NXP Semiconductors
User’s Guide

Document Number: KTSOTAFUUG
Rev. 5, 12/2016

Kinetis Thread Stack Over-the-Air (OTA) Firmware Update User’s Guide

1 About This Document

This document provides an overview of the Over the Air Update (OTA) module, interface, and usage for the Kinetis Thread Stack.

Contents

2 OTA Firmware Updates for Kinetis Thread Stack

The system setup consists of one OTA server and one or more clients, with potential intermediary multihop routers, as shown in Figure 1. The serial connections with PCs are for displaying status messages and for sending messages to the devices in the network. The OTA client serial connections are optional as the network nodes can also function in autonomous mode.

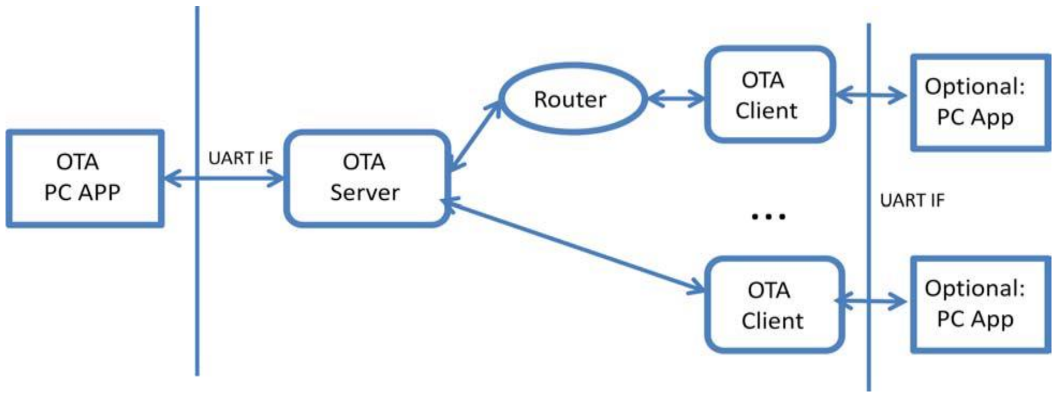


Figure 1. OTA Block Diagram

The download of an OTA image from server to client requires an application bootloader installed and sufficient memory (external or internal) to store the newly loaded image. The Thread framework includes an application interface in the OtaSupport header and module files for using the external storage modules while receiving the firmware image blocks over the air. The application can start to write a new image to the external storage, push one or more blocks to a storage location, or commit an image to indicate that the bootloader should use it for updating the internal flash at the next reboot.

2.1 OTA file format

The OTA file format is composed of a header followed by a number of sub-elements. The header describes general information about the file such as version, the manufacturer that created it, and the device it is intended for. Sub-elements in the file may contain upgrade data for the embedded device, certificates, configuration data, log messages, or other manufacturer-specific pieces.

Table 1 OTA file format using CRC

Name	**Header**	**ImageTag**	**Image**	**Bitmap Tag**	**Bitmap**	**CRC tag**	**CRC**
Size	60 bytes	6 bytes	Variable	6 bytes	32 bytes	6 bytes	4 bytes

The bitmap sub-element of an OTA cluster image file indicates the sectors that should or should not be erased and reprogrammed in the internal flash. Setting a bit to 1 allows the erasure of the corresponding flash sector. If a bit is set to 0, the corresponding flash sector is protected.

2.2 OTA commands:

2.2.1 gOtaCmd_ImageNotify_c (/otaserver) OTA Image Notify:

- Sent unicast or multicast the Image Notify command represents the way by which the server alerts clients that a new image is available.
- URI-Path NON POST coap://<dest>/otaserver
- CoAP payload:

Table 2 OTA Image Notify CoAP payload

Parameter	Size	Comments
CommandId	1	Command Identifier: gOtaCmd_ImageNotify_c = 0x00
TransferType	1	Transfer Type: Default Value = 0x00 (OtaUnicast)
ManufacturerCode	2	Manufacturer Code: Default Value = 0x1004
ImageType	2	Image Type: Default Value = 0x0000
ImageSize	4	ImageSize
FileSize	4	FileSize
FileVersion	4	New File version: Default Value = 0x40034005
ServerDownloadPort	2	ServerDownloadPort
FragmentSize	2	FragmentSize

Note: The TransferType field is only present in the projects in which OTA multicast feature is activated.

2.2.2 The TransferType field is only present in the projects in which OTA multicast feature is activated.gOtaCmd_QueryImageReq_c (/otaclient) OTA Query Image Request:

- Automatically sent when the client receives an Image Notify Command.
- URI-Path NON GET coap://<dest>/otaclient
- CoAP payload:

Table 3 Query Image Request CoAP payload

Parameter	Size	Comments
CommandId	1	Command Identifier: gOtaCmd_ QueryImageReq _c = 0x01
ManufacturerCode	2	Manufacturer Code: Default Value = 0x1004
ImageType	2	Image Type: Default Value = 0x0000
FileVersion	4	Current File version
Hardware Version	2	Hardware version

2.2.3 gOtaCmd_QueryImageRsp_c (/otaserver) OTA Query Image Response

- The server response for Query Image Req command. Based on the status parameter, the OTA client decides whether or not to start downloading the new image.
- URI-Path NON POST coap://<dest>/otaserver
- CoAP payload:

Table 4 OTA Query Image Response CoAP payload

Parameter	Size	Comments	
CommandId	1	Command Identifier: gOtaCmd_ QueryImageRsp_c = 0x02	

Status	1	Possible values:
		gOtaFileStatus_Success_c = 0x00
		gOtaFileStatus_WaitForData_c = 0x97
		gOtaFileStatus_NoImageAvailable_c = 0x98	

Data	Variable	If status = gOtaFileStatus_Success_c	ManufacturerCode [2 bytes]
			ImageType(2 bytes)
			FileVersion[4 bytes]
			FileSize[4 bytes]
			Server Download Port[2 bytes]
		If status != gOtaFileStatus_Success_c	CurrentTime [4 bytes]
			RequestTime[4 bytes]

2.2.4 gOtaCmd_BlockReq_c (/otaclient) OTA Block Request

- This command is used to download the new image, by requesting a specific block. This command uses a socket open on a port that is announced by the Server on a ML_EID address.
- Default port: 0XF0BE
- Socket payload:

Table 5 OTA Block Request payload

Parameter	Size	Comments
CommandId	1	Command Identifier: gOtaCmd_ BlockReq _c = 0x03
ManufacturerCode	2	Manufacturer Code: Default Value = 0x1004
ImageType	2	Image Type: Default Value = 0x0000
FileVersion	4	Downloaded File version
FileOffset	4	Image Offset
MaxDataSize	1	Maximum block length

2.2.5 gOtaCmd_BlockRsp_c (/otaclient) OTA Block Response

- This is the server response for Block Req command. Based on the status parameter, the OTA client decides whether or not to continue the download procedure. This command uses a socket open on a port that is announced by the Server on a ML_EID address. The response is sent on the port on which the command is received.
- Default port: 0XF0BE
- Socket payload:

Table 6 OTA Block Response payload

Parameter	Size	Comments	
CommandId	1	Command Identifier: gOtaCmd_ BlockRsp_c = 0x04	

Status	1	Possible values:
		gOtaFileStatus_Success_c = 0x00
		gOtaFileStatus_Abort_c = 0x95
		gOtaFileStatus_NotAuthorized_c = 0x7E
		gOtaFileStatus_InvalidImage_c = 0x96
		gOtaFileStatus_ServerBusy_c = 0x97
		gOtaFileStatus_NoImageAvailable_c = 0x98	

Data	Variable	If status = gOtaFileStatus_Success_c	FileVersion [4 bytes]
			FileOffset [4 bytes]
			DataSize[1 byte]
			Data [DataSize param bytes]
		If status != gOtaFileStatus_Success_c	CurrentTime [4 bytes]
			RequestTime[4 bytes]

2.2.6 gOtaCmd_UpgradeEndReq_c (/otaclient) OTA Upgrade End Request

- This command is used to complete the transfer
- URI-Path NON GET coap://<dest>/otaclient
- CoAP payload:

Table 7 OTA Upgrade End Request CoAP payload

Parameter	Size	Comments
CommandId	1	Command Identifier: gOtaCmd_UpgradeEndReq_c = 0x05

Status	1	Possible values:
		gOtaFileStatus_Success_c = 0x00
		gOtaFileStatus_Abort_c = 0x95
		gOtaFileStatus_InvalidImage_c = 0x96

ManufacturerCode	2	Manufacturer Code: Default Value = 0x1004
ImageType	2	Image Type: Default Value = 0x0000
FileVersion	4	Downloaded File version

2.2.7 gOtaCmd_UpgradeEndRsp_c (/otaserver) OTA Upgrade End Response

- This is the server response for Upgrade End Req command and contains the delay used by the client before a re-flash/reboot to the new image (only if it detects the transfer procedure has been successful).
- URI-Path NON POST coap://<dest>/otaserver
- CoAP payload:

Table 8 OTA Upgrade End Response CoAP payload

Parameter	Size	Comments
CommandId	1	Command Identifier: gOtaCmd_UpgradeEndRsp_c = 0x06

