUALCOMM TECHNOLOGIES, NOR ANY OF ITS AFFILIATES AND/OR LICENSORS, SHALL BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY EXPENSES, LOSSES, USE, OR ACTIONS HOWSOEVER INCURRED OR UNDERTAKEN BY YOU IN RELIANCE ON THIS MATERIAL.

Certain product kits, tools and other items referenced in this Material may require You to accept additional terms and conditions before accessing or using those items.

Technical data specified in this Material may be subject to U.S. and other applicable export control laws. Transmission contrary to U.S. and any other applicable law is strictly prohibited.

Nothing in this Material is an offer to sell any of the components or devices referenced herein.

This Material is subject to change without further notification.

In the event of a conflict between these Terms of Use and the *Website Terms of Use* on www.qualcomm.com, the *Qualcomm Privacy Policy* referenced on www.qualcomm.com, or other legal statements or notices found on prior pages of the Material, these Terms of Use will control. In the event of a conflict between these Terms of Use and any other agreement (written or click-through, including, without limitation any non-disclosure agreement) executed by You and Qualcomm Technologies or a Qualcomm Technologies affiliate and/or licensor with respect to Your access to and use of this Material, the other agreement will control.

These Terms of Use shall be governed by and construed and enforced in accordance with the laws of the State of California, excluding the U.N. Convention on International Sale of Goods, without regard to conflict of laws principles. Any dispute, claim or controversy arising out of or relating to these Terms of Use, or the breach or validity hereof, shall be adjudicated only by a court of competent jurisdiction in the county of San Diego, State of California, and You hereby consent to the personal jurisdiction of such courts for that purpose.

2) Trademark and Product Attribution Statements.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the U.S. and/or elsewhere. The Bluetooth® word mark is a registered trademark owned by Bluetooth SIG, Inc. Other product and brand names referenced in this Material may be trademarks or registered trademarks of their respective owners.

Snapdragon and Qualcomm branded products referenced in this Material are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated.