AN13469

AWS OTA Update on i.MX RT Platform

Rev. 0 — 6 December 2021

Application Note

Contents

1 Introduction

This application note introduces the Over-the-Air (OTA) update progress on i.MX RT1xxx series platform based on the Amazon Web Services (AWS).

The AWS OTA update is implemented via two projects: Secure Bootloader (SBL) and Secure Firmware (SFW).

The SBL is a secure bootloader for OTA project, and it can guarantee the OTA trust chain with NXP SoC secure engine.

SFW is designed based on FreeRTOS, NXP SDK, and other functional modules. SFW can work with SBL to provide a complete secure OTA solution. It supports OTA via SD card, U-Disk, AWS, or Aliyun cloud. This application note introduces the OTA via AWS.

2 AWS OTA overview

The AWS OTA process is as shown in Figure 1. To update a new firmware, perform the following steps:

1. Upload the signed image with new version to S3 bucket.
2. Create an OTA update job. This job notifies RT1170 device that a firmware update is available.
3. The RT1170 device downloads the new image, validates it, and then updates its application code.

Now, the device is updated, the new application code runs until a new update is available, and the device updates the job status to AWS.

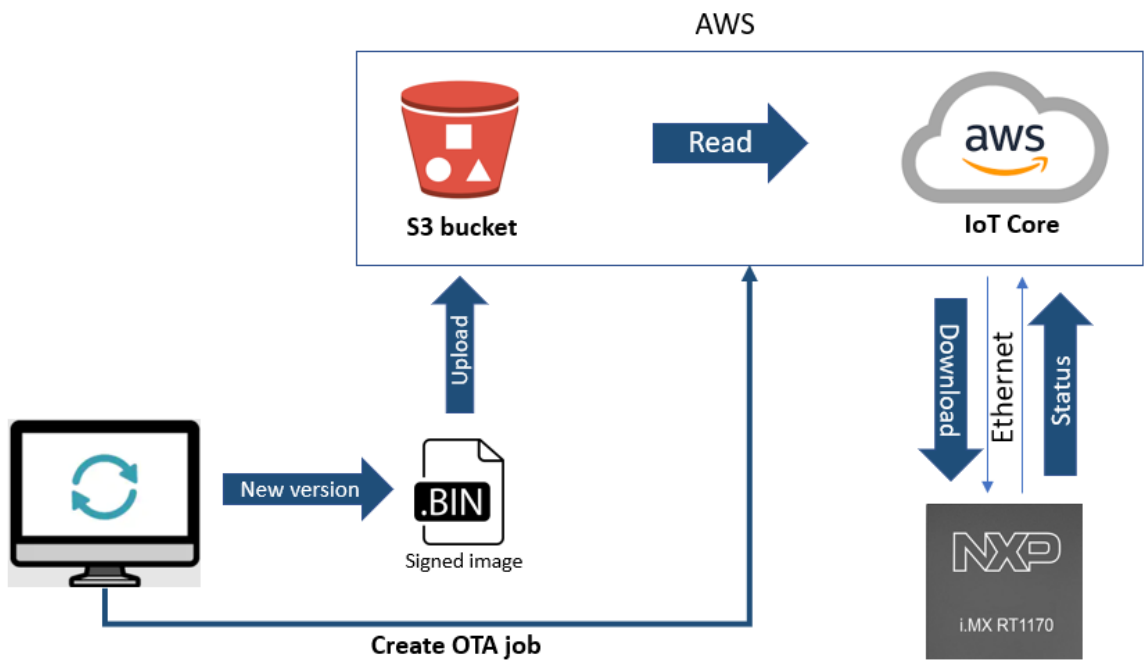

Figure 1. AWS OTA process diagram

3 AWS configuration

To configure AWS services to make an AWS OTA on i.MX RT1xxx series platform, perform the following steps:

1. Create an IAM role with OTA update, S3, IoT policies, and permissions.
2. Use OpenSSL and AWS CLI commands to create a code signing certificate.
3. Create an AWS IoT thing with the code signing certificate.

For detailed configuration process, see **Chapter 7.3.1.1 AWS OTA Prerequisites** in *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG).

4 AWS OTA

SFW can support AWS OTA on i.MX RT1020, RT1050, RT1060, RT1064, and RT1170 platforms. SFW cooperates with SBL to provide a complete secure OTA solution. SFW connects to AWS via Ethernet. This chapter introduces how to use SBL and SFW to implement AWS OTA on RT1170-EVK board.

Download SBL and SFW projects from below links:

- SBL: https://github.com/NXPmicro/sbl
- SFW: https://github.com/NXPmicro/sfw

4.1 Preparing the SBL

To download SBL project to the target board, perform the following steps:

1. Enter the `sbl/target/evkmimxrt1170` path and double click the `env.bat`.
2. In `env.bat`, run the `scons --menuconfig` command, and the SBL configuration menu is generated, as shown in Figure 2.

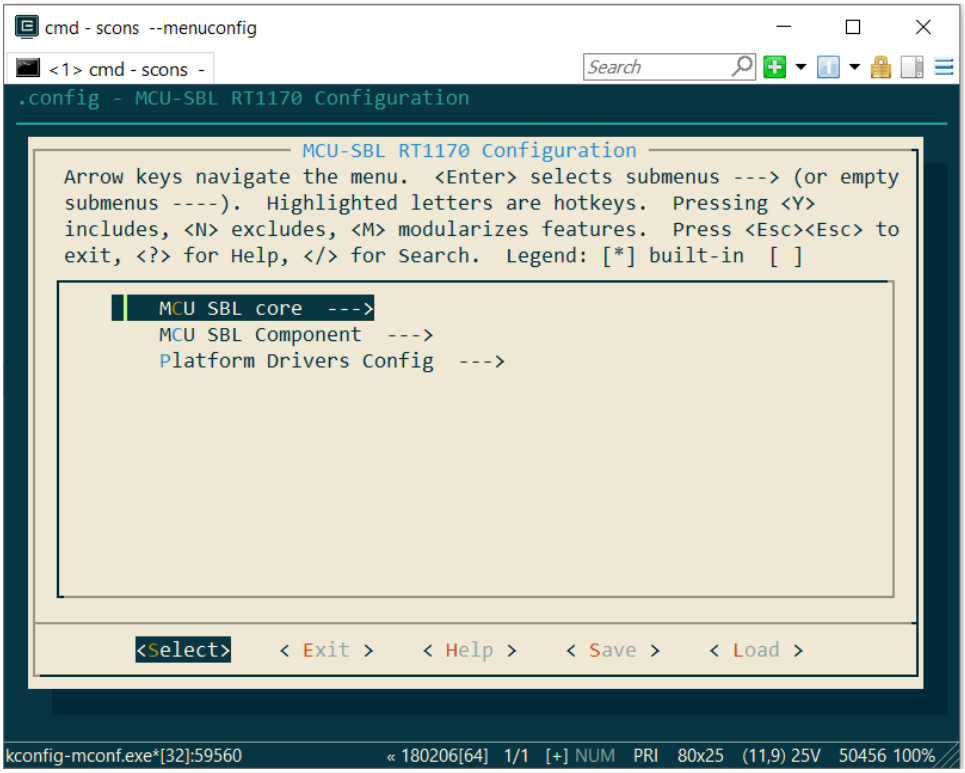

Figure 2. SBL configuration menu

3. In the menu, to disable single image mode or MCU ISP support, disable the **Enable single image function** or **Enable mcu isp support** option. After the configuration completes, save and exit the menu.
4. In the `env.bat` file, run the `scons --ide=iar` command to generate `sbl.eww` IAR project in the `sbl/target/evkmimxrt1170/iar` path.