Status	1	Possible values:
		gOtaFileStatus_Success_c = 0x00
		gOtaFileStatus_Abort_c = 0x95
		gOtaFileStatus_NotAuthorized_c = 0x7E
		gOtaFileStatus_InvalidImage_c = 0x96
		gOtaFileStatus_ServerBusy_c = 0x97
		gOtaFileStatus_NoImageAvailable_c = 0x98

CurrentTime	4	Current time
RequestTime	4	Request time
FileVersion	4	File Version

2.2.8 gOtaCmd_ServerDiscovery_c (/otaclient) OTA Server discovery

- This command is sent multicast and represents the way to discover a server by requesting a specific manufacturer and image type. A server will respond with an Image Notify unicast if success.
- URI-Path NON POST coap://<dest>/otaclient
- CoAP payload:

Table 9 OTA Server discovery CoAP payload

Parameter	Size	Comments
CommandId	1	Command Identifier: gOtaCmd_ImageNotify_c = 0x00
ManufacturerCode	2	Manufacturer Code: Default Value = 0x1004
ImageType	2	Image Type: Default Value = 0x0000

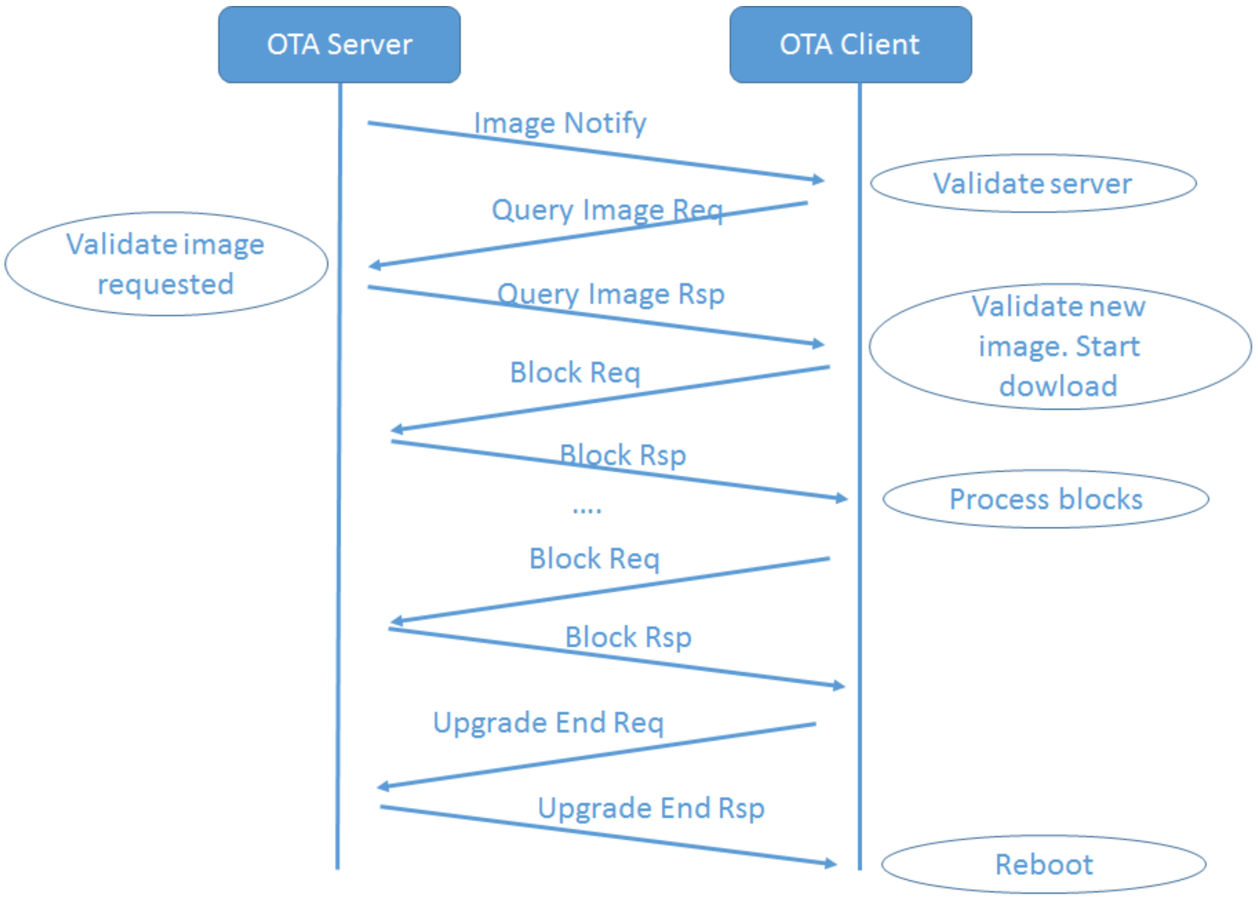


Figure 2. OTA Upgrade Diagram

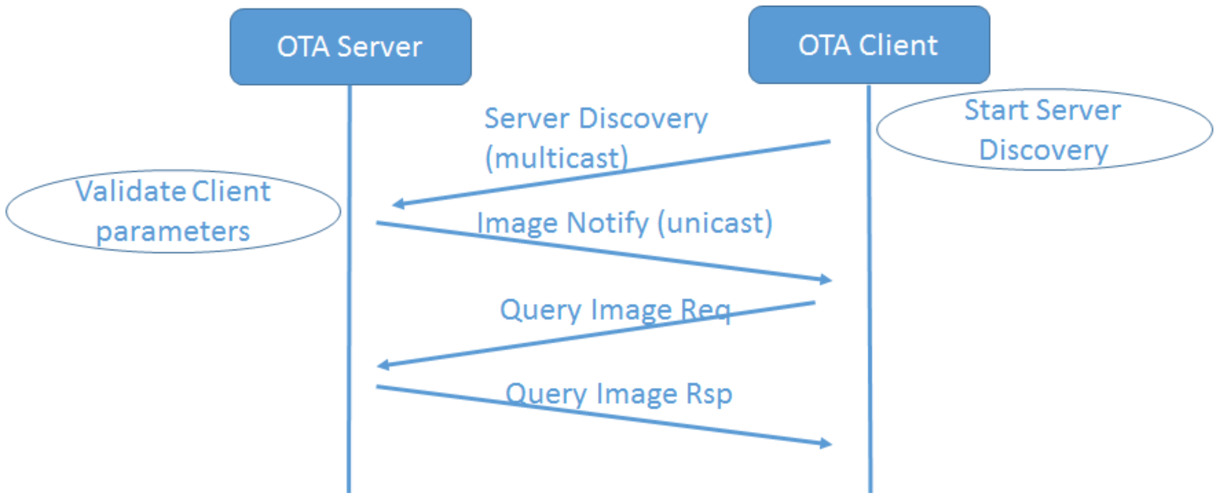


Figure 3. OTA Server Discovery Diagram

3 Software implementation

The files app_ota.h, app_ota_client.c, app_ota_server.c, OtaSupport.c, OtaSupport.h comprise the OTA Upgrade functionality. All configurations are done at compile time. The configuration settings needed to enable the OTAServer and OTAClient functionalities are available in the config.h file of the corresponding example project and have already been set to the appropriate values.

- gEnableOTAServer_d = TRUE – enables the OTA Server functionality;
- gEnableOTAClient_d = TRUE – enables the OTA Client Functionality;
- gEepromType_d = gEepromDevice_ AT45DB161E_c – use this particular EEPROM device type to store the image in the external flash of the client;

