SIERRA WIRELESS

WP Series Customer Secure Boot

41112164-4

Author:	Sierra Wireless	Date:	Oct 17, 2019

APN Content Level	BASIC		INTERMEDIATE	✓	ADVANCED		Confidentiality	Public	✓	Private	

Hardware Compatibility	Product Line	AirPrime	Series	WP76xx	
				WP77xx	

Software Compatibility	ALL	Document Type	Application Note	✓	Technical Note	

Version

This document may be updated over its lifetime. To ensure you design with the correct version, please check source.sierrawireless.com for the latest version. (Search for “41112164” (this document number).)

1 Introduction

This document is provided to Sierra Wireless distributors and clients to aid more rapid development of embedded applications using the Sierra Wireless portfolio of cellular solutions. To request a new application/technical note, contact your regional Sierra Wireless Product Marketing Manager.

2 Glossary

Term/Initials	Definition
Secure Boot	Mechanism to ensure only legitimate images are loaded and run by the hardware at boot time
PKI	Public Key Infrastructure
Keystore	Write-once image stored on module, used to authenticate Linux images

3 Application Note Description

This document describes Customer Secure Boot for Linux images, an extension of the Secure Boot functionality that Sierra Wireless WP series modules automatically provide for firmware images, which ensures only authenticated images are loaded and run on WP modules. Considerations for choosing to use Sierra Wireless-authenticated or customer-authenticated images, and the procedures used to implement Customer Secure Boot are also presented.

4 Secure Boot / Customer Secure Boot Overview

By default, Secure Boot protects the WP module from loading or running untrusted Modem firmware images, using digital signature technology to make sure only images signed by a trusted source can run on the module.

Expanding on the default Secure Boot functionality, Customer Secure Boot can be enabled to authenticate (using Android Verified Boot Signature) the Linux distribution’s bootloader and kernel images and, by association, the Linux root filesystem image. For details about the Android Verified Boot Signature format, see https://source.android.com/security/verifiedboot/verified-boot#signature_format.

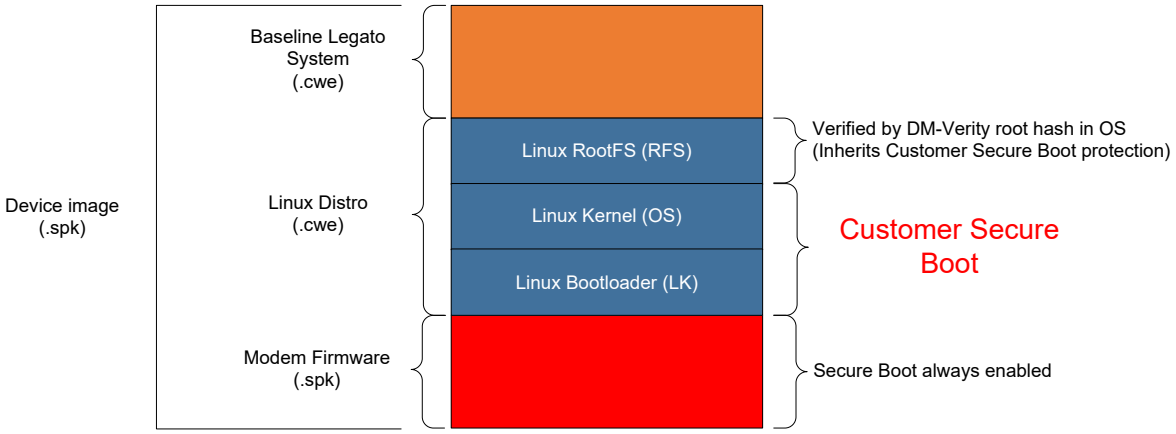

Figure 1 Secure Boot application to image components

As shown in Figure 1, a complete device image includes a Modem firmware that is automatically authenticated by Secure Boot, and a Linux distribution that can be authenticated by Customer Secure Boot.

Note: Sierra Wireless strongly recommends Customer Secure Boot be enabled to protect the WP module from potential issues due to using unapproved or malicious images.

5 Authentication Options

The customer must determine the authentication key type that Customer Secure Boot authenticates images against, as described in Table 1.

Table 1 Customer Secure Boot Authentication Method

Customer Secure Boot state	Authentication key type	Description
Enabled (Authenticated)	Customer key	Recommended, least risk to customer Benefits: • Only customer-approved images can be loaded on the module. • Customers can customize images. Requirement: • Customer must manage key and sign Linux images
Enabled (Authenticated)	Sierra Wireless key	Recommended, low risk to customer Benefits: • No customer image signing required • Only Sierra Wireless-provided images can be loaded on the module. Drawbacks: • Customer cannot customize Linux images.
Disabled (Unauthenticated)	None	*Note: Factory default state for Generic SKUs.* Not recommended, high risk to customer Drawbacks: • Any Linux image can be loaded on the module. • No verification that the image is approved by either Sierra Wireless or customer. • Malicious image could be developed by third party and loaded.