NOTE

For how to generate a `keil` or `gcc` project, see **Chapter 2 Quick start** in *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG).

5. Compile the SBL project and download it to the RT1170-EVK board.

4.2 Configuring the SFW

To download an SFW project to the target board, configure the SFW project based on AWS requirements as below:

1. Generate the `aws_clientcredential_keys.h` file.
   a. Enter the `sfw/firmware/aws_ota/tool` path.
   b. Using a web browser to open the `CertificateConfigurator.html`.
   c. Browse to the Certificate and Key files previously created in the **Create an AWS IoT Thing** part. For details, see **Chapter 7.3.1.1 AWS OTA Prerequisites** in *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG). Click **Generate and save aws_clientcredential_keys.h**.

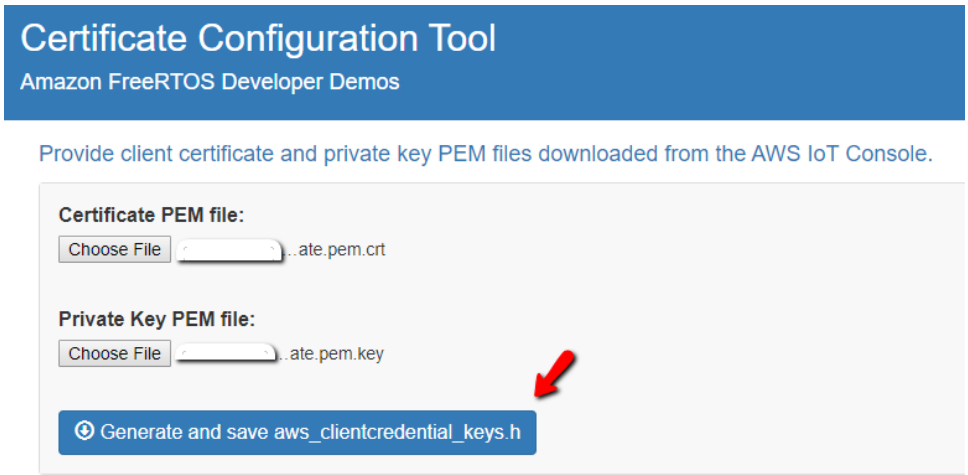

Figure 3. Generating aws_clientcredential_keys.h

d. Replace the `sfw/firmware/aws_ota/demos/include/aws_clientcredential_keys.h` with the file generated in Step c.

2. Modify the `aws_ota_codesigner_certificate.h` file.

   a. Open the `ecdsaigner.crt` file generated in the **Windows Pre-Requisites** part using a text editor. For details, see **Chapter 7.3.1.1 AWS OTA Prerequisites** in *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG).

   b. Open the `sfw/firmware/aws_ota/demos/include/aws_ota_codesigner_certificate.h` file.

   c. Copy all the contents in `ecdsaigner.crt` and paste to `aws_ota_codesigner_certificate.h` in the `signingcredentialSIGNING_CERTIFICATE_PEM`.

NOTE

Be sure to add **"** at the begging of a line and **\n"** on every line break, as shown in Figure 4.

```
/*
 * PEM-encoded code signer certificate
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----\n"
 * "...base64 data...\n"
 * "-----END CERTIFICATE-----\n";
 */
static const char signingcredentialSIGNING_CERTIFICATE_PEM[] =
"-----BEGIN CERTIFICATE-----\n"
"MIIBYTCCAQegAwIBAgIJAKCX9bIhkilFMAoGCCqGSM49BAMCMCMxITAfBgNVBAMM\n"
"GEFsZWphbmRyYS5HdXptYW5AbnhwLmNvbTAeFw0xOTEwMjMxNjMzNDJaFw0yMDEw\n"
"MjIxNjMzNDJaMCMxITAfBgNVBAMMGEFsZWphbmRyYS5HdXptYW5AbnhwLmNvbTBZ\n"
"MBMGByqGSM49AgEGCCqGSM49AwEHA0IABGUghBD51mF1J3wf4LYsQ2VgOaDpg98G\n"
"dNC38FWGS7owT4NC5848JumrD8SonnnXpu77Pt7ShuW39hC3Vdi7z1GjJDAiMAsG\n"
"A1UdDwQEAwIHgDATBgNVHSUEDDAKBggrBgEFBQcDAzAKBggqhkjOPQQDAgNIADBF\n"
"AiEAr0pNz1aMax4arCPNiW9HYFdQTvUGyZdRLcDrUo1/LQoCIH2U2REoZ59V7r6z\n"
"CMLfHA+kWq84IjxDUE2OgV60RVvC\n"
"-----END CERTIFICATE-----\n";
```

Figure 4. Certificate format

3. Enter the `sfw/target/evkmimxrt1170` path.
4. Double-click the batch file, `env.bat`.
5. In the `env.bat`, run the `scons --menuconfig` command to configure the `evkmimxrt1170` project.
6. In the configuration menu, select **MCU SFW core**, disable the `Enable sfw standalone xip` option, enable OTA, and select **AWS** OTA cloud.

```
[ ] Enable sfw standalone xip
[*] Enable OTA
[*]   OTA from cloud
[]      OTA Cloud Select (AWS)  --->
        AWS Config  --->
[*]   OTA from sdcard
[*]   OTA from u-disk
    MCU SFW Flash Map  --->
    MCU SFW metadata header  --->
```

Figure 5. SFW configuration

7. In the configuration menu, select **AWS Config** to enter the configuration menu, as shown in Figure 6.

```
()  Set MQTT broker DNS name (NEW)
()  Set IoT Thing name (NEW)
```

Figure 6. AWS configuration

8. In the **AWS Configuration** menu, input MQTT DNS name.
   a. Open AWS IoT console website.
   b. In the navigation pane, choose **Manage** → **Things**. Select the Thing previously created in the **Create an AWS IoT Thing** part. For details, see **Chapter 7.3.1.1 AWS OTA Prerequisites** in *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG).
   c. In the navigation pane, choose **Interact**.

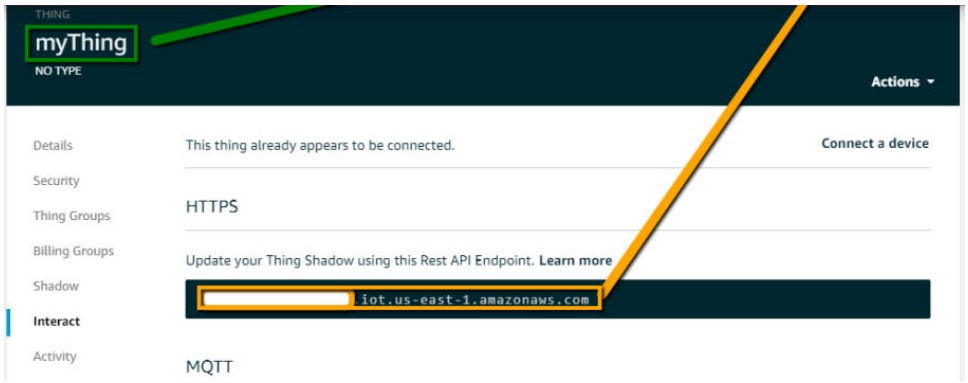

Figure 7. Interact of Thing

   d. In the **AWS Configuration** menu, select **Set MQTT broker DNS name**, copy **Rest API Endpoint**, as shown in Figure 7, and paste.