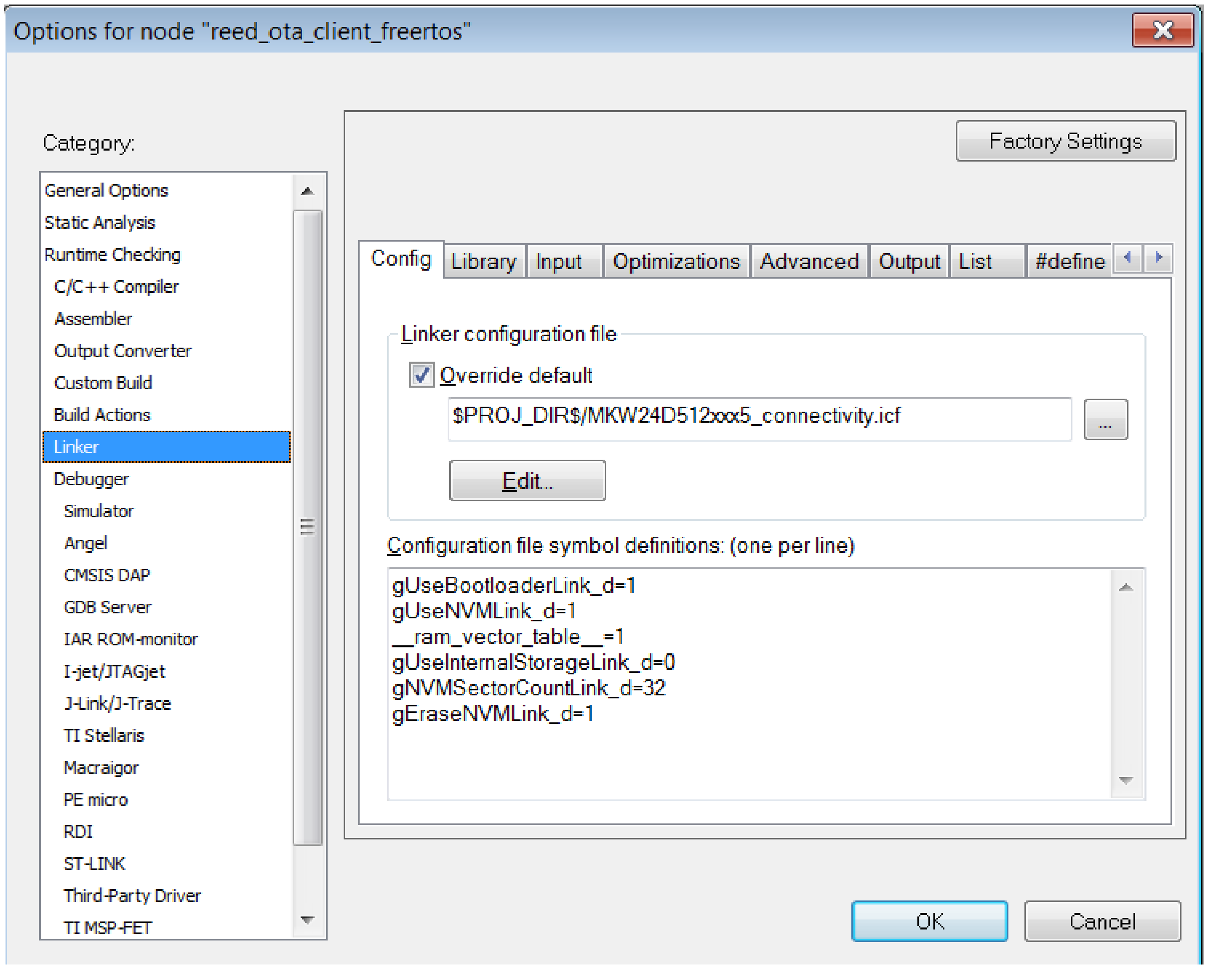
OTA client/server initialization is performed by the following API:

- OtaClientInit(mpAppThreadMsgQueue);
- OtaServerInit(mpAppThreadMsgQueue);

OTA client project options:

- gUseBootloaderLink_d=1 – sets the linker configuration file to reserve the first flash sector for bootloader use with the application firmware following in the subsequent sectors
- gUseBootloaderLink_d=0 – does not use the internal storage; uses the external flash instead
- gNVMSectorCountLink_d=32 – allocates 32 sectors of NVM, each having 2 kb of memory
- gUseNVMLink_d=1 – sets the linker configuration to access the NVM
- __ram_vector_table__=1 – places the vector table in RAM and allows dynamic insertion of an ISR handler

The following figures describe how to update the symbols definition of the Linker configuration files for IAR IDE and KDS IDE.



Linker configuration file

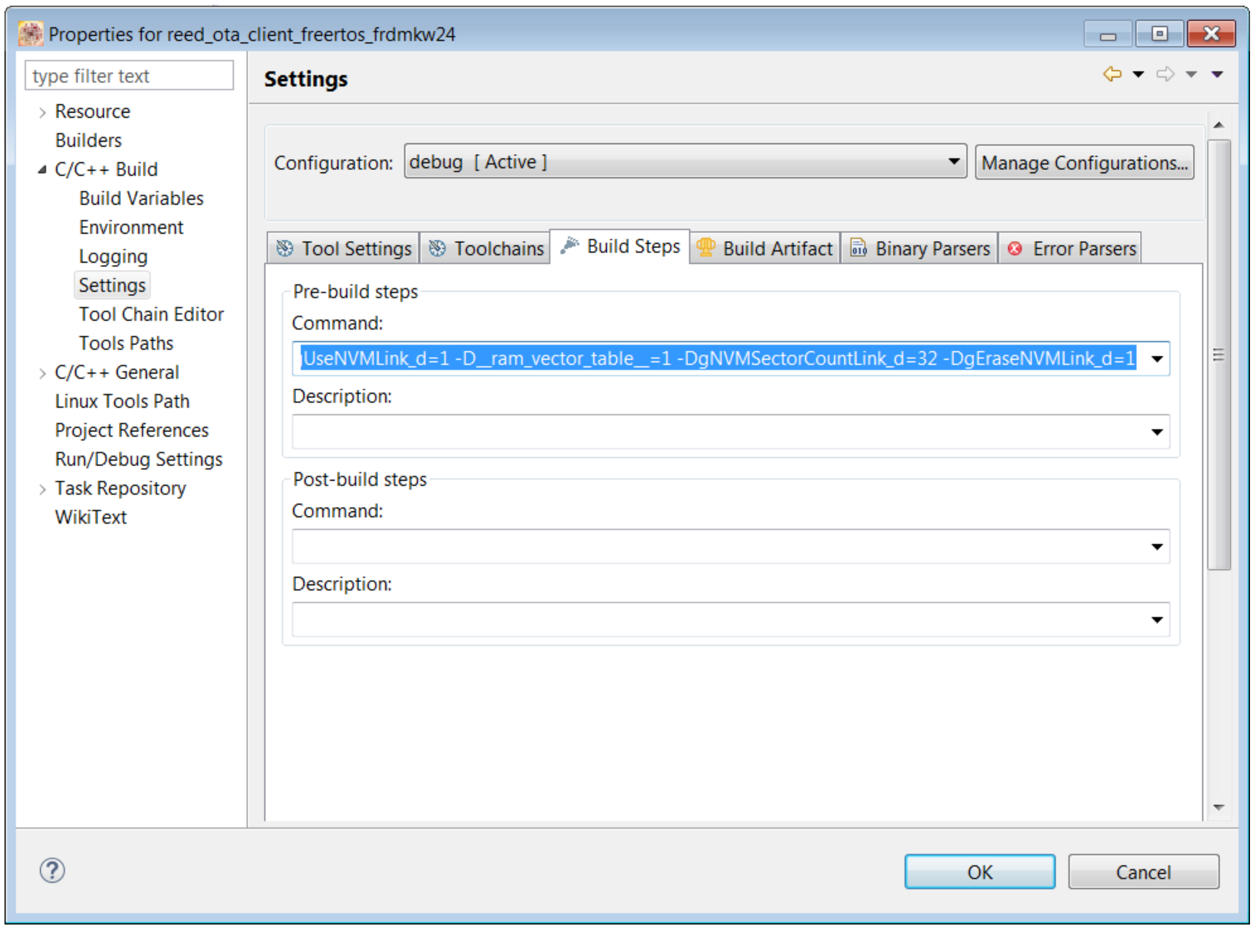
[✓] Override default

$PROJ_DIR$/MKW24D512xxx5_connectivity.icf

Configuration file symbol definitions: (one per line)

```
gUseBootloaderLink_d=1
gUseNVMLink_d=1
__ram_vector_table__=1
gUseInternalStorageLink_d=0
gNVMSectorCountLink_d=32
gEraseNVMLink_d=1
```

Figure 4. IAR Linker Configuration Options


Properties for reed_ota_client_freertos_frdmkw24

Settings

Configuration: debug [Active]

Pre-build steps

Command:

```
UseNVMLink_d=1 -D__ram_vector_table__=1 -DgNVMSectorCountLink_d=32 -DgEraseNVMLink_d=1
```

Figure 5. KDS Linker Configuration Options

4 Test Tool OTA View

The OTAP Thread view used for uploading binary firmware images (*.srec and *.bin) to the OTA server or directly to the OTA client is integrated into the Test Tool for Connectivity Products. Ensure that you use the latest Test Tool release available because it is updated for this Thread release.
Users can configure the *Image Type, File Version, Min* and *Max HW Version, Sector Bitmap and Signature type (CRC)* as shown below:

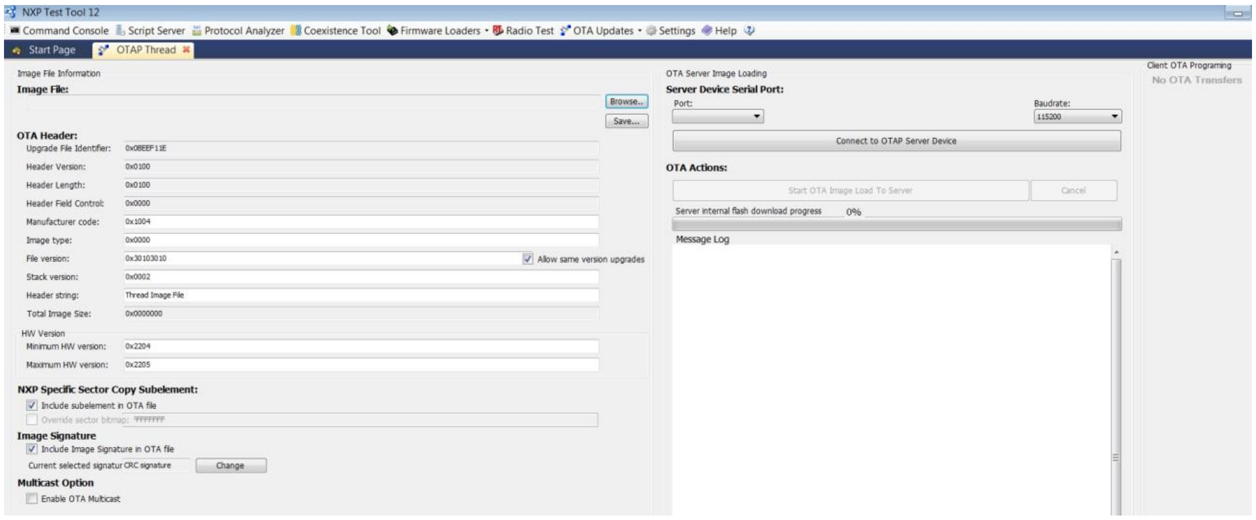


Image File Information
Image File:

OTA Header:

Upgrade File Identifier:	0x0BEEF11E
Header Version:	0x0100
Header Length:	0x0100
Header Field Control:	0x0000
Manufacturer code:	0x1004
Image type:	0x0000
File version:	0x30103010
Stack version:	0x0002
Header string:	Thread Image File
Total Image Size:	0x0000000

[✓] Allow same version upgrades

HW Version

Minimum HW version:	0x2204
Maximum HW version:	0x2205

NXP Specific Sector Copy Subelement:
[✓] Include subelement in OTA file
[ ] Override sector bitmap: FFFFFFF

Image Signature
[✓] Include Image Signature in OTA file
Current selected signatur CRC signature

Multicast Option
[ ] Enable OTA Multicast

OTA Server Image Loading
Server Device Serial Port:
Port:
Baudrate: 115200

OTA Actions:
Server internal flash download progress 0%

Client OTA Programing

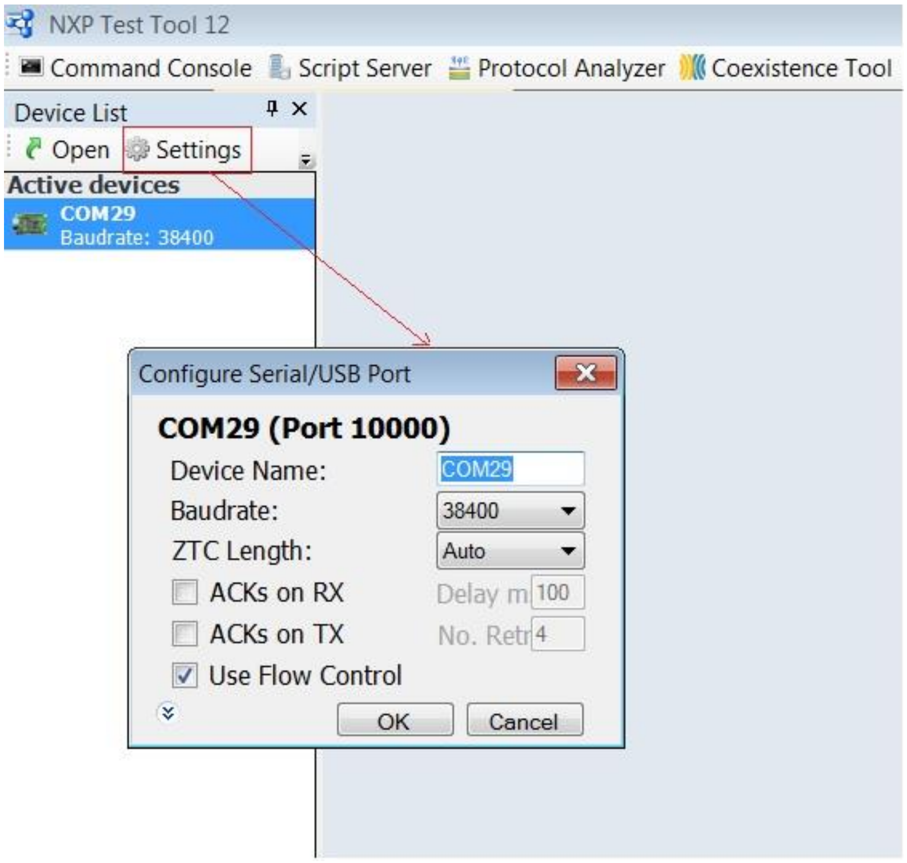
Figure 6. Test Tool overview

To start downloading the new image to the server, select the COM port on which the server board is connected, enter the baud rate (usually 115200 bps), and then press the *Connect to OTA Server Device*.
The transfer can be started by pressing S*tart OTA Image Load To Server* button.

5 Running a unicast OTA scenario

To create OTA enabled applications and perform an OTA image upgrade from a THCI based server to an OTA client perform the following steps:

1. Select the demo applications from C:\NXP\MKW24D_ConnSw_1.0.1\boards\frdmkw24\wireless_examples\thread :
   a. OTA Server: hcd_ota_server
   b. OTA Client: reed_ota_client
2. Validate configurations:
   a. On the server side. Open *config.h* file in *source* group in IAR workspace and make sure the following defines are set accordingly:
      i. #define gEnableOTAServer_d 1
      ii. #define THR_SERIAL_TUN_ROUTER 0
   b. On the client side. Open *config.h* file in *source* group in IAR workspace and make sure the following defines are set accordingly:
      i. #define gEnableOTAClient_d 1
      ii. #define gEepromType_d gEepromDevice_AT45DB161E_c
      iii. OTA client project options (Figure 5)
3. Load the *hcd_ota_server* and project onto one of the boards and make note of its COM port which will be used by the Test Tool in the steps below. Then, load the *reed_ota_client* project onto one or more of the other boards.
4. Use the Test Tool application to communicate with the OTA server board via the THCI interface. Create a new Thread network, start the commissioner, and enable joiners (the client nodes) to start joining the network.
   a. If a THCI_UART_ENABLE is selected, use UART or OpenSDA connection.
   b. Connect the OTA server board to the Test Tool application. First, click on the Command Console button to open the Command Console tab. Then, select the COM port that the board enumerated as and open up the Settings dialog box. Set them to the values shown in the Figure below. Once set, click "Open", and for the "Loaded Command Set" drop down, select ThreadIP.xml to get access to the available Thread THCI commands.



COM29 (Port 10000)

Device Name: COM29

Baudrate: 38400

ZTC Length: Auto

☐ ACKs on RX — Delay m 100

☐ ACKs on TX — No. Retr 4

☑ Use Flow Control

Figure 7. Test Tool Command Console

c. Use *THR_CreateNwk.Request* command to create a new network (Instance ID parameter set to 0x00).

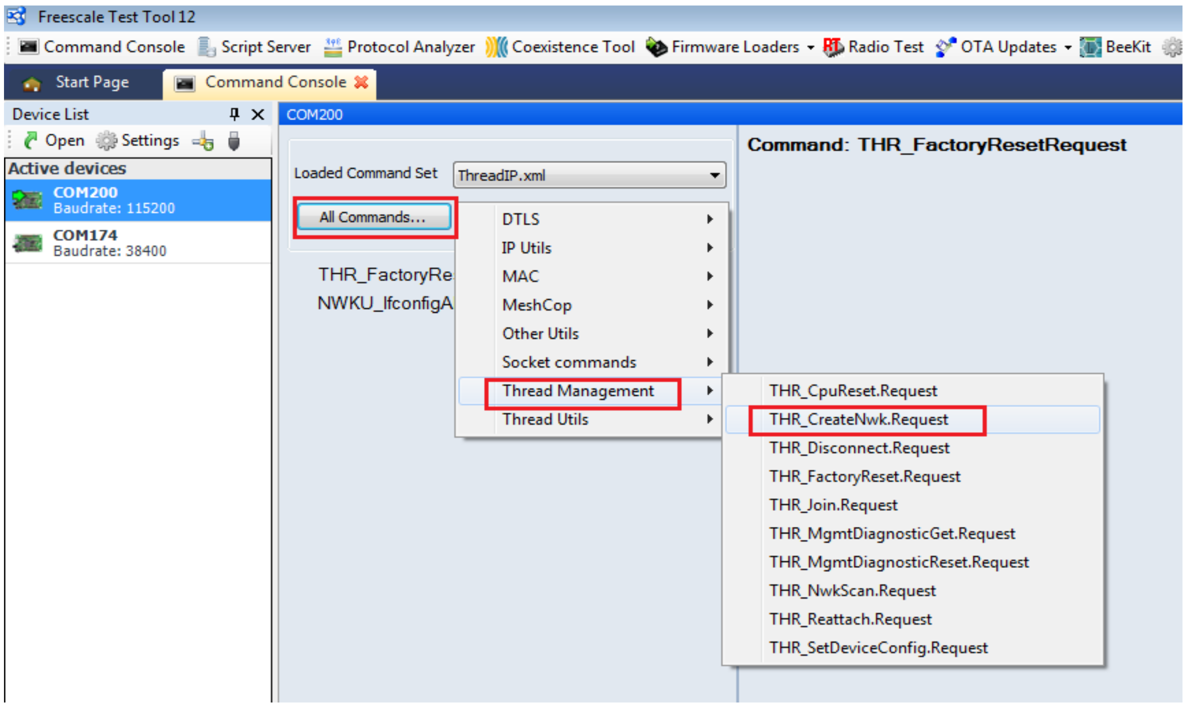
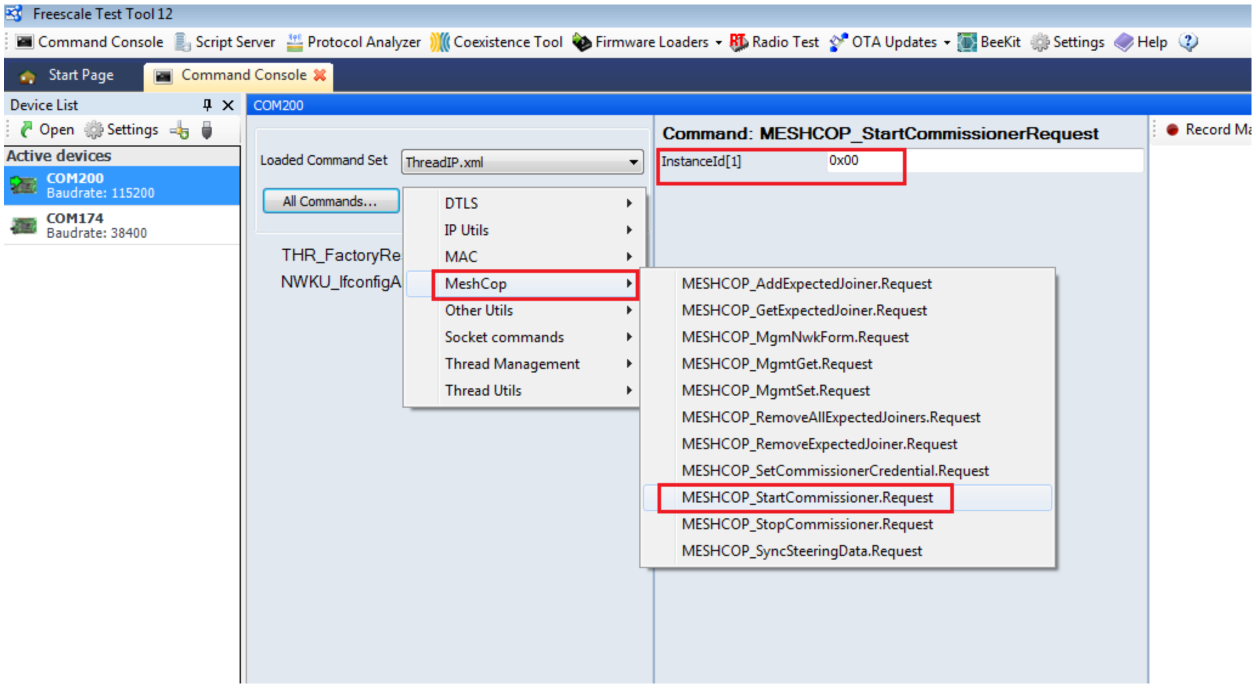


