SG55xD&SG56xD Ubuntu Secure Boot Application Note

Smart Module Series

Version: 1.1

Date: 2024-09-25

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2023-08-31	Morpheus MO	Creation of the document
1.0	2024-04-26	Morpheus MO	First official release
1.1	2024-09-25	Crown YU	1. Added the applicable modules SG555D and SG565D 2. Added the applicable module table (Chapter 1.1)

Contents

Table Index

Figure Index

1 Introduction

This document outlines how to enable Secure Boot on Quectel SG55xD and SG56xD modules.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Family	Module
SG55xD	SG550D
	SG555D
SG56xD	SG560D
	SG565D

NOTE

This document applies to SG55xD and SG56xD modules running Ubuntu operating system.

2 Secure Boot Overview

2.1. Secure Boot Overview

Secure Boot refers to the secure boot-up sequence established on a trusted platform, providing a secure and trusted execution environment for user application. The Secure Boot system adds signature verification to each stage of the module's downloading and booting process to prevent any unauthorized or maliciously modified software from running on the module. Secure Boot verifies and loads the images with cryptographic authentication algorithm, and then runs the verified images.

Secure Boot is intended to:

- Prohibit the burning of unauthorized firmware
- Prohibit the running of unauthorized firmware
- Prohibit the illegal tracing and debugging of code
- Allow secure upgrade

2.2. Firmware Boot-up Sequence

Booting up the module comprises multiple stages. Each firmware in each stage performs the specific function and verifies each image included in each firmware of the next stage. PBL is a firmware included inside the chip, which cannot be modified, and therefore it is considered as the most trusted entity in the booting process. Before the next firmware in the boot-up sequence is executed, the booting process authenticates the firmware to ensure that it contains the authorized software. For example, after PBL is authenticated successfully, the authentication control is passed to SBL. Since SBL is now trusted, the images authenticated by SBL can also be trusted. Based on this authentication sequence, Secure Boot continues the firmware authentication and boots up the module.

NOTE

Secure Boot must be started during the factory phase. It is prohibited to start by upgrading to a version that supports the Secure Boot, otherwise it may cause some features to be abnormal and cannot be restored. Before enabling Secure Boot, module configuration operations such as SIMLOCK configuration should not be performed on the module.

3 Enable Secure Boot

3.1. Set up Signing Environment

3.1.1. Install/Download Tools

1. Install Python 2.7 and export the relative path to environment variable.
2. Install the latest OpenSSL tool (optional).
3. Contact Quectel Technical Support to obtain the Unpacking Tool corresponding to the module. The toolkit package is usually named *SGxx_Ubuntu_xxx_Unpacking_Tool_xxxxxx.zip*. After unzipping the package, remove the read-only permission of its root directory and subdirectory and change the permission of its directory to readable and writable.
4. Contact Quectel Technical Support to obtain *sec_boot_tools_SGxx.zip* corresponding to the module. The toolkit includes configuration files used to enable Secure Boot. The toolkit cannot be used independently. After unzipping the package, please copy all files inside it to the Unpacking Tool and overwrite the files with the same name.
5. Make sure that the QFIL tool is installed.

NOTE

Qualcomm license is needed when QFIL tool is used. Contact Quectel Technical Support (support@quectel.com) for assistance in installing and using QFIL tool.

3.1.2. Download Prebuilt Firmware Package

Contact Quectel Technical Support to obtain the *prebuilt* firmware package corresponding to the module. The firmware package is named *SGxxRxxAxx_BPxx_Ubuntuxxx_prebuilt.zip* and contains the binary images of the firmware at the module's BP side, such as adsp, modem, tz, rpm, boot. After unzipping the *prebuilt* firmware package, please copy all images inside it to the Unpacking Tool's directory with the same name.

NOTE

If the images at BP side are customized, replace the images in the original *prebuilt* firmware package with the custom ones. The specific image path can be found in *contents.xml* of the Unpacking Tool.