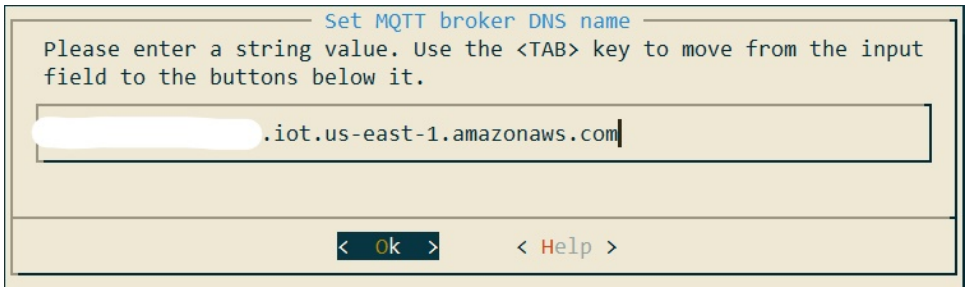

Figure 8. MQTT broker DNS name set

e. Select **<ok>**.

9. In the **AWS Configuration** menu, input IoT Thing name.

   a. Select **Set IoT Thing name**, copy **IoT Thing** name, as shown in Figure 7, and paste.

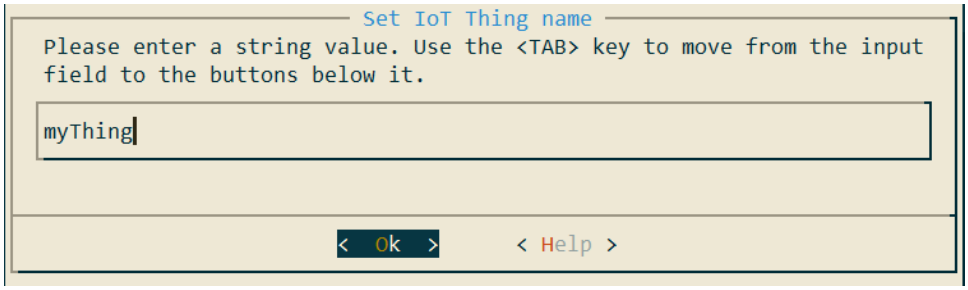

Figure 9. Thing name set

   b. Select **<ok>**.

10. In the configuration menu, select **MCU SFW Component -> secure**, select **enable mbedtls** and modify `mbedtls configuration` file to `aws_mbedtls_config.h`, and then select **<ok>**.

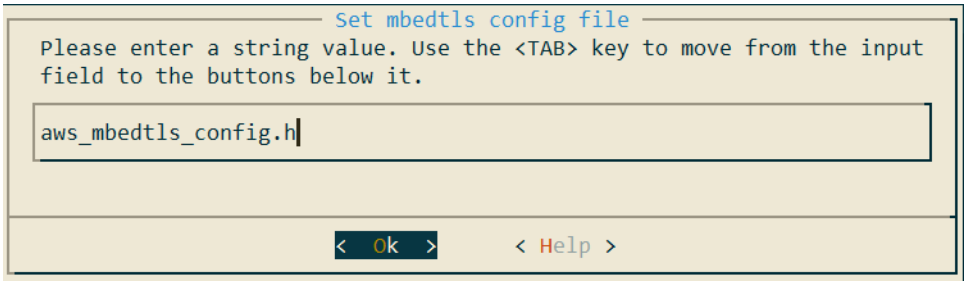

Figure 10. Modifying `mbedtls` configuration file

11. Exit and save the configurations.

4.3 Preparing image

For AWS OTA, prepare two SFW images: One is for downloading to the target board and the other is uploaded to AWS bucket for OTA upgrade. After SFW configurations complete, perform the following steps to generate an SFW image:

1. Enter the `sfw/target/evkmimxrt1170` path and double click the batch file, *env.bat*.
2. To generate an `iar` project, input the `scons --ide=iar` command.

NOTE

To generate a `keil` or `gcc` project, see **Chapter 2 Quick start** in *MCU-OTA SBL and SFW User Guide* (document MCUOTASBLSFWUG).

3. Enter the `sfw/target/evkmimxrt1170/iar` path and open the `sfw.eww` project.
4. Go to **Options**, select **Generate additional output**, and choose **Raw binary**.

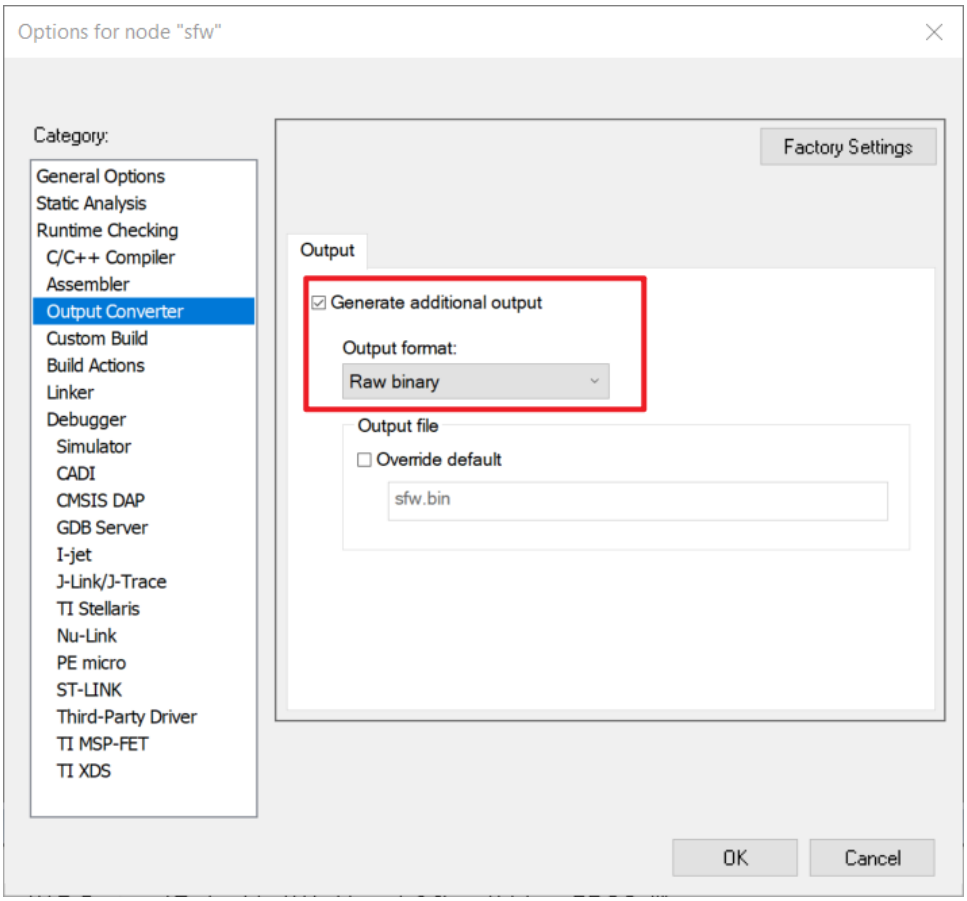

Figure 11. Modifying to generate `.bin` file

5. Check the application version in the `sfw/firmware/aws_ota/main_enet.c` file.

```
#define APP_VERSION_MAJOR 0
#define APP_VERSION_MINOR 9
#define APP_VERSION_BUILD 2
```

Figure 12. Application version

6. To start building the application, click the **Make** button.
7. If the build is successful, the `sfw.bin` is generated in the `sfw/target/evkmimxrt1170/iar/build/iar/Exe` folder. Change its name according to the application version. Move `sfw_092.bin` to the `sbl/component/secure/mcuboot/scripts` folder.
8. To build a newer image for OTA, change the value of `APP_VERSION_BUILD` to **3**. Rename the new bin file to `sfw_093.bin` and also move it to *sbl/component/secure/mcuboot/scripts* folder.