Figure 8. THR_CreateNwk.Request

d. Use *MESHCOP_StartCommissioner.Request* command to start the commissioner:



InstanceId[1] 0x00

Figure 9. MESHCOP_StartCommissioner.Request

e. Add expected joiner and synchronize steering data. In this scenario we will accept any EUI64 address.
Use the following commands with the associated parameters.

MESHCOP_AddExpectedJoiner.Request:

- InstanceId = 0x00
- Selected = TRUE
- EuiType = 0x01 (LongEUI)
- LongEUI = 0xFFFFFFFFFFFFFFFF (All FF's mean a joiner with any EUI may join this network)
- PSKdSize = 0x7
- PSKd = kinetis (the same used by OTA Client - default configuration -)

MESHCOP_SyncSteeringDataRequest:

- InstanceId = 0x00
- EuiMask = 0x01 (AllFFs)

5. After the server node has started a Thread network and a commissioner, the OTA client nodes can join the network by pressing any switch on the board. OTA client has the Shell enabled by default:

```
#define THREAD_USE_SHELL        1
```

6. Start the OTA procedure:

a. Close the Command Console tab in Test Tool.
b. From TestTool choose *OTA Updates* -> *OTAP Thread* to launch the OTA Update View, as shown in the Figure below.

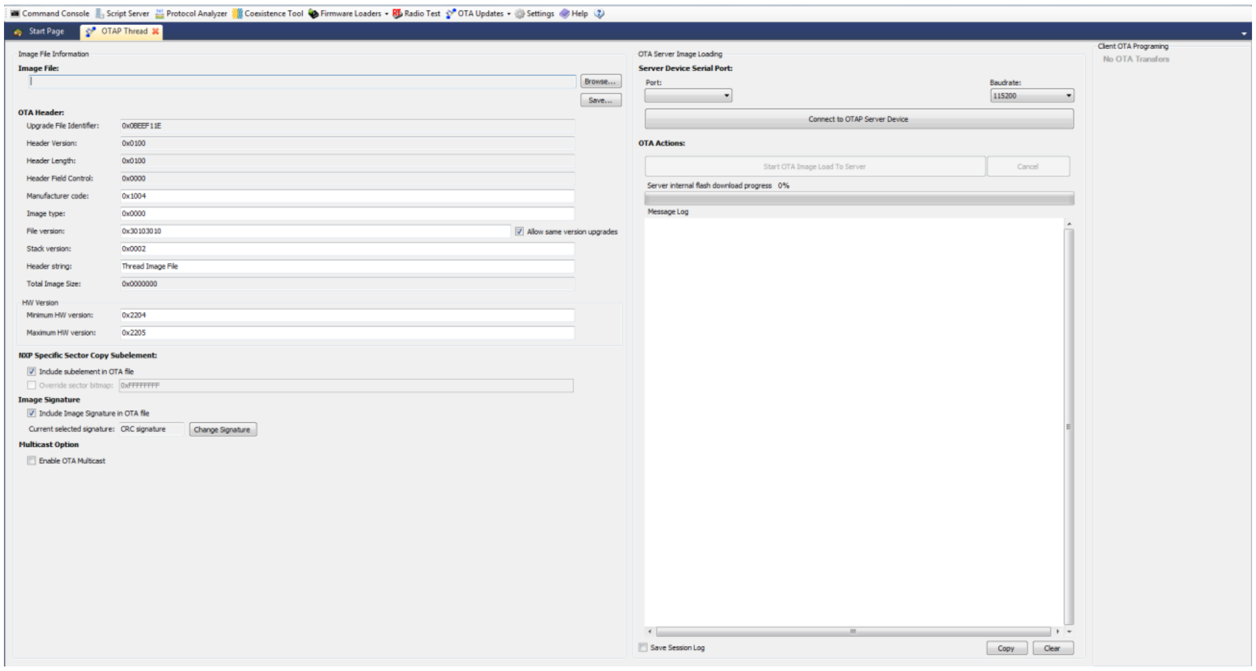


Image File Information

Image File:

OTA Header:

Upgrade File Identifier:	0x0BEEF11E
Header Version:	0x0100
Header Length:	0x0100
Header Field Control:	0x0000
Manufacturer code:	0x1004
Image type:	0x0000
File version:	0x30103010
Stack version:	0x0002
Header string:	Thread Image File
Total Image Size:	0x0000000

[✓] Allow same version upgrades

HW Version

Minimum HW version:	0x2204
Maximum HW version:	0x2205

NXP Specific Sector Copy Subelement:

[✓] Include subelement in OTA file

[ ] Override sector bitmap: 0xFFFFFFFF

Image Signature

[✓] Include Image Signature in OTA file

Current selected signature: CRC signature

Multicast Option

[ ] Enable OTA Multicast

OTA Server Image Loading

Server Device Serial Port:

Port:

Baudrate: 115200

OTA Actions:

Server internal flash download progress 0%

Message Log

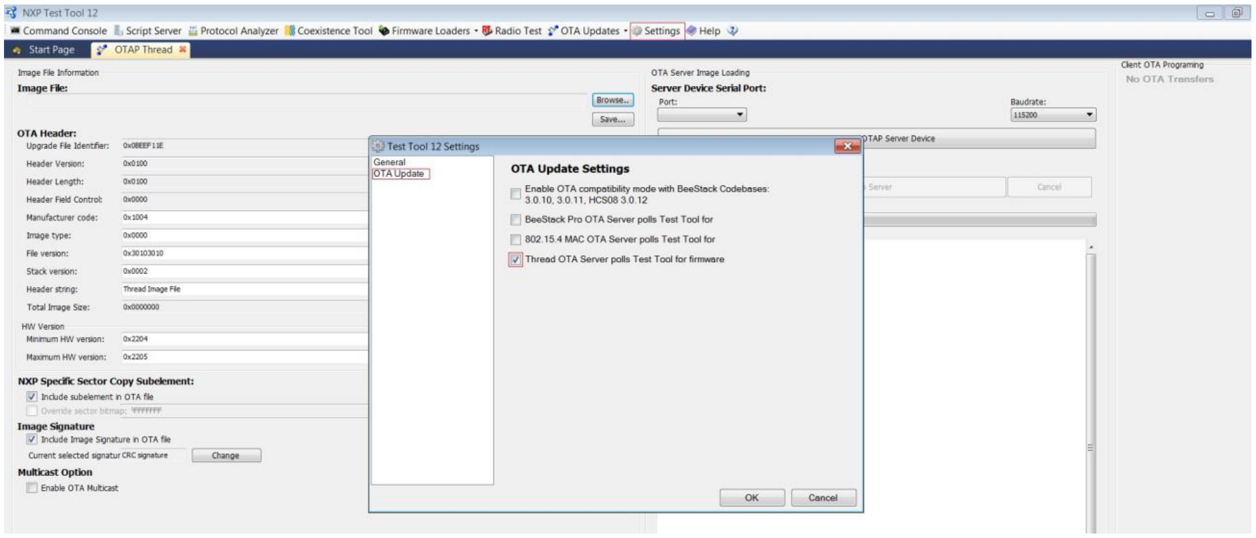
[ ] Save Session Log

Client OTA Programing

No OTA Transfers

Figure 10. Thread OTAP view

c. In the OTA process the image file can be either i) stored on the OTA server in the external flash memory, called stand-alone mode, or ii) held by Test Tool with the OTA server polling for each chunk when it is required by the client, called dongle mode. By default, the Test Tool inquires the OTA Server about the extended memory support and adjusts the image storage location in accordance with it. The user may choose to keep the image in the Test Tool, regardless of the OTA server configuration, by selecting the “Thread OTA Server polls Test Tool for firmware file fragments” checkbox as displayed in the Figure below.