Requirements for implementing Customer Secure Boot depend on the key type selected for authentication:

- Customer key:
  - Customer must manage keys/certificates for image signing. Sierra Wireless strongly recommends using a secure signing server. See 8 Signing Server Configuration for signing server setup and key/certificate generation details.
  - Customer must sign Linux images before installing on the module.
  - Customer must enable Customer Secure Boot and load the customer key into each module.
- Sierra Wireless key:
  - Minimal setup — Enable Customer Secure Boot and load the Sierra Wireless-provided key and image into each module.

Note: The Sierra Wireless or Customer key is contained in a 'keystore' CWE image, which is loaded onto the module using FDT or fwupdate. The keystore is then used (when Customer Secure Boot is enabled) to authenticate any Linux image being loaded or run on the module.

- No key – No image verification, Customer Secure Boot not enabled.

6 Implementing Customer Secure Boot Implementation

6.1 Implementation Process

The following procedure describes a typical process for implementing Customer Secure Boot:

1. Receive factory-configured WP modules with a device image comprised of Modem firmware (with Secure Boot enabled), a Linux distribution (Customer Secure Boot not enabled), and a base Legato system.
2. Decide whether modules will have Customer Secure Boot enabled for Linux images, and which authentication method to use (see Table 1 Customer Secure Boot Authentication for details):
   - Customer key/certificate authentication – If a signing server has not yet been set up, and keys/certificates and a keystore CWE have not been created, follow the instructions in 8 Signing Server Configuration.
   - Sierra Wireless key authentication – Get the module series-specific Sierra Wireless-provided keystore CWE file from https://source.sierrawireless.com. (Search for "Sierra Wireless Public Keys", or select Find more Resources from an appropriate WP Series device page.)
     (Make sure to use the correct keystore CWE file – each module series (e.g. WP76, WP77, etc.) uses a unique keystore.)
   - No authentication — Secure Boot will be disabled for Linux images. No special setup is required. End of process.
     (This method is not recommended due to high risk to customer.)
3. Before Customer Secure Boot can be enabled on the module, the Linux image on the module (which is signed with the Sierra Wireless key), must be re-signed or replaced:
   - If a customer key is used, a customer-built and signed image can be used, or a signed Sierra Wireless-provided image can be re-signed with the customer key:
     - Customer image: See 6.2 Sign an Image to build and sign the image.
     - Sierra Wireless image: Contact Sierra Wireless for instructions.

       Note: A forthcoming revision of this document will include detailed instructions.
   - If the Sierra Wireless key is used, get the appropriate image from https://source.sierrawireless.com (all images on the Source are signed).

4. Load the signed Linux image on the WP module, using FDT.
   (If a properly signed image is not loaded and Customer Secure Boot is enabled, the module will not authenticate the image and will go to download mode.)

   Note: FDT is strongly recommended for local downloads. If FDT download is not possible for some reason, the fwupdate tool can be used.

5. Enable Customer Secure Boot. See 7 Enable Customer Secure Boot.

   The module is now Customer Secure Boot-protected.

6.2 Sign an Image

To sign a customer image:

1. On a build machine, build the unsigned image.
2. Copy the unsigned image components (LK, OS) from the build machine to the secure signing server:
   - OS component:
     From: Build system output (tmp/deploy/images/swi-mdm9x28*/boot-yocto-md-mdm9x28.4k.unsigned.img)
     To: Signing server path: User-defined path
   - LK component:
     From: Build system output (tmp/deploy/images/swi-mdm9x28*/ appsboot.mbn.unsigned)
     To: Signing server path: User-defined path
3. On the signing server, sign both components using the script that contains the android_signature_add() function):
   For example:

```
android_signature_add /boot boot-yocto-mdm9x28.4k.unsigned.img boot-yocto-
             mdm9x28.48.img
android_signature_add /aboot appsboot.mbn.unsigned appsboot.mbn
```

4. Package the signed images into CWE images. For details, refer to the following script in the build system output – meta-swi/meta-swi-mdm9x28/recipes-core/images/mdm9x28-image-cwe.inc.

7 Enable Customer Secure Boot

To enable Customer Secure Boot, a keystore CWE image (containing either the Sierra Wireless key or customer-generated key) must be stored in the module and AT commands must be run. (The keystore CWE was obtained/generated earlier in 6 Implementing Customer Secure Boot Implementation, step 2.)