NOTE

The image version used for OTA upgrade must be newer than the currently running image version.

```
#define APP_VERSION_MAJOR 0
#define APP_VERSION_MINOR 9
#define APP_VERSION_BUILD 3
```

Figure 13. New version

9. To generate `sfw092.bin` and `sfw093.bin` files, sign `sfw_092.bin` and `sfw_093.bin` images with RSA by using the following commands.

```
python imgtool.py sign --key sign-rsa2048-priv.pem --align 4 --version "0.9.2" --header-size
0x400 --pad-header --slot-size 0x100000 --max-sectors 32 sfw_092.bin sfw092.bin
```

```
python imgtool.py sign --key sign-rsa2048-priv.pem --align 4 --version "0.9.3" --header-size
0x400 --pad-header --slot-size 0x100000 --max-sectors 32 sfw_093.bin sfw093.bin
```

4.4 Uploading new image to S3 bucket

After SFW image is prepared, upload the image with newer version to AWS S3 bucket.

1. Use AWS console to open the S3 service, https://console.aws.amazon.com/s3.
2. Select the bucket previously created in AWS configuration.
3. Click **Upload**.
4. Drag and drop `sfw093.bin`.
5. Click **Upload**.

4.5 Running the application

Before updating the OTA, download the SFW application to the target board.

1. Use the MCUBootUtility tool to download the `sfw092.bin` generated in Preparing image to the first slot of the board. The default location of Slot1 is the `flash_offset+0x100000` to `flash_offset+0x200000`. The whole slot size is 1 MB.

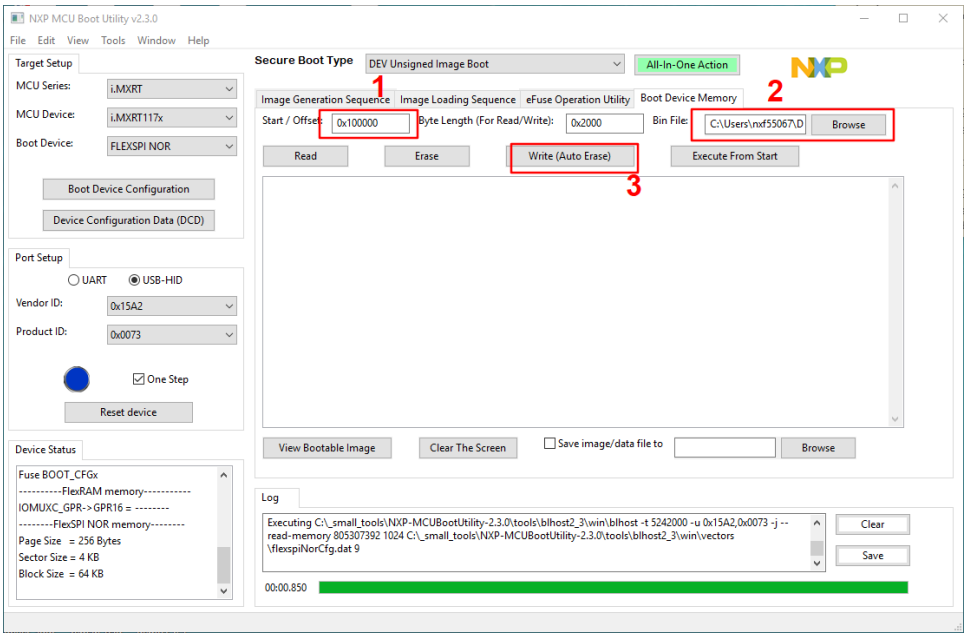

Figure 14. Downloading `sfw092.bin` to Slot1

2. After successfully downloading the image, to start the project, insert the Ethernet cable to the 1 G port (**J4**) on the RT1170-EVK board and press the **Reset** button. The serial terminal prints the application log, as shown in Figure 15.