Test Tool 12 Settings

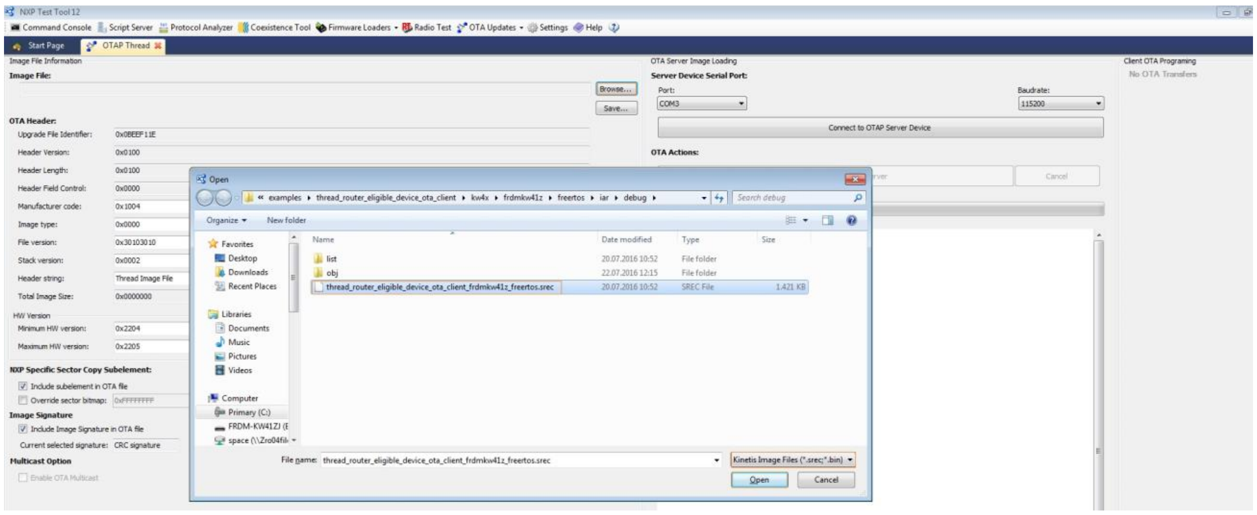
OTA Update Settings

- [ ] Enable OTA compatibility mode with BeeStack Codebases: 3.0.10, 3.0.11, HCS08 3.0.12
- [ ] BeeStack Pro OTA Server polls Test Tool for
- [ ] 802.15.4 MAC OTA Server polls Test Tool for
- [✓] Thread OTA Server polls Test Tool for firmware

Figure 11. OTA Update settings

d. In the OTA Update View in the Image File Information area, click Browse… and navigate to the binary folder.

e. Select Kinetis Image Files (*.srec, *.bin) in the *Files of type* drop down in the Open Window as shown in the Figure below.



Open

« examples ▸ thread_router_eligible_device_ota_client ▸ kw4x ▸ frdmkw41z ▸ freertos ▸ iar ▸ debug ▸

Name	Date modified	Type	Size
list	20.07.2016 10:52	File folder	
obj	22.07.2016 12:15	File folder	
thread_router_eligible_device_ota_client_frdmkw41z_freertos.srec	20.07.2016 10:52	SREC File	1,421 KB

File name: thread_router_eligible_device_ota_client_frdmkw41z_freertos.srec

Kinetis Image Files (*.srec;*.bin)

Figure 12. Selecting S-Record file type

NOTE

The new OTA client image should have the same format as the original image, including the bootloader configuration presented in Figure 5.

f. Select the *reed_ota_client.srec* file that was compiled previously for the client nodes. You may want to modify it slightly and recompile it to verify that the client nodes got updated using the OTA firmware update. The OTA headers in the

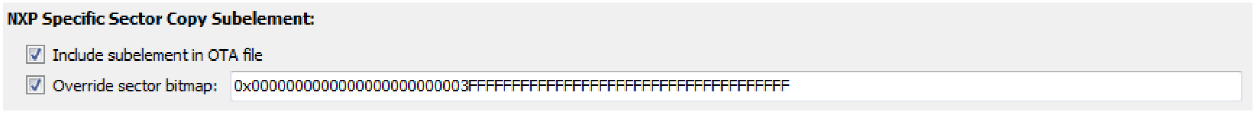
OTA Update View for the file are filled in automatically. Configure the *Image Type, File Version, Sector Bitmap* and *Signature* (only CRC signature supported at this moment).

g. Select the processor type of the OTA client. By default, the NVM check box is selected, which means that the NVM will be preserved.

Note

By checking the "Image contains bootloader" checkbox, the user informs the test tool that the selected image contains a bootloader. For .bin files, this information is passed to the Test Tool by the user. For .srec files, the Test Tool is able to obtain this information automatically. Starting with Test Tool 12.5.4, the bootloader is no longer sent over the air during the OTA process, thus the OTA process duration has been decreased.

h. In a second level of configuration, the user can explicitly define which internal memory sectors to erase by selecting the “Override sector bitmap” checkbox, as shown in the Figure below. For more information, see the Connectivity Framework Reference Manual chapter 3.20.2. OTAP Bootloader.



NXP Specific Sector Copy Subelement:

[✓] Include subelement in OTA file

[✓] Override sector bitmap: 0x000000000000000000000003FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Figure 13. Override sector bitmap

i. Click *Start Over the Air Programming*. This uses the *hcd_ota_server* to initiate the OTA process by informing the clients that a new image is available, using a multicast *ImageNotify* command and in the same time to start pushing the *reed_ota_client.srec* to the OTA Client application and displaying the progress as shown in the Figure below.

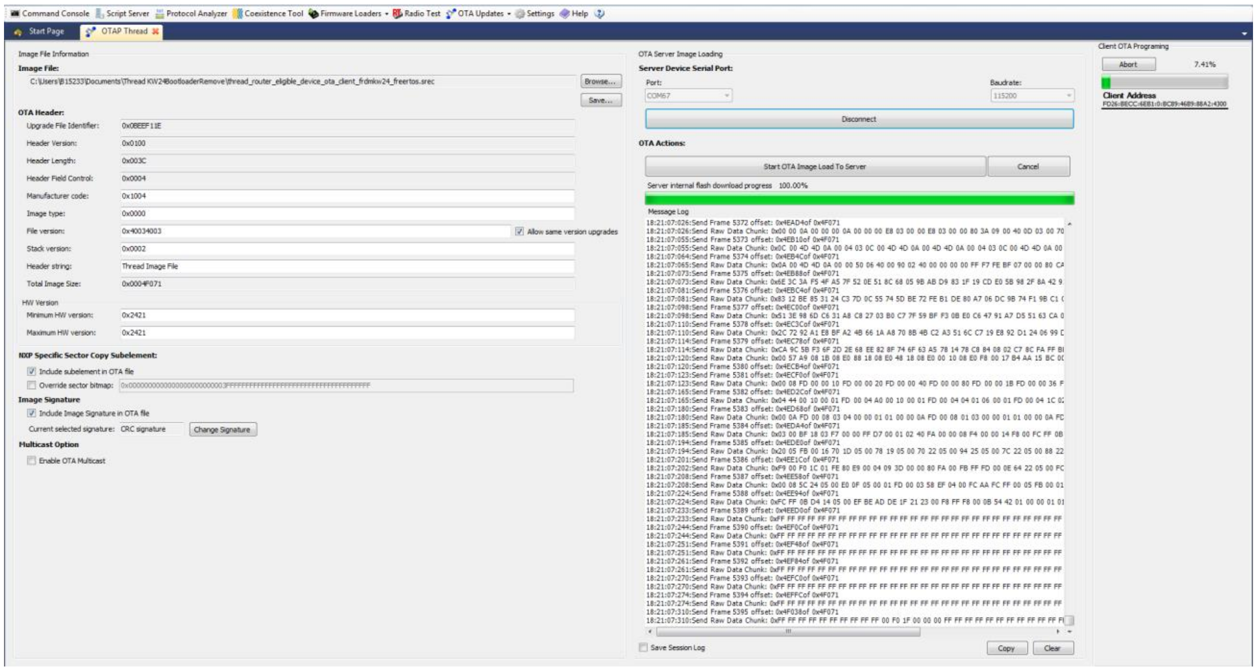


Image File Information

Image File:

C:\Users\B15233\Documents\Thread KW24BootloaderRemove\thread_router_eligible_device_ota_client_frdmkw24_freertos.srec

OTA Header:

Upgrade File Identifier:	0x0BEEF11E
Header Version:	0x0100
Header Length:	0x003C
Header Field Control:	0x0004
Manufacturer code:	0x1004
Image type:	0x0000
File version:	0x40034003
Stack version:	0x0002
Header string:	Thread Image File
Total Image Size:	0x0004F071

[✓] Allow same version upgrades

HW Version

Minimum HW version:	0x2421
Maximum HW version:	0x2421

NXP Specific Sector Copy Subelement:

[✓] Include subelement in OTA file

[ ] Override sector bitmap: 0x000000000000000000000003FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Image Signature

[✓] Include Image Signature in OTA file

Current selected signature: CRC signature

Multicast Option

[ ] Enable OTA Multicast

OTA Server Image Loading

Server Device Serial Port:

Port: COM67

Baudrate: 115200

OTA Actions:

Server internal flash download progress 100.00%

Message Log

```
18:21:07:026:Send Frame 5372 offset: 0x4EAD4of 0x4F071
18:21:07:026:Send Raw Data Chunk: 0x00 00 0A 00 00 00 0A 00 00 00 E8 03 00 00 E8 03 00 00 80 3A 09 00 40 0D 03 00 70
18:21:07:055:Send Frame 5373 offset: 0x4EB10of 0x4F071
18:21:07:055:Send Raw Data Chunk: 0x0C 00 4D 4D 0A 00 04 03 0C 00 4D 4D 0A 00 4D 4D 0A 00 04 03 0C 00 4D 4D 0A 00
18:21:07:064:Send Frame 5374 offset: 0x4EB4Cof 0x4F071
18:21:07:065:Send Raw Data Chunk: 0x0A 00 4D 4D 0A 00 00 50 06 40 00 90 02 40 00 00 00 00 FF F7 FE BF 07 00 00 80 CA
18:21:07:073:Send Frame 5375 offset: 0x4EB88of 0x4F071
18:21:07:073:Send Raw Data Chunk: 0x6E 3C 3A F5 4F A5 7F 52 0E 51 8C 68 05 9B AB D9 83 1F 19 CD E0 5B 98 2F 8A 42 9
18:21:07:081:Send Frame 5376 offset: 0x4EBC4of 0x4F071
18:21:07:081:Send Raw Data Chunk: 0x83 12 BE 85 31 24 C3 7D 0C 55 74 5D BE 72 FE B1 DE 80 A7 06 DC 9B 74 F1 9B C1 (
18:21:07:098:Send Frame 5377 offset: 0x4EC00of 0x4F071
18:21:07:098:Send Raw Data Chunk: 0x51 3E 98 6D C6 31 A8 C8 27 03 B0 C7 7F 59 BF F3 0B E0 C6 47 91 A7 D5 51 63 CA 0
18:21:07:110:Send Frame 5378 offset: 0x4EC3Cof 0x4F071
18:21:07:110:Send Raw Data Chunk: 0x2C 72 92 A1 E8 BF A2 4B 66 1A A8 70 8B 4B C2 A3 51 6C C7 19 E8 92 D1 24 06 99 D
18:21:07:114:Send Frame 5379 offset: 0x4EC78of 0x4F071
18:21:07:114:Send Raw Data Chunk: 0xCA 9C 5B F3 6F 2D 2E 68 EE 82 8F 74 6F 63 A5 78 14 78 C8 84 08 02 C7 8C FA FF BI
18:21:07:120:Send Raw Data Chunk: 0x00 57 A9 08 1B 08 E0 88 18 08 E0 48 18 08 E0 00 10 08 E0 F8 00 17 B4 AA 15 BC 0C
18:21:07:120:Send Frame 5380 offset: 0x4ECB4of 0x4F071
18:21:07:123:Send Frame 5381 offset: 0x4ECF0of 0x4F071
18:21:07:123:Send Raw Data Chunk: 0x00 08 FD 00 00 10 FD 00 00 20 FD 00 00 40 FD 00 00 80 FD 00 00 1B FD 00 00 36 F
18:21:07:165:Send Frame 5382 offset: 0x4ED2Cof 0x4F071
18:21:07:165:Send Raw Data Chunk: 0x04 44 00 10 00 01 FD 00 04 A0 00 10 00 01 FD 00 04 04 01 06 00 01 FD 00 04 1C 02
18:21:07:180:Send Frame 5383 offset: 0x4ED68of 0x4F071
18:21:07:180:Send Raw Data Chunk: 0x00 0A FD 00 08 03 04 00 00 01 01 00 00 0A FD 00 08 01 03 00 00 01 01 00 00 0A FC
18:21:07:185:Send Frame 5384 offset: 0x4EDA4of 0x4F071
18:21:07:185:Send Raw Data Chunk: 0x03 00 BF 18 03 F7 00 00 FF D7 00 01 02 40 FA 00 00 08 F4 00 00 14 F8 00 FC FF 0B
18:21:07:194:Send Frame 5385 offset: 0x4EDE0of 0x4F071
18:21:07:194:Send Raw Data Chunk: 0x20 05 FB 00 16 70 1D 05 00 78 19 05 00 70 22 05 00 94 25 05 00 7C 22 05 00 88 22
18:21:07:201:Send Frame 5386 offset: 0x4EE1Cof 0x4F071
18:21:07:202:Send Raw Data Chunk: 0xF9 00 F0 1C 01 FE 80 E9 00 04 09 3D 00 00 80 FA 00 FB FF FD 00 0E 64 22 05 00 FC
18:21:07:208:Send Frame 5387 offset: 0x4EE58of 0x4F071
18:21:07:208:Send Raw Data Chunk: 0x00 08 5C 24 05 00 E0 0F 05 00 01 FD 00 03 58 EF 04 00 FC AA FC FF 00 05 FB 00 01
18:21:07:224:Send Frame 5388 offset: 0x4EE94of 0x4F071
18:21:07:224:Send Raw Data Chunk: 0xFC FF 0B D4 14 05 00 EF BE AD DE 1F 21 23 00 F8 FF F8 00 0B 54 42 01 00 00 01 01
18:21:07:233:Send Frame 5389 offset: 0x4EED0of 0x4F071
18:21:07:233:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:21:07:244:Send Frame 5390 offset: 0x4EF0Cof 0x4F071
18:21:07:244:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:21:07:251:Send Frame 5391 offset: 0x4EF48of 0x4F071
18:21:07:251:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:21:07:261:Send Frame 5392 offset: 0x4EF84of 0x4F071
18:21:07:261:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:21:07:270:Send Frame 5393 offset: 0x4EFC0of 0x4F071
18:21:07:270:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:21:07:274:Send Frame 5394 offset: 0x4EFFCof 0x4F071
18:21:07:274:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:21:07:310:Send Frame 5395 offset: 0x4F038of 0x4F071
18:21:07:310:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF 00 F0 1F 00 00 00 FF FF FF FF FF FF FF FF FF FF FI
```

[ ] Save Session Log

Client OTA Programing

7.41%

Client Address

FD26:BECC:6EB1:0:BC89:4689:88A2:4300

Figure 14. Running OTA procedure

j. Wait until the process reaches 100%. The client board resets itself when it has received the full image. The bootloader on the client board then runs and reprograms the internal flash of the client node. The client node reboots when the programming is finished and starts running the new image.

6 Running a multicast OTA scenario

Note

The Multicast OTA feature is available as an add-on package.
Contact your NXP sales representative for more information.

To run a multicast OTA scenario follow the same steps as for a unicast scenario. Select the “Enable OTA multicast” checkbox before starting the over the air programing, as shown in the Figure below.

Wait until the process reaches 100%. The client board resets itself when it has received the full image. The bootloader on the client board then runs and reprograms the internal flash of the client node. The client node reboots when the programming is finished and starts running the new image.