Warning: *A properly-signed Linux image (signed with the private key matching the keystore) must be loaded on the module before enabling Customer Secure Boot. If not, Customer Secure Boot will not authenticate the incorrectly-signed image and the module will go to download mode.*

For each module:

1. After a properly-signed Linux image has been loaded on the module in 6.1 Implementation Process, step 4, enable Customer Secure Boot:

```
AT!ENTERCND=<password>
AT!OEMAUTH=1
```

2. Download the keystore CWE image to the module using FDT.
   The module automatically reboots after downloading the image and Customer Secure Boot will authenticate the Linux imdage.

   Warning: *The module uses 'write-once' storage for the keystore CWE image – once written, the keystore cannot be replaced or removed. If a keystore CWE has already been stored on the module, error code 0x97 is returned to the host tool (e.g. FDT or fwupdate).*

3. When the module has rebooted, confirm Customer Secure Boot is enabled and the OEM cert hash matches the customer key hash:

```
AT!OEMAUTH?

!OEMAUTH: Enabled
OEM cert hash: 8AD127ABAE8285B582EA36745F220AB8FE397FFB3B068DF19CA22D122C7B3B86

OK
```

8 Signing Server Configuration

Note: It is the customer's responsibility to keep private key(s) secure. Sierra Wireless strongly recommends using a secure signing server (see 8.3 Secure Signing Server Setup) to generate, protect, and use the private key(s)/certificate(s) being generated.

8.1 Secure Signing Server Setup

Customers that will use their own keys to sign images are strongly recommended to use a secure signing server. The customer is responsible for:

- Server setup, security, access and maintenance
- Management and protection of customer keys/certificates

To sign images, the signing server must include the following tools/scripts that are produced as part of the build system output, as described in the procedure following this list:

- Android open-source signing tool – Used to sign Linux images (LK and OS) using the Android Verified Boot Signature format. (For details about this format, see https://source.android.com/security/verifiedboot/verified-boot#signature_format.)
- 'make_key' script – Used to generate the public/private key pair that will be used to sign images with single-level certificates (customer-managed).
- 'swi-make-cert-chain.sh' script – Used to generate the certificate chain and key pair used to sign images with customer-managed PKI (multi-level certificates).
- Signing script

Warning: *The total size of signature and certificates must not exceed 8 KB. Earlier releases had a 4 KB limit.*

To set up the signing server with the required tools, scripts, and keys/certificates:

1. Make sure openssl is installed on the signing server.
2. Build an image on the build system, to generate the build output that includes the required items. These items do not change between builds, so only need to be copied one time to the signing server.
3. On the signing server, designate a directory to be used for signing images, and create an environment variable (ANDROID_SIGNING_DIR) for this path.
   Note: $ANDROID_SIGNING_DIR is used in the signing script that will be created in step 5.

4. Copy the following files from the build system to the signing server:
   - From: Build system output (tmp/sysroots/x86_64-linux/usr/share/android-signing/*)
     To: Signing server ($ANDROID_SIGNING_DIR)
     (This copies the Android open-source signing tool, make_key script, and default (dummy) keys/certificates to the signing server.)
   - From: Build system output (meta-swi/meta-swi-mdm9xxx/recipes-bsp/android-signing/files/swi-make-cert-chain.sh)
     To: Signing server path: $ANDROID_SIGNING_DIR. (Suggested; path choice is user preference.)
5. On the signing server, create a signing script containing the android_signature_add() function shown below. This function uses the boot_signer (signing tool) to sign images with a specified key and (if using customer-managed PKI) certificate.
   (Note: The function can also be copied from the build system at meta-swi/common/classes/android-signing.bbclass.)

```
android_signature_add() {
    local image_type=$1
    local unsigned_image_path=$2
    local signed_image_path=$3

    ${ANDROID_SIGNING_DIR}/verity/boot_signer $image_type \
                ${unsigned_image_path} \
                ${ANDROID_SIGNING_DIR}/security/verity.pk8
                                ${ANDROID_SIGNING_DIR}/security/verity.x509.pem \
                ${signed_image_path}

    # append cert chain if exists
    if [ -e ${ANDROID_SIGNING_DIR}/security/AttestationCA.der ]; then
      cat ${ANDROID_SIGNING_DIR}/security/AttestationCA.der >> ${signed_image_path}
    fi
    if [ -e ${ANDROID_SIGNING_DIR}/security/RootCA.der ]; then
      cat ${ANDROID_SIGNING_DIR}/security/RootCA.der >> ${signed_image_path}
    fi
}
```

   The signing server now has the files required to sign images, and the scripts for generating the keys and (if required) certificate.
6. Determine which image signing method will be used:
   - Single-level certificate — See 8.1.1 Prepare for Single Certificate Signing (Customer-managed).
   - PKI with multi-level certificate chain — See 8.1.2 Prepare for PKI Multi-level Certificate Signing (Third Party or Customer-managed).

The signing server is now configured to sign images.

8.1.1 Prepare for Single Certificate Signing (Customer-managed)

To generate the private/public key pair that will be used for image signing, and the keystore CWE image that will be stored in the modules:

Note: This procedure only needs to be run one time. Once created, the key pair and CWE image can be used repeatedly.