   The serial terminal displays **The image now in PRIMARY_SLOT slot** and **Getting IP address from DHCP…**, indicating that the program in Slot 1 is running successfully. **IPv4 Address: 192.168.8.106** and **OTA demo version 0.9.2** indicate the successful network connection and the version of the current application.

```
hello sbl.
Bootloader Version 0.0.1
Remap type: none

The image now in PRIMARY_SLOT slot

Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the first image slot
hello sfw!
host init done
This example to demonstrate how to use U-Disk to implement ota.
Hello world1.
Hello world2.
Initializing PHY...
This example to demonstrate how to use SD card to implement ota.
0 49 [Tmr Svc] Write certificate...
Hello world1.
Hello world2.
Hello world1.
Hello world2.
Hello world1.
Hello world2.
Please insert a card into board.
Please plug in a u-disk to board.
1 3324 [Tmr Svc] Getting IP address from DHCP ...
2 17326 [Tmr Svc] IPv4 Address: 192.168.8.106
3 17326 [Tmr Svc] DHCP OK
4 17333 [iot_thread] [INFO ][DEMO][17329] ---------STARTING DEMO---------

5 17345 [iot_thread] [INFO ][INIT][17345] SDK successfully initialized.
6 17346 [iot_thread] [INFO ][DEMO][17346] Successfully initialized the demo. Network type for the d7 17347 [iot_thread] [INFO ][MQTT][17347] MQTT library succ
essfully initialized.
8 17347 [iot_thread] OTA demo version 0.9.2
9 17348 [iot_thread] Creating MQTT Client...
10 27520 [iot_thread] Connecting to broker...
11 27520 [iot_thread] [INFO ][MQTT][27520] Establishing new MQTT connection.
12 27541 [iot_thread] [INFO ][MQTT][27540] Anonymous metrics (SDK language, SDK version) will be pr13 27553 [iot_thread] [INFO ][MQTT][27553] (MQTT connection
 202d61a8, CONNECT operation 202d62c0) W14 27812 [iot_thread] [INFO ][MQTT][27811] (MQTT connection 202d61a8, CONNECT operation 202d62c0) W15 27824 [iot_threa
d] [INFO ][MQTT][27823] New MQTT connection 202c169c established.
16 27824 [iot_thread] Connected to broker.
17 27840 [iot_thread] [OTA_AgentInit_internal] OTA Task is Ready.
18 27847 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [Ready] Event [Start] New19 27866 [OTA Agent Task] [INFO ][MQTT][27865] (MQTT connec
tion 202d61a8) SUBSCRIBE operation sched20 27875 [OTA Agent Task] [INFO ][MQTT][27875] (MQTT connection 202d61a8, SUBSCRIBE operation 202d6Hello world1.
Hello world2.
21 28150 [OTA Agent Task] [INFO ][MQTT][28149] (MQTT connection 202d61a8, SUBSCRIBE operation 202d622 28160 [OTA Agent Task] [prvSubscribeToJobNotificationTop
ics] OK: $aws/things/SFWOTA1060/jobs/$next23 28177 [OTA Agent Task] [INFO ][MQTT][28177] (MQTT connection 202d61a8) SUBSCRIBE operation sched24 28187 [OTA Agent
 Task] [INFO ][MQTT][28187] (MQTT connection 202d61a8, SUBSCRIBE operation 202d625 28422 [OTA Agent Task] [INFO ][MQTT][28422] (MQTT connection 202d61a8, SUBS
CRIBE operation 202d626 28434 [OTA Agent Task] [prvSubscribeToJobNotificationTopics] OK: $aws/things/SFWOTA1060/jobs/not27 28443 [OTA Agent Task] [prvRequestJ
ob_Mqtt] Request #0
28 28460 [OTA Agent Task] [INFO ][MQTT][28459] (MQTT connection 202d61a8) MQTT PUBLISH operation qu29 28469 [OTA Agent Task] [INFO ][MQTT][28469] (MQTT connec
tion 202d61a8, PUBLISH operation 202d63b30 28685 [OTA Agent Task] [INFO ][MQTT][28685] (MQTT connection 202d61a8, PUBLISH operation 202d63b31 28711 [OTA Agent
 Task] [prvOTAAgentTask] Called handler. Current State [RequestingJob] Event [Re32 28720 [OTA Agent Task] [prvParseJobDoc] Size of OTA_FileContext_t [64]
37 28763 [OTA Agent Task] [prvParseJSONbyModel] parameter not present: afr_ota
38 28771 [OTA Agent Task] [prvParseJSONbyModel] parameter not present: protocols
39 28779 [OTA Agent Task] [prvParseJSONbyModel] parameter not present: files
40 28787 [OTA Agent Task] [prvParseJSONbyModel] parameter not present: filepath
41 28794 [OTA Agent Task] [prvParseJSONbyModel] parameter not present: filesize
42 28802 [OTA Agent Task] [prvParseJSONbyModel] parameter not present: fileid
43 28810 [OTA Agent Task] [prvParseJSONbyModel] parameter not present: certfile
44 28818 [OTA Agent Task] [prvParseJSONbyModel] parameter not present: sig-sha256-ecdsa
45 28827 [OTA Agent Task] [prvDefaultCustomJobCallback] Received Custom Job inside OTA Agent which 46 28836 [OTA Agent Task] [prvParseJobDoc] Ignoring job wit
hout ID.
47 28840 [iot_thread] State: Ready  Received: 1   Queued: 0   Processed: 0   Dropped: 0
48 28851 [OTA Agent Task] [prvOTA_Close] Context->0x202e65d4
49 28857 [OTA Agent Task] [OTA-NXP] Abort
50 28862 [OTA Agent Task] [OTA-NXP] SetPlatformImageState 4
51 28868 [OTA Agent Task] [OTA-NXP] GetPlatformImageState
52 28874 [OTA Agent Task] [OTA-NXP] ota_status = 0x0
53 28880 [OTA Agent Task] [prvOTAAgentTask] Handler failed. Current State [WaitingForJob] Event  [RHello world1.
Hello world2.
54 29840 [iot_thread] State: WaitingForJob  Received: 1   Queued: 0   Processed: 0   Dropped: 0
```

Figure 15. Application log

3. When running the application, wait until the **OTA State Ready** message is shown on the serial terminal, as shown in Figure 16. This message means that the OTA agent is ready and is waiting for an OTA job.

```
54 28610 [iot_thread] State: WaitingForJob  Received: 1   Queued: 0   Processed: 0   Dropped: 0
Hello world1.
Hello world2.
55 29610 [iot_thread] State: WaitingForJob  Received: 1   Queued: 0   Processed: 0   Dropped: 0
Hello world1.
Hello world2.
56 30610 [iot_thread] State: WaitingForJob  Received: 1   Queued: 0   Processed: 0   Dropped: 0
Hello world1.
Hello world2.
```

Figure 16. OTA ready log

Now, the whole preparation is complete and the AWS OTA update job is started to implement OTA process.

4.6 OTA update process

To implement the OTA process via AWS OTA, perform the following steps to create a job.

1. Open the AWS IoT console website.
2. In the navigation pane, choose **Manage** -> **Jobs**.
3. Select **Create job**.
4. Choose **Create FreeRTOS OTA update job** and then click **Next**.
5. Input **Job name** and then click **Next**.