3.1.3. Modify Partition Table

By default, QFIL tool does not automatically update the firmware at BP side, such as *non-hlos*, *tz*, *rpm*, *xbl*, when generating the firmware package. However, to enable Secure Boot, it is needed to sign the firmware images and download the signed images to the module. Therefore, you should modify the partition table to enable the update and downloading of the firmware at BP side.

The partition table path is available in *contents.xml* of the Unpacking Tool. The module's partition table is *common/config/ufs/partition_ext.xml*. The example of modifying the partition table is as follows:

```
33 FILTERED LINES
    <partition label="xbl_a" size_in_kb="3604" type="DEA0BA2C-CBDD-4805-B4F9-F428251C3E98" bootable="false" readonly="true" filename="xbl.elf"/>
    <partition label="xbl_config_a" size_in_kb="232" type="5A325AE4-4276-B66D-0ADD-3494DF27706A" bootable="false" readonly="false" filename="xbl_config.elf"/>
5 FILTERED LINES
    <partition label="xbl_b" size_in_kb="3604" type="DEA0BA2C-CBDD-4805-B4F9-F428251C3E98" bootable="false" readonly="true" filename="xbl.elf"/>
    <partition label="xbl_config_b" size_in_kb="232" type="5A325AE4-4276-B66D-0ADD-3494DF27706A" bootable="false" readonly="false" filename="xbl_config.elf"/>
21 FILTERED LINES
    <partition label="aop_a" size_in_kb="512" type="D69E90A5-4CAB-0071-F6DF-AB977F141A7F" bootable="false" readonly="true" filename="aop.mbn"/>
    <partition label="tz_a" size_in_kb="4096" type="A053AA7F-40B8-4B1C-BA08-2F68AC71A4F4" bootable="false" readonly="true" filename="tz.mbn"/>
    <partition label="hyp_a" size_in_kb="8192" type="E1A6A689-0C8D-4CC6-B4E8-55A4320FBD8A" bootable="false" readonly="false" filename="hypvm.mbn"/>
    <partition label="modem_a" size_in_kb="225280" type="EBD0A0A2-B9E5-4433-87C0-68B6B72699C7" bootable="false" readonly="true" filename="NON-HLOS.bin"/>
    <partition label="bluetooth_a" size_in_kb="4096" type="6cb747f1-c2ef-4092-add0-ca39f79c7af4" bootable="false" readonly="true" filename="BTFM.bin"/>
    <partition label="dsp_a" size_in_kb="65536" type="7EFE5010-2A1A-4A1A-B8BC-990257813512" bootable="false" readonly="true" filename="dspso.bin"/>
    <partition label="keymaster_a" size_in_kb="512" type="A11D2A7C-D82A-4C2F-8A01-1805240E6626" bootable="false" readonly="true" filename="km41.mbn"/>
    <partition label="devcfg_a" size_in_kb="128" type="F65D4B16-343D-4E25-AAFC-BE99B6556A6D" bootable="false" readonly="false" filename="devcfg.mbn"/>
    <partition label="qupfw_a" size_in_kb="80" type="21d1219f-2ed1-4ab4-930a-41a16ae75f7f" bootable="false" readonly="false" filename="qupv3fw.elf"/>
    <partition label="uefisecapp_a" size_in_kb="2048" type="BE8A7E08-1B7A-4CAE-993A-D5B7FB55B3C2" bootable="false" readonly="false" filename="uefi_sec.mbn"/>
        <partition label="imagefv_a" size_in_kb="2048" type="17911177-C9E6-4372-933C-804B678E666F" bootable="false" readonly="false" system="true" filename="imagefv.elf"/>
        <partition label="shrm_a" size_in_kb="128" type="CB74CA22-2F0D-4B82-A1D6-C4213F348D73" bootable="false" readonly="true" filename="shrm.elf"/>
        <partition label="multiimgoem_a" size_in_kb="32" type="E126A436-757E-42D0-8D19-0F362F7A62B8" bootable="false" readonly="true" filename="multi_image.mbn"/>