1. On the signing server, create a private/public key pair:
   - If the key pair will be customer-generated:
     a. Run the make_key script to generate a new private/public key pair:

```
make_key mykey '/C=US/ST=California/L=Mountain
            View/O=Android/OU=Android/
            CN=Android/emailAddress=android@android.com
```

     b. When prompted, choose a password to protect the private key file that is being generated.

     The private/public key pair (mykey.pk8, mykey.x509.pem) are created in the current folder.
   - If the key pair is not customer-generated, obtain a key pair from a third-party certificate authority.

Note: Whether or not a password is used, the private key file and secure signing server must be properly protected.

2. Replace the default (dummy) key pair with the new key pair:

```
cp mykey.pk8 $ANDROID_SIGNING_DIR/security/verity.pk8
cp mykey.pk8 $ANDROID_SIGNING_DIR/security/ verity.x509.pem
```

3. Generate the keystore CWE image corresponding to the new key:

```
$ANDROID_SIGNING_DIR/swi-key-cwe.sh verity.x509.pem
```

8.1.2 Prepare for PKI Multi-level Certificate Signing (Third Party or Customer-managed)

To generate the private/public key pair and certificate chain that will be used for image signing, and the keystore CWE image that will be stored in the modules:

Note: This procedure only need to be run one time. Once created, the key pair and CWE image can be used repeatedly.

1. On the signing server, generate a certificate chain and private/public key pair:
   - If PKI services will be provided by a trusted third party, follow the procedures provided by the third party to generate the certificate chain and key pair (Attestation.pk8, Attestation.pem, RootCA.der, AttestationCA.der).
   - If PKI services will be customer-managed, run the following script:

```
$ANDROID_SIGNING_DIR/swi-make-cert-chain.sh
```

     The certificate chain and key pair are created in the current folder.

2. Replace the default key pair with the new key pair:

```
cp Attestation.pk8 $ANDROID_SIGNING_DIR/security/ verity.pk8
cp Attestation.pem $ANDROID_SIGNING_DIR/security/ verity.x509.pem
```

3. Copy the certificate chain:

```
cp RootCA.der $ANDROID_SIGNING_DIR/security
cp AttestationCA.der $ANDROID_SIGNING_DIR/security
```

4. Generate the keystore CWE image corresponding to the new RootCA certificate generated in step 1:

```
$ANDROID_SIGNING_DIR/swi-key-cwe.sh RootCA.pem
```

8.2 Testing Customer Secure Boot

Images built with the Linux build system are automatically signed with a default development private key/public certificate, and a corresponding keystore CWE image is available in the build system. Customers wanting to test their processes for managing keys, signing images, and enabling Secure Boot can use these files (key, certificate, keystore CWE) in their test build environment.

The files are located in the Linux build system:

- Development key — tmp/sysroots/x86_64-linux/usr/share/android-signing/security/verity.pk8
- Development certificate — tmp/sysroots/x86_64-linux/usr/share/android-signing/security/verity.x509.pem
- Keystore CWE — tmp/sysroots/x86_64-linux/usr/share/android-signing/swi-keys.cwe
- Linux images generated by build system:
  - Linux Bootloader:
    (signed) tmp/deployimages/*/appsboot.mbn
    (unsigned) tmp/deployimages/*/appsboot.mbn.unsigned
  - Linux Kernel:
    (signed) tmp/deployimages/*/boot-yocto-*.img
    (unsigned) tmp/deployimages/*/boot-yocto-*.unsigned.img

Note: These files can be used for testing, but any modules that have the default keystore CWE image stored cannot be re-used to test with 'real' key/certificate/CWE since the keystore is a write-once value.

9 Standard AT Commands

AT commands used for this application note are described below. For additional details, refer to AirPrimeWP8548/WP75xx/WP76xx/WP77xx AT Command Reference (Document #4118047, available from https://source.sierrawireless.com.

AT Command Syntax		
AT!ENTERCND="<password>"		
Parameters	Values	Description
<password>	ASCII string	Password to access protected AT commands
AT!OEMAUTH=<option>		
Parameters	Values	Description
<option>	1	Enable downloading of keystore CWE image

10 Software Compatibility Matrix

Firmware	Legato Application Framework	Library/Libraries
From FW Release 7		

11 Support

For direct clients: contact your Sierra Wireless FAE

For distributor clients: contact your distributor FAE

For distributors: contact your Sierra Wireless FAE

12 Document History

Level	Date	History
1	Feb 20, 2018	Creation
2	July 18, 2018	Content reorganization, clarification, and expansion
3	February 01, 2019	Indicated fdt as preferred download tool, minor terminology clarifications
4	October 11, 2019	Added storage size warning for signature+certificate(s)

13 Legal Notice