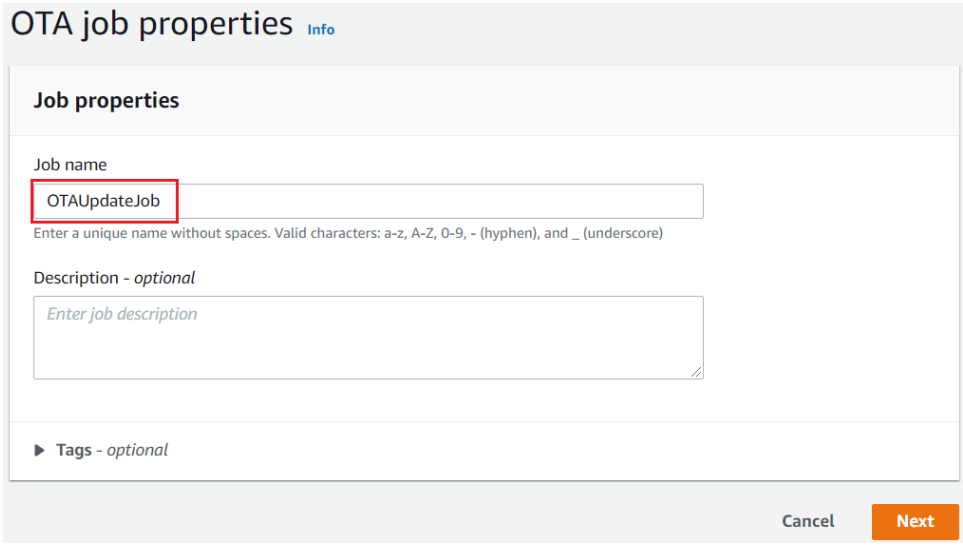

Figure 17. Inputting job name

6. Choose **Thing** created in AWS configuration. Then, choose **MQTT** and **Sign a new file for me**.

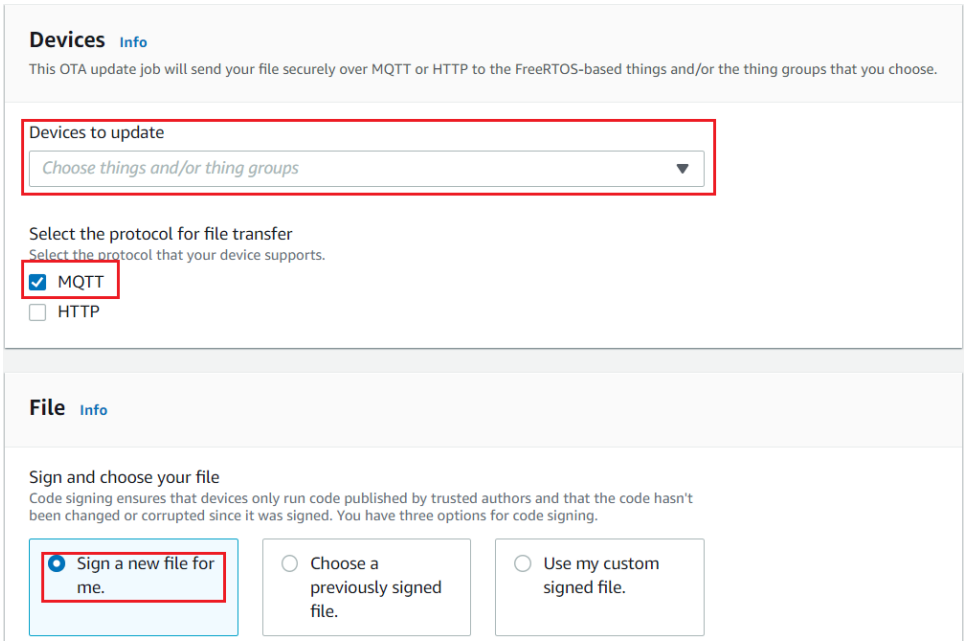

Figure 18. Selecting Thing, protocol and file

7. Under **Code signing profile**, choose **Create new profile**.
8. Enter a name for the code-signing profile.
    a. Under **Device hardware platform**, choose **Windows Simulator**.

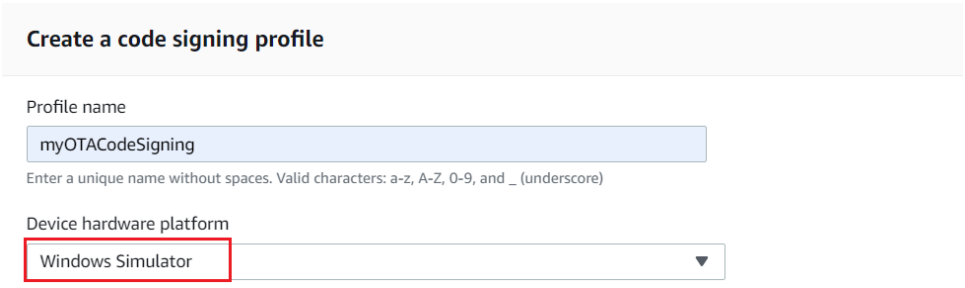

Figure 19. Device hardware platform

b. Under **Code signing certificate**, choose **Import new code signing certificate**, browse for the certificate files created with AWS CLI in AWS configuration, and then choose **Import**.

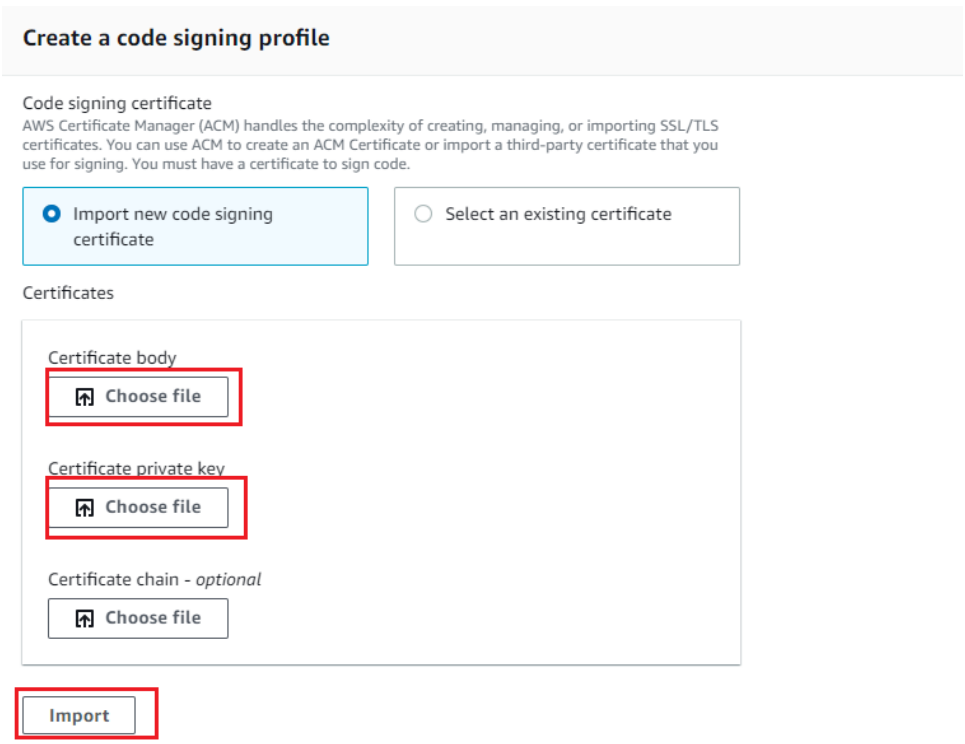

Figure 20. Selecting certificate and key

c. Under **Pathname of code signing certificate on device**, type the default `path/certificates/authcert.pem`, and then click **Create**.

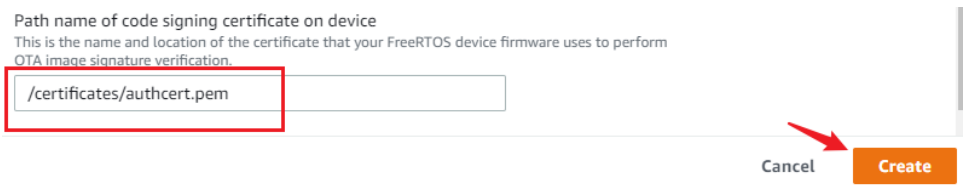

Figure 21. Pathname of certificate

9. Under **File**, choose **Select an existing file**. Then, choose **Browse S3** and select `sfw093.bin` file previously uploaded in S3.

NOTE

Make sure that the region is the correct one where the bucket is located. Otherwise, the uploaded binary can't be found.