        <partition label="cpucp_a" size_in_kb="1024" type="1E8615BD-6D8C-41AD-B3EA-50E8BF40E43F" bootable="false" readonly="true" filename="cpucp.elf"/>
        <partition label="featenabler_a" size_in_kb="128" type="741813D2-8C87-4465-8C69-032C771CCCE7" bootable="false" readonly="false" filename="featenabler.mbn"/>
5 FILTERED LINES
        <partition label="aop_b" size_in_kb="512" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="false" filename="aop.mbn"/>
        <partition label="tz_b" size_in_kb="4096" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="false" filename="tz.mbn"/>
    <partition label="hyp_b" size_in_kb="8192" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="false" filename="hypvm.mbn"/>
    <partition label="modem_b" size_in_kb="225280" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="true" filename="NON-HLOS.bin"/>
    <partition label="bluetooth_b" size_in_kb="4096" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="true" filename="BTFM.bin"/>
    <partition label="dsp_b" size_in_kb="65536" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="true" filename="dspso.bin"/>
    <partition label="keymaster_b" size_in_kb="512" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="true" filename="km41.mbn"/>
    <partition label="devcfg_b" size_in_kb="128" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="false" filename="devcfg.mbn"/>
    <partition label="qupfw_b" size_in_kb="80" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="false" filename="qupv3fw.elf"/>
    <partition label="uefisecapp_b" size_in_kb="2048" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="false" filename="uefi_sec.mbn"/>
        <partition label="imagefv_b" size_in_kb="2048" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="false" system="true" filename="imagefv.elf"/>
        <partition label="shrm_b" size_in_kb="128" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="true" filename="shrm.elf"/>
        <partition label="multiimgoem_b" size_in_kb="32" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="true" filename="multi_image.mbn"/>
        <partition label="cpucp_b" size_in_kb="1024" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="true" filename="cpucp.elf"/>
        <partition label="featenabler_b" size_in_kb="128" type="77036CD4-03D5-42BB-8ED1-37E5A88BAA34" bootable="false" readonly="false" filename="featenabler.mbn"/>
1 FILTERED LINES
    <partition label="devinfo" size_in_kb="4" type="65ADDCF4-0C5C-4D9A-AC2D-D90B5CBFCD03" bootable="false" readonly="true" filename="devinfo"/>
15 FILTERED LINES
        <partition label="rtice" size_in_kb="512" type="e396d1ea-0eac-4241-bb82-7150645cfc45" bootable="false" readonly="true" filename="rtice.mbn"/>
        <partition label="apdp" size_in_kb="256" type="E6E98DA2-E22A-4D12-AB33-169E7DEAA507" bootable="false" readonly=" false" filename="apdp.mbn"/>
    <partition label="logfs" size_in_kb="8192" type="BC0330EB-3410-4951-A617-03898DBE3372" bootable="false" readonly="false" filename="logfs_ufs_8mb.bin"/>
        <partition label="storsec" size_in_kb="128" type="02DB45FE-AD1B-4CB6-AECC-0042C637DEFA" bootable="false" readonly="true" filename="storsec.mbn"/>
```