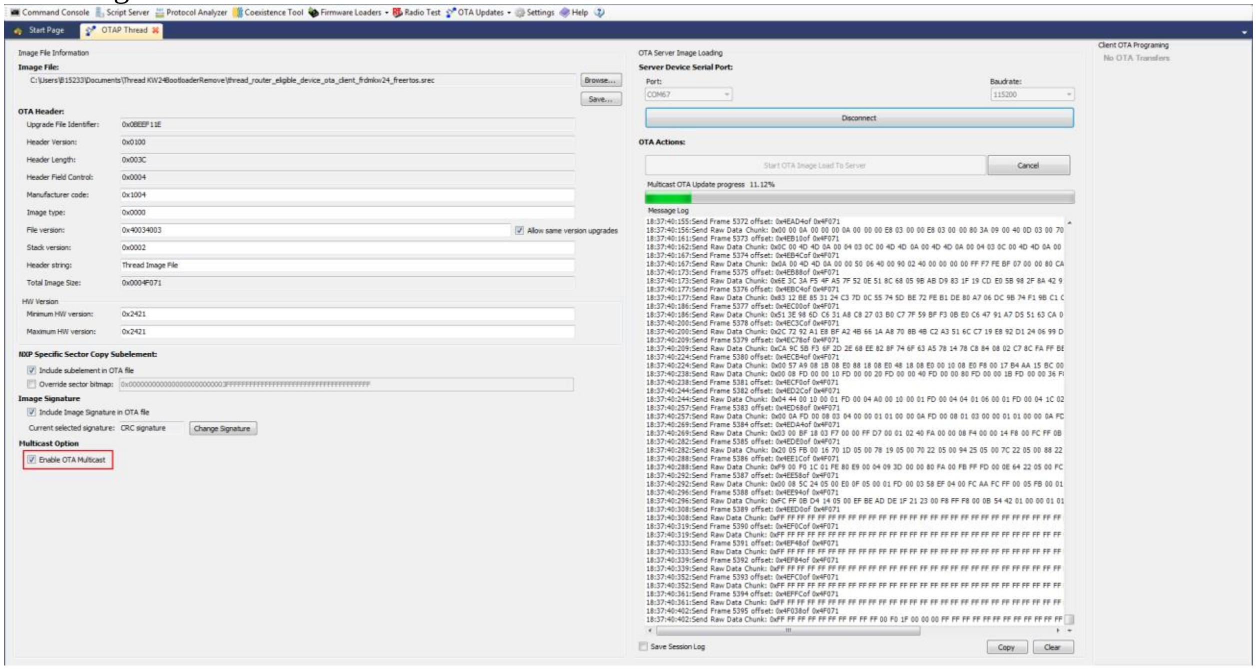


Image File Information

Image File: C:\Users\B15233\Documents\Thread KW24BootloaderRemove\thread_router_eligible_device_ota_client_frdmkw24_freertos.srec

OTA Header:

Upgrade File Identifier:	0x0BEEF11E
Header Version:	0x0100
Header Length:	0x003C
Header Field Control:	0x0004
Manufacturer code:	0x1004
Image type:	0x0000
File version:	0x40034003
Stack version:	0x0002
Header string:	Thread Image File
Total Image Size:	0x0004F071

[✓] Allow same version upgrades

HW Version

Minimum HW version:	0x2421
Maximum HW version:	0x2421

NXP Specific Sector Copy Subelement:

[✓] Include subelement in OTA file

[ ] Override sector bitmap: 0x000000000000000000000003FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Image Signature

[✓] Include Image Signature in OTA file

Current selected signature: CRC signature

Multicast Option

[✓] Enable OTA Multicast

OTA Server Image Loading

Server Device Serial Port:

Port: COM67

Baudrate: 115200

OTA Actions:

Multicast OTA Update progress 11.12%

Message Log

```
18:37:40:155:Send Frame 5372 offset: 0x4EAD4of 0x4F071
18:37:40:156:Send Raw Data Chunk: 0x00 00 0A 00 00 00 0A 00 00 00 E8 03 00 00 E8 03 00 00 80 3A 09 00 40 0D 03 00 70
18:37:40:161:Send Frame 5373 offset: 0x4EB10of 0x4F071
18:37:40:162:Send Raw Data Chunk: 0x0C 00 4D 4D 0A 00 04 03 0C 00 4D 4D 0A 00 4D 4D 0A 00 04 03 0C 00 4D 4D 0A 00
18:37:40:167:Send Frame 5374 offset: 0x4EB4Cof 0x4F071
18:37:40:167:Send Raw Data Chunk: 0x0A 00 4D 4D 0A 00 00 50 06 40 00 90 02 40 00 00 00 00 FF F7 FE BF 07 00 00 80 CA
18:37:40:173:Send Frame 5375 offset: 0x4EB88of 0x4F071
18:37:40:173:Send Raw Data Chunk: 0x6E 3C 3A F5 4F A5 7F 52 0E 51 8C 68 05 9B AB D9 83 1F 19 CD E0 5B 98 2F 8A 42 9
18:37:40:177:Send Frame 5376 offset: 0x4EBC4of 0x4F071
18:37:40:177:Send Raw Data Chunk: 0x83 12 BE 85 31 24 C3 7D 0C 55 74 5D BE 72 FE B1 DE 80 A7 06 DC 9B 74 F1 9B C1 C
18:37:40:186:Send Frame 5377 offset: 0x4EC00of 0x4F071
18:37:40:186:Send Raw Data Chunk: 0x51 3E 98 6D C6 31 A8 C8 27 03 B0 C7 7F 59 BF F3 0B E0 C6 47 91 A7 D5 51 63 CA 0
18:37:40:200:Send Frame 5378 offset: 0x4EC3Cof 0x4F071
18:37:40:200:Send Raw Data Chunk: 0x2C 72 92 A1 E8 BF A2 4B 66 1A A8 70 8B 4B C2 A3 51 6C C7 19 E8 92 D1 24 06 99 D
18:37:40:209:Send Frame 5379 offset: 0x4EC78of 0x4F071
18:37:40:209:Send Raw Data Chunk: 0xCA 9C 5B F3 6F 2D 2E 68 EE 82 8F 74 6F 63 A5 78 14 78 C8 84 08 02 C7 8C FA FF BE
18:37:40:224:Send Frame 5380 offset: 0x4ECB4of 0x4F071
18:37:40:224:Send Raw Data Chunk: 0x00 57 A9 08 1B 08 E0 88 18 08 E0 48 18 08 E0 00 10 08 E0 F8 00 17 B4 AA 15 BC 00
18:37:40:238:Send Raw Data Chunk: 0x00 08 FD 00 00 10 FD 00 00 20 FD 00 00 40 FD 00 00 80 FD 00 00 1B FD 00 00 36 F
18:37:40:238:Send Frame 5381 offset: 0x4ECF0of 0x4F071
18:37:40:244:Send Frame 5382 offset: 0x4ED2Cof 0x4F071
18:37:40:244:Send Raw Data Chunk: 0x04 44 00 10 00 01 FD 00 04 A0 00 10 00 01 FD 00 04 04 01 06 00 01 FD 00 04 1C 02
18:37:40:257:Send Frame 5383 offset: 0x4ED68of 0x4F071
18:37:40:257:Send Raw Data Chunk: 0x00 0A FD 00 08 03 04 00 00 01 01 00 00 0A FD 00 08 01 03 00 00 01 01 00 00 0A FC
18:37:40:269:Send Frame 5384 offset: 0x4EDA4of 0x4F071
18:37:40:269:Send Raw Data Chunk: 0x03 00 BF 18 03 F7 00 00 FF D7 00 01 02 40 FA 00 00 08 F4 00 00 14 F8 00 FC FF 0B
18:37:40:282:Send Frame 5385 offset: 0x4EDE0of 0x4F071
18:37:40:282:Send Raw Data Chunk: 0x20 05 FB 00 16 70 1D 05 00 78 19 05 00 70 22 05 00 94 25 05 00 7C 22 05 00 88 22
18:37:40:288:Send Frame 5386 offset: 0x4EE1Cof 0x4F071
18:37:40:288:Send Raw Data Chunk: 0xF9 00 F0 1C 01 FE 80 E9 00 04 09 3D 00 00 80 FA 00 FB FF FD 00 0E 64 22 05 00 FC
18:37:40:292:Send Frame 5387 offset: 0x4EE58of 0x4F071
18:37:40:292:Send Raw Data Chunk: 0x00 08 5C 24 05 00 E0 0F 05 00 01 FD 00 03 58 EF 04 00 FC AA FC FF 00 05 FB 00 01
18:37:40:296:Send Frame 5388 offset: 0x4EE94of 0x4F071
18:37:40:296:Send Raw Data Chunk: 0xFC FF 0B D4 14 05 00 EF BE AD DE 1F 21 23 00 F8 FF F8 00 0B 54 42 01 00 00 01 01
18:37:40:308:Send Frame 5389 offset: 0x4EED0of 0x4F071
18:37:40:308:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:37:40:319:Send Frame 5390 offset: 0x4EF0Cof 0x4F071
18:37:40:319:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:37:40:333:Send Frame 5391 offset: 0x4EF48of 0x4F071
18:37:40:333:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:37:40:339:Send Frame 5392 offset: 0x4EF84of 0x4F071
18:37:40:339:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:37:40:352:Send Frame 5393 offset: 0x4EFC0of 0x4F071
18:37:40:352:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:37:40:361:Send Frame 5394 offset: 0x4EFFCof 0x4F071
18:37:40:361:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
18:37:40:402:Send Frame 5395 offset: 0x4F038of 0x4F071
18:37:40:402:Send Raw Data Chunk: 0xFF FF FF FF FF FF FF FF FF FF 00 F0 1F 00 00 00 FF FF FF FF FF FF FF FF FF FF FF FF
```

[ ] Save Session Log

Client OTA Programing

No OTA Transfers

Figure 15. Running OTA procedure multicast

7 Revision History

This table summarizes revisions to this document.

Table 10 Revision history

Revision number	Date	Substantive changes
0	10/2015	Initial release
1	03/2016	Removed ambiguous references to FSCI length field
2	06/2016	Updates related to the Thread OTA Multicast implementation and its corresponding Test Tool updates
3	08/2016	Updates for Thread KW41 Beta Release
4	09/2016	Updates for Thread KW41 GA Release
5	12/2016	Updates for Thread KW24D GA Release

Document number: KTSOTAFUUG
Rev. 5
12/2016