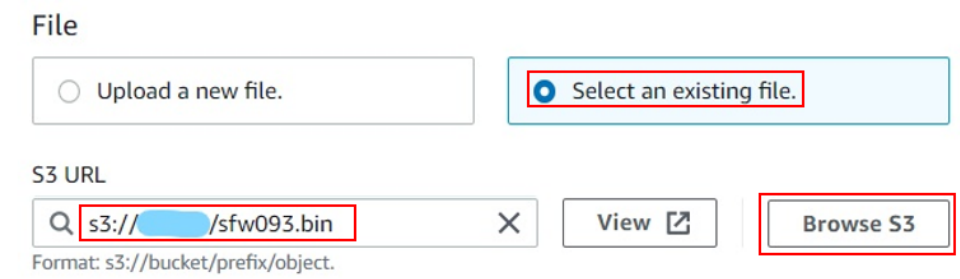

Figure 22. Selecting image

10. Under **Pathname of file on device**, type the default **path /device/updates**.

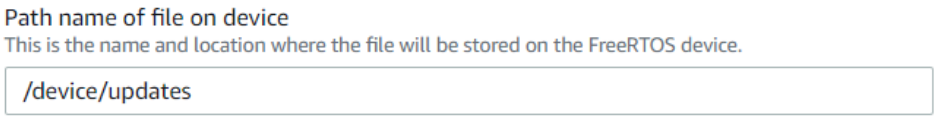

Figure 23. Pathname of file

11. Under **IAM role**, choose the role created in AWS configuration.

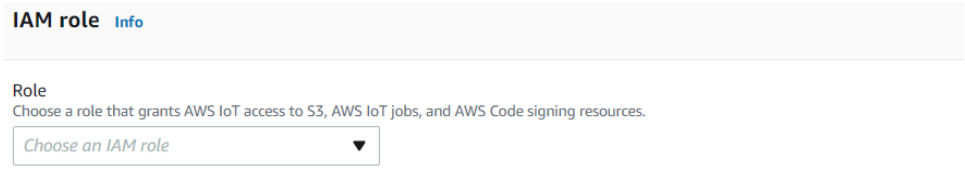

Figure 24. Selecting a role

12. Choose **Next**.
13. Under **Job run type**, choose the first item and then click **Create job**.

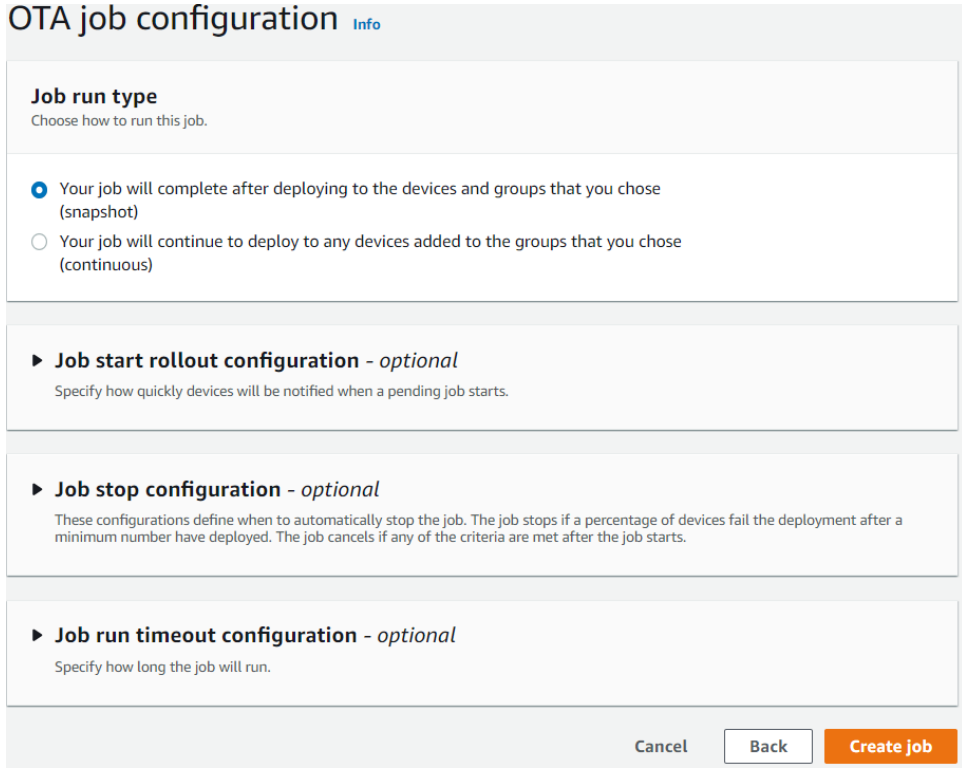

Figure 25. Job run type

14. The OTA process starts and the serial terminal outputs are shown as below.

    a. Transfer the file.

```
70 40447 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ filesize: 475842 ]
71 40455 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ fileid: 0 ]
72 40464 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ certfile: /certificates/authc73 40473 [OTA Agent Task] [prvParseJSONbyModel] Extracted p
arameter [ sig-sha256-ecdsa: MEUCICnhyW+74 40483 [OTA Agent Task] [prvParseJobDoc] Job was accepted. Attempting to start transfer.
75 40492 [OTA Agent Task] [OTA-NXP] GetPlatformImageState
76 40499 [OTA Agent Task] [OTA-NXP] ota_status = 0x0
77 40504 [OTA Agent Task] [OTA-NXP] CreateFileForRx
78 40510 [OTA Agent Task] [OTA-NXP] image_position= 1
79 40515 [OTA Agent Task] [OTA-NXP] File_Addr = 0x00200000, File_Size = 0x00100000
80 40524 [OTA Agent Task] [prvSetDataInterface] Data interface is set to MQTT.
81 40531 [OTA Agent Task] [prvProcessJobHandler] Setting OTA data inerface.
92 41394 [OTA Agent Task] [prvIngestDataBlock] Received file block 4, size 1024
93 41403 [OTA Agent Task] [OTA-NXP] WriteBlock 1000 : 400
94 41413 [OTA Agent Task] [prvIngestDataBlock] Remaining: 464
```

Figure 26. Start file transfer log

    b. Receive the whole file.

```
2254 150969 [OTA Agent Task] [OTA-NXP] WriteBlock 74000 : 2c2
2255 150976 [OTA Agent Task] [prvIngestDataBlock] Received final expected block of file.
2256 150985 [OTA Agent Task] [prvStopRequestTimer] Stopping request timer.
```

Figure 27. Received whole file log

    c. Check the file signature.

```
2257 150992 [OTA Agent Task] [OTA-NXP] CloseFile
2258 150997 [OTA Agent Task] [OTA-NXP] CheckFileSignature
```

Figure 28. File signature check log

    d. Check the image version.

```
2266 156992 [OTA Agent Task] [OTA-NXP] cmp result=1
2267 156997 [OTA Agent Task] [OTA-NXP] new image verison: 0.9.3
2268 157004 [OTA Agent Task] [prvIngestDataBlock] File receive complete and signature is valid.
2269 157013 [OTA Agent Task] [prvStopRequestTimer] Stopping request timer.
2270 157021 [OTA Agent Task] [prvUpdateJobStatus_Mqtt] Msg: {"status":"IN_PROGRESS","statusDetails"2271 157045 [OTA Agent Task] [INFO ][MQTT][157045] (MQTT co
nnection 202d61a8) MQTT PUBLISH operatio2272 157055 [OTA Agent Task] [INFO ][MQTT][157055] (MQTT connection 202d61a8, PUBLISH operation 202Hello world1.
```

Figure 29. Image version check log

e. Write the update type.

```
2282 158200 [OTA Agent Task] [OTA-NXP] Write update type
write update type = 0x3
```

Figure 30. Write update type log

f. Write the image trailer.

```
2283 158208 [OTA Agent Task] [OTA-NXP] Write image trailer
write magic number offset = 0xffff0
```

Figure 31. Write image trailer log

g. Active the new image and the device resets.

```
2284 158218 [OTA Agent Task] [OTA-NXP] ActivateNewImage
2285 158224 [OTA Agent Task] [OTA-NXP] ResetDevice
```

Figure 32. Active new image log

h. Run the new image.

```
hello sbl.
Bootloader Version 0.0.1
Remap type: test

The image now in SECONDARY_SLOT slot

Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the first image slot
hello sfw!
host init done
This example to demonstrate how to use U-Disk to implement ota.
Hello world1.
Hello world2.
Initializing PHY...
This example to demonstrate how to use SD card to implement ota.
0 49 [Tmr Svc] Write certificate...
2 17172 [Tmr Svc] IPv4 Address: 192.168.8.106
3 17172 [Tmr Svc] DHCP OK
4 17175 [iot_thread] [INFO ][DEMO][17175] --------STARTING DEMO--------

5 17191 [iot_thread] [INFO ][INIT][17191] SDK successfully initialized.
6 17192 [iot_thread] [INFO ][DEMO][17192] Successfully initialized the demo. Network type for the d7 17193 [iot_thread] [INFO ][MQTT][17193] MQTT library succ
essfully initialized.
8 17193 [iot_thread] OTA demo version 0.9.3
9 17194 [iot_thread] Creating MQTT Client...
```