Figure 1: Modify Partition Table partition_ext.xml

NOTE

1. Refer to the partition table, modified by Quectel, in the latest update package *SGxxRxxAxx_BPxx_Ubuntuxxx_update.zip* corresponding to the module for modifying thepartition table.
2. *persist* partition cannot be added.

3.2. Generate Keys

Open Windows Terminal under the root directory of the Unpacking Tool and execute the following command to generate keys:

```
build_SG560D_qcm6490_secboot.bat newkey
```

The generated keys are in *common/sectools/resources/data_prov_assets/Signing/Local/oem_certs* directory of the host.

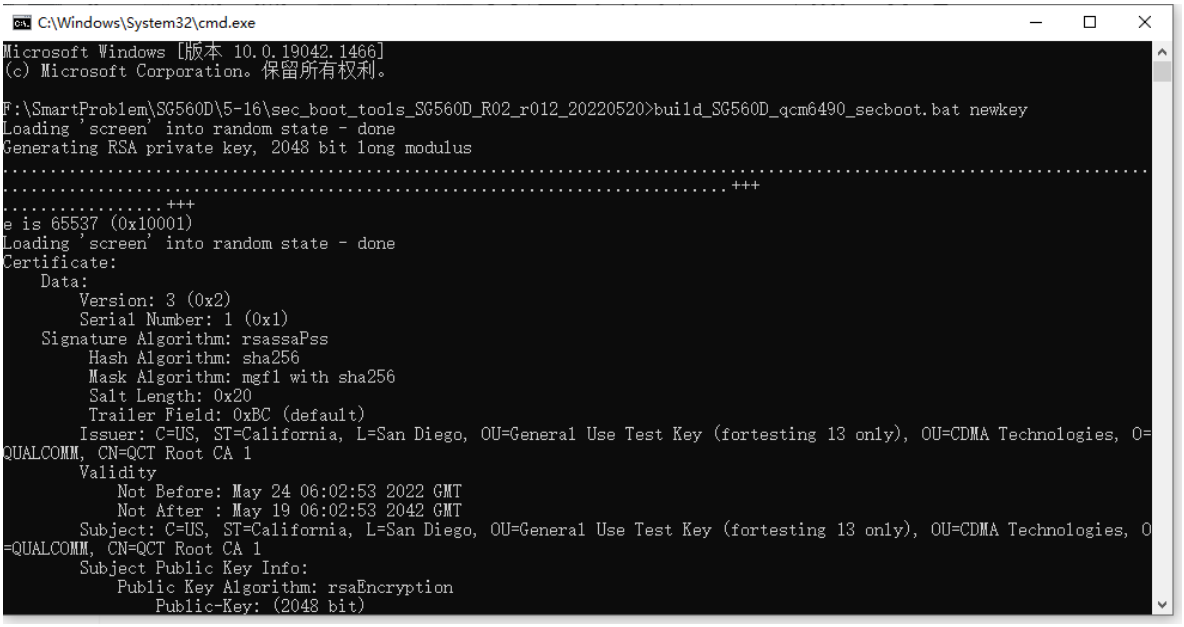

Figure 2: Generate Keys

NOTE

1. The keys only need to be generated once. Please keep them protected to prevent loss or leakage.
   - If you want to customize the keys, you can directly replace the keys in *common/sectools/resources/data_prov_assets/Signing/Local/oem_certs* of the host.
   - If you need to generate secboot firmware again, you only need to copy the keys to *common/sectools/resources/data_prov_assets/Signing/Local/oem_certs* of the host.
2. Once *sec.elf* file containing the keys is burned into the module to enable Secure Boot, only the images signed by the keys in *common/tools/sectools/resources/data_prov_assets/Signing/Local/oem_certs* directory can be used.
3. For the module, you need to copy the generated keys in *common/sectools/resources/data_prov_assets/Signing/Local/oem_certs* directory to *vendor/qcom/proprietary/sectools/resources/data_prov_assets/Signing/Local* directory in Ubuntu source code at AP side to replace the files with the same name.

3.3. Sign Images

3.3.1. Sign zap Image

The zap image is located at Ubuntu side, and will be verified during the Secure Boot boot-up process. Therefore, it should be signed and packed into Ubuntu images.

1. Copy *src/le-bins-1/adreno/lib/firmware/a660_zap.elf* image file of Ubuntu source code to *LINUX/firmware/* of the Unpacking Tool.

2. Execute the following command under the root directory of the Unpacking Tool to sign the zap image:

```
build_SG560D_qcm6490_secboot.bat zap
```

3. The signed zap image is in *common/sectools/secimage_output/kodiak/gfx_microcode/* directory of the Unpacking Tool. Copy all images under this directory to Ubuntu source code directory (*src/le-bins-1/adreno/lib/firmware/*) and replace the files with the same name.

NOTE

The zap image needs to be signed for the module, otherwise the screen will not light up after power-on.

3.3.2. Sign ipa_fws Image

The ipa_fws image will be verified during the Secure Boot boot-up process. Therefore, it should be signed and packed into Ubuntu images.

1. Copy the image file *poky/meta-qti-data-prop/recipes/ipa-fws/files/ipa_fws/ipa_fws.elf* of Ubuntu source code to *LINUX/firmware/* of the Unpacking Tool.

2. Execute the following command under the root directory of the Unpacking Tool to sign the ipa_fws image:

```
build_SG560D_qcm6490_secboot.bat    ipa_fws
```

3. The signed ipa_fws image is in *common/sectools/secimage_output/kodiak/ipa_fw/* directory of the Unpacking Tool. Copy all images under this directory to Ubuntu source code directory (*poky/meta-qti-data-prop/recipes/ipa-fws/files/ipa_fws/*) and replace the files with the same name.

4. Re-compile Ubuntu source code and ensure that the image file generated in *build-qti-distro-rb-debug/tmp-glibc* directory is the latest image file generated by the compilation (you can directly delete *build-qti-distro-rb-debug/tmp-glibc* to re-compile Ubuntu).

NOTE

The ipa_fws image needs to be signed for the module, otherwise the module may crash and reboot after power-on.

3.3.3. Sign adsp Image

The adsp image will be verified during the Secure Boot boot-up process. Therefore, it should be signed and packed into Ubuntu images.

1. Execute the following command under the root directory of the Unpacking Tool to sign the adsp image:

```
build_SG560D_qcm6490_secboot.bat adsp
```

2. The signed adsp image is in *common/sectools/secimage_output/kodiak/adsp/* directory of the Unpacking Tool. Copy all images under this directory to Ubuntu source code directory (*src/vendor/qcom/proprietary/adsp-firmware/prebuilt_HY11/firmware/*) and replace the files with the same name.

3. Re-compile Ubuntu source code to make sure the images in *build-qti-distro-rb-debug/tmp-glibc* directory are the latest images (you can directly delete *build-qti-distro-rb-debug/tmp-glibc* to re-compile Ubuntu).

NOTE

1. The adsp image needs to be signed for the module, otherwise the audio and video related functions are abnormal after power-on.
2. Please make sure the used *prebuilt* package is the latest.
3. If adsp image is modified, you have to replace the original image with the modified image. The original image path can be found in *contents.xml* of the Unpacking Tool.

3.3.4. Sign Required Images

1. Copy the Ubuntu images (**.ext4*, **.img*, **.bin* and **.elf*) with updated keys, zap image and adsp image in *build-qti-distro-Linux-fullstack-debug/tmp-glibc/deploy/images/qcs6490-odk/* of Ubuntu source code to *LINUX/android/out/target/product/lahaina/* of the Unpacking Tool.

2. Execute the following command under the root directory of the Unpacking Tool to generate required images. The signed download proxy image *prog_firehose_SG560D_ddr_quectel.elf* is generated in *common/sectools/secimage_output/kodiak/prog_firehose_ddr/* of the Unpacking Tool.

```
build_SG560D_qcm6490_secboot.bat
```

NOTE

After the required images are signed, you can generate a firmware package without *sec.elf* file first by using QFIL tool and burn it into the module to check whether the module can boot normally after the aforementioned steps. If the module boots successfully, you can continue the next steps.

3.4. Generate sec.elf File

sec.elf file contains keys (for the generation of keys, see ***Chapter 3.2***). You can burn the file into the module to enable Secure Boot.

Execute the following command in Windows Terminal to generate *sec.elf* file containing keys information. The generated file is in *common/sectools/fuseblower_output/v2/* of the host.

```
build_SG560D_qcm6490_secboot.bat sec.elf
```

3.5. Enable Secure Boot

The module supports enabling Secure Boot by burning *sec.elf* file only or burning the secboot firmware package containing *sec.elf* file.

NOTE

1. It is recommended to enable Secure Boot by burning *sec.elf* file only.
2. Before you enable Secure Boot by burning the secboot firmware package containing *sec.elf* file, it is recommended to burn a secboot firmware package without *sec.elf* file into the module whose Secure Boot is not enabled first, which can test whether the related functions work normally.