Figure 33. Run new image log

i. Perform the self-test.

```
56 27756 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [WaitingForJob] Event [Re57 27765 [OTA Agent Task] [prvInSelfTestHandler] prvInSelfT
estHandler, platform is in self-test.
58 27774 [OTA Agent Task] [OTA-NXP] GetPlatformImageState
59 27781 [OTA Agent Task] [OTA-NXP] ota_status = 0x1
```

Figure 34. Self-test log

j. Write the **OK** flag.

```
64 27814 [OTA Agent Task] [OTA-NXP] ota_status = 0x1
Write OK flag: off = 0xfffe0
```

Figure 35. Write OK flag log

k. OTA succeeds.

```
65 27824 [OTA Agent Task] [prvStopSelfTestTimer] Stopping the self_test_timer.
66 27832 [OTA Agent Task] [prvUpdateJobStatus_Mqtt] Msg: {"status":"SUCCEEDED","statusDetails":{"re67 27855 [OTA Agent Task] [INFO ][MQTT][27855] (MQTT connec
tion 202d61a8) MQTT PUBLISH operation qu68 27865 [OTA Agent Task] [INFO ][MQTT][27865] (MQTT connection 202d61a8, PUBLISH operation 202d63bHello world1.
Hello world2.
69 28113 [OTA Agent Task] [INFO ][MQTT][28113] (MQTT connection 202d61a8, PUBLISH operation 202d63b70 28124 [OTA Agent Task] [prvUpdateJobStatus_Mqtt] 'SUCCEE
DED' to $aws/things/SFW0TA1060/jobs/AFR_71 28133 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [CreatingFile] Event [Sta72 28435 [iot_threa
d] State: WaitingForJob  Received: 1   Queued: 0   Processed: 0   Dropped: 0
```

Figure 36. OTA success log

Now go back to the AWS IoT console website to view the status of the OTA job. The **Completed** status of **Job details** with **Succeeded** status of **Job executions** indicate the success of this OTA update.

15. After OTA succeeds, press the **Reset** button on the board to double confirm whether OTA update is successful. If the update is successful, the `sfw093.bin` log is printed, as shown in Figure 37.

```
hello sbl.
Bootloader Version 0.0.1
Remap type: none

The image now in SECONDARY_SLOT slot

Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the first image slot
hello sfw!
host init done
This example to demonstrate how to use U-Disk to implement ota.
Hello world1.
Hello world2.
Initializing PHY...
This example to demonstrate how to use SD card to implement ota.
0 49 [Tmr Svc] Write certificate...
Hello world1.
Hello world2.
Hello world1.
Hello world2.
Hello world1.
Hello world2.
Please insert a card into board.
Please plug in a u-disk to board.
1 3291 [Tmr Svc] Getting IP address from DHCP ...
2 17293 [Tmr Svc] IPv4 Address: 192.168.8.106
3 17293 [Tmr Svc] DHCP OK
4 17300 [iot_thread] [INFO ][DEMO][17296] ---------STARTING DEMO---------

5 17312 [iot_thread] [INFO ][INIT][17311] SDK successfully initialized.
6 17313 [iot_thread] [INFO ][DEMO][17312] Successfully initialized the demo. Network type for the d7 17313 [iot_thread] [INFO ][MQTT][17313] MQTT library succ
essfully initialized.
8 17314 [iot_thread] OTA demo version 0.9.3
9 17314 [iot_thread] Creating MQTT Client...
10 26240 [iot_thread] Connecting to broker...
11 26240 [iot_thread] [INFO ][MQTT][26240] Establishing new MQTT connection.
12 26247 [iot_thread] [INFO ][MQTT][26247] Anonymous metrics (SDK language, SDK version) will be pr13 26280 [iot_thread] [INFO ][MQTT][26279] (MQTT connection
 202d61a8, CONNECT operation 202d62c0) W14 26502 [iot_thread] [INFO ][MQTT][26501] (MQTT connection 202d61a8, CONNECT operation 202d62c0) W15 26514 [iot_threa
d] [INFO ][MQTT][26513] New MQTT connection 202c169c established.
16 26514 [iot_thread] Connected to broker.
17 26529 [iot_thread] [OTA_AgentInit_internal] OTA Task is Ready.
18 26535 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [Ready] Event [Start] New19 26554 [OTA Agent Task] [INFO ][MQTT][26554] (MQTT connec
tion 202d61a8) SUBSCRIBE operation sched20 26564 [OTA Agent Task] [INFO ][MQTT][26564] (MQTT connection 202d61a8, SUBSCRIBE operation 202d621 26804 [OTA Agent
 Task] [INFO ][MQTT][26803] (MQTT connection 202d61a8, SUBSCRIBE operation 202d622 26814 [OTA Agent Task] [prvSubscribeToJobNotificationTopics] OK: $aws/thing
s/SFW0TA1060/jobs/$ne23 26833 [OTA Agent Task] [INFO ][MQTT][26833] (MQTT connection 202d61a8) SUBSCRIBE operation sched24 26843 [OTA Agent Task] [INFO ][MQTT
][26843] (MQTT connection 202d61a8, SUBSCRIBE operation 202d6Hello world1.
Hello world2.
25 27065 [OTA Agent Task] [INFO ][MQTT][27064] (MQTT connection 202d61a8, SUBSCRIBE operation 202d626 27077 [OTA Agent Task] [prvSubscribeToJobNotificationTop
ics] OK: $aws/things/SFW0TA1060/jobs/not27 27086 [OTA Agent Task] [prvRequestJob_Mqtt] Request #0
28 27104 [OTA Agent Task] [INFO ][MQTT][27103] (MQTT connection 202d61a8) MQTT PUBLISH operation qu29 27114 [OTA Agent Task] [INFO ][MQTT][27113] (MQTT connec
tion 202d61a8, PUBLISH operation 202d63b30 27326 [OTA Agent Task] [INFO ][MQTT][27325] (MQTT connection 202d61a8, PUBLISH operation 202d63b31 27336 [OTA Agent
 Task] [prvOTAAgentTask] Called handler. Current State [RequestingJob] Event [Re32 27346 [OTA Agent Task] [prvParseJobDoc] Size of OTA_FileContext_t [64]
52 27494 [OTA Agent Task] [prvOTAAgentTask] Handler failed. Current State [WaitingForJob] Event  [R53 27529 [iot_thread] State: Ready  Received: 1   Queued: 0
   Processed: 0   Dropped: 0
Hello world1.
Hello world2.
54 28529 [iot_thread] State: WaitingForJob  Received: 1   Queued: 0   Processed: 0   Dropped: 0
Hello world1.
Hello world2.
55 29529 [iot_thread] State: WaitingForJob  Received: 1   Queued: 0   Processed: 0   Dropped: 0
```

Figure 37. New image log

5 References

1. SBL project
2. SFW project

6 Revision history

Revision number	Date	Substantive changes
0	6 December 2021	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 6 December 2021
Document identifier: AN13469