3.5.1. Download sec.elf File

After *sec.elf* file is generated, you can execute the following command to burn *sec.elf* file into the module to enable Secure Boot:

```
fastboot flash secdata sec.elf
```

3.5.2. Burn secboot Firmware Package

Step 1: After the images are signed, collect the signed images to generate the secboot firmware package to be burned into the module. See below for details.

Open QFIL tool, and select "**ufs**" in "**Storage Type**".

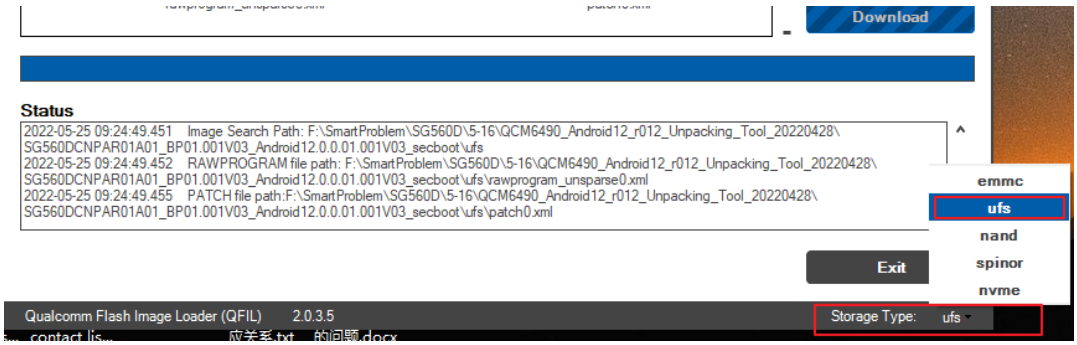

Figure 3: Select Storage Type

Select "**Tools**" → "**Flat Meta Build**". Click "**Browse…**" in "**Content XML**" to select *contents.xml* of the Unpacking Tool. Click "**Browse…**" in "**Flat Build Path**" to select the storage path of the secboot firmware package to be generated. Select "**ufs**" in "**Meta Build Available Storage Type**". Finally, click "**OK**" to start generating the secboot firmware package.

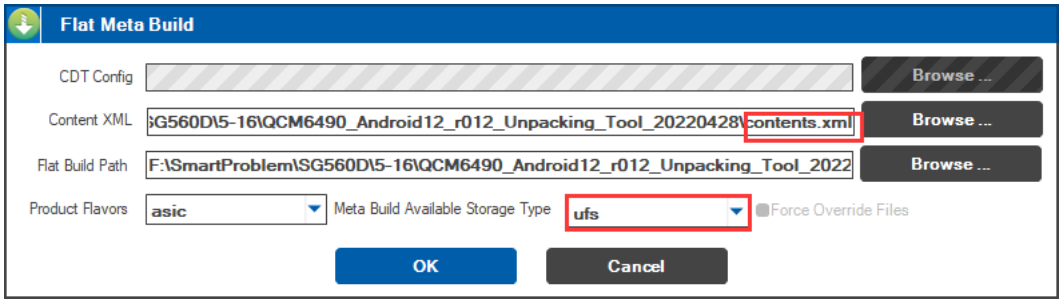

Figure 4: Generate secboot Firmware Package

Step 2： Modify *common/config/ufs/partition_ext.xml* to add *sec.elf* file:

```
<partition label="secdata" size_in_kb="25" type="76cfc7ef-039d-4e2c-b81e-4dd8c2cb2a93" bootable="false" readonly="true" filename="sec.elf"/>
```

Step 3： After the secboot firmware package is generated, copy *prog_firehose_SG560D_ddr_quectel.elf* from *common/sectools/secimage_output/kodiak/prog_firehose_ddr/* and *sec.elf* file from *common/sectools/fuseblower_output/v2/* of the Unpacking Tool to the root directory of the secboot firmware package.

NOTE

To generate a secboot firmware package without *sec.elf* file, ignore ***Step 2*** and ***Step 3***.

Step 4： After the images are replaced successfully, for the module whose Secure Boot is not enabled yet, you can burn the secboot firmware package into the module by selecting the igned download proxy image *prog_firehose_SG560D_ddr_quectel.elf* or the unsigned download proxy image *prog_firehose_ddr.elf* through QFIL tool. After the secboot firmware package is downloaded successfully, reboot the module to enable Secure Boot.

Take selecting the signed download proxy image *prog_firehose_SG560D_ddr_quectel.elf* to burn the firmware package as an example, the detailed configuration on QFIL tool is as follows:

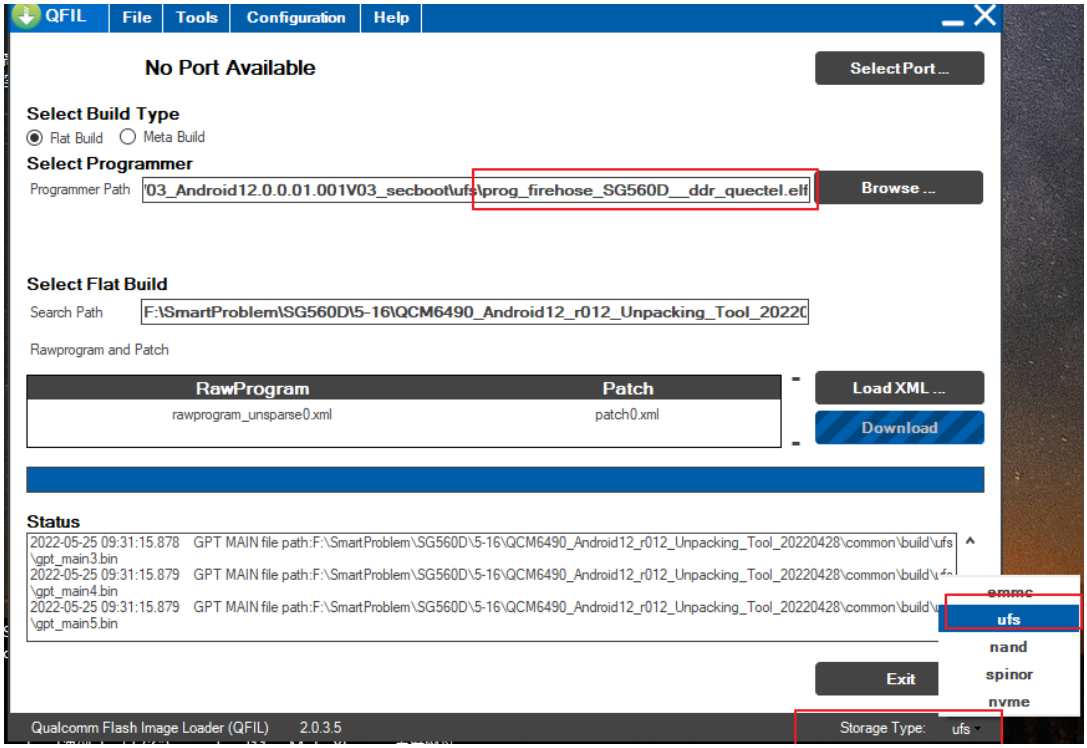

Figure 5: Burn secboot Firmware Package

NOTE

1. If *devinfo* cannot be found during the downloading process, you can directly copy it from *SGxxRxxAxx_BPxx_Ubuntuxxx_update.zip* to the image directory.
2. For the module whose Secure Boot has been enabled, verification will be performed when the signed firmware is re-burned. Therefore, you must use the signed download proxy *prog_firehose_SG560D_ddr_quectel.elf* to burn the firmware package.

4 Appendix References

Table 2: Terms and Abbreviations

Abbreviation	Description
AP	Application Processor
BP	Baseband Processor
PBL	Primary Boot Loader
QFIL	Qualcomm Flash Image Loader
SBL	Secondary Boot Loader