NXP Semiconductors
User's Guide

Document Number: KTSOTAFUUG
Rev 7, 01/2018

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

Contents

Chapter 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.The transfer of an OTA image from the server to the client is done by unicast or multicast messages through the Thread network.

Chapter 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.

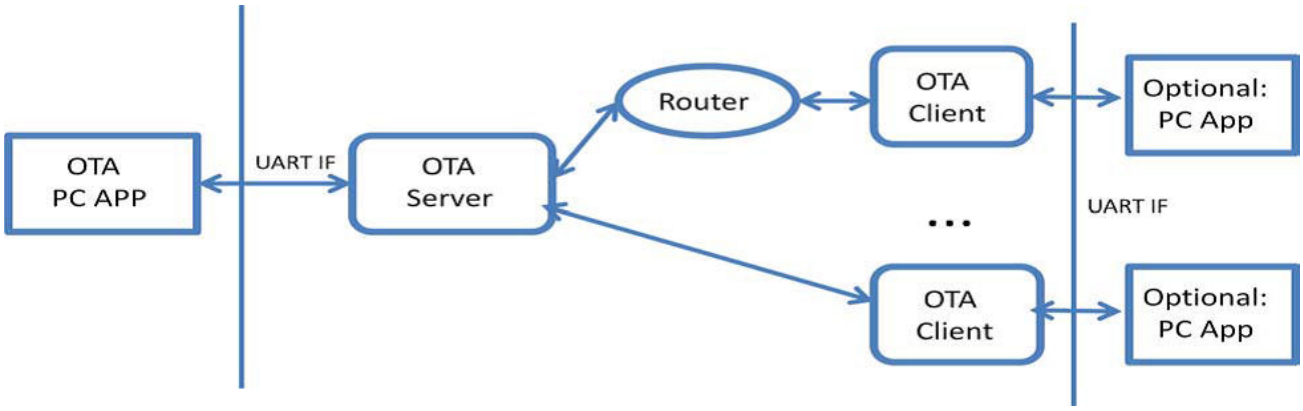

Figure 1. OTA Block Diagram

The download of an OTA image from server to client requires an application bootloader installed and sufficient memory (external or internal) to store the recently 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

2.2.2 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

Table continues on the next page...

Table 5. OTA Block Request payload (continued)

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

<table>
<tr><th>Parameter</th><th>Size</th><th colspan="2">Comments</th></tr>
<tr><td>CommandId</td><td>1</td><td colspan="2">Command Identifier:
gOtaCmd_ BlockRsp_c = 0x04</td></tr>
<tr><td>Status</td><td>1</td><td colspan="2">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</td></tr>
<tr><td rowspan="5">Data</td><td rowspan="5">Variable</td><td rowspan="4">If status = gOtaFileStatus_Success_c</td><td>FileVersion [4 bytes]</td></tr>
<tr><td>FileOffset [4 bytes]</td></tr>
<tr><td>DataSize [1 byte]</td></tr>
<tr><td>Data [DataSize param bytes]</td></tr>
<tr><td rowspan="2">If status != gOtaFileStatus_Success_c</td><td>CurrentTime [4 bytes]</td></tr>
<tr><td>RequestTime [4 bytes]</td></tr>
</table>

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	
Data	12	If status = gOtaFileStatus_Success_c	Current Time [4 bytes]
			Upgrade Time [4 bytes]
			File Version [4 bytes]
		If status != gOtaFileStatus_Success_c	Current Time [4 bytes]
			Request Time [4 bytes]

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. The OTA Server discovery feature is disabled if the data regarding the OTA server (like the server's short address) is contained in a Service TLV in Network Data packets.
- URI-Path: NON POST coap://<dest>/otaclient
- CoAP payload:

Table 9. OTA Server discovery CoAP payload

Parameter	Size	Comments
CommandId	1	Command Identifier: gOtaCmd_ServerDiscovery_c = 0x07
ManufacturerCode	2	Manufacturer Code: Default Value = 0x1004
ImageType	2	Image Type: Default Value = 0x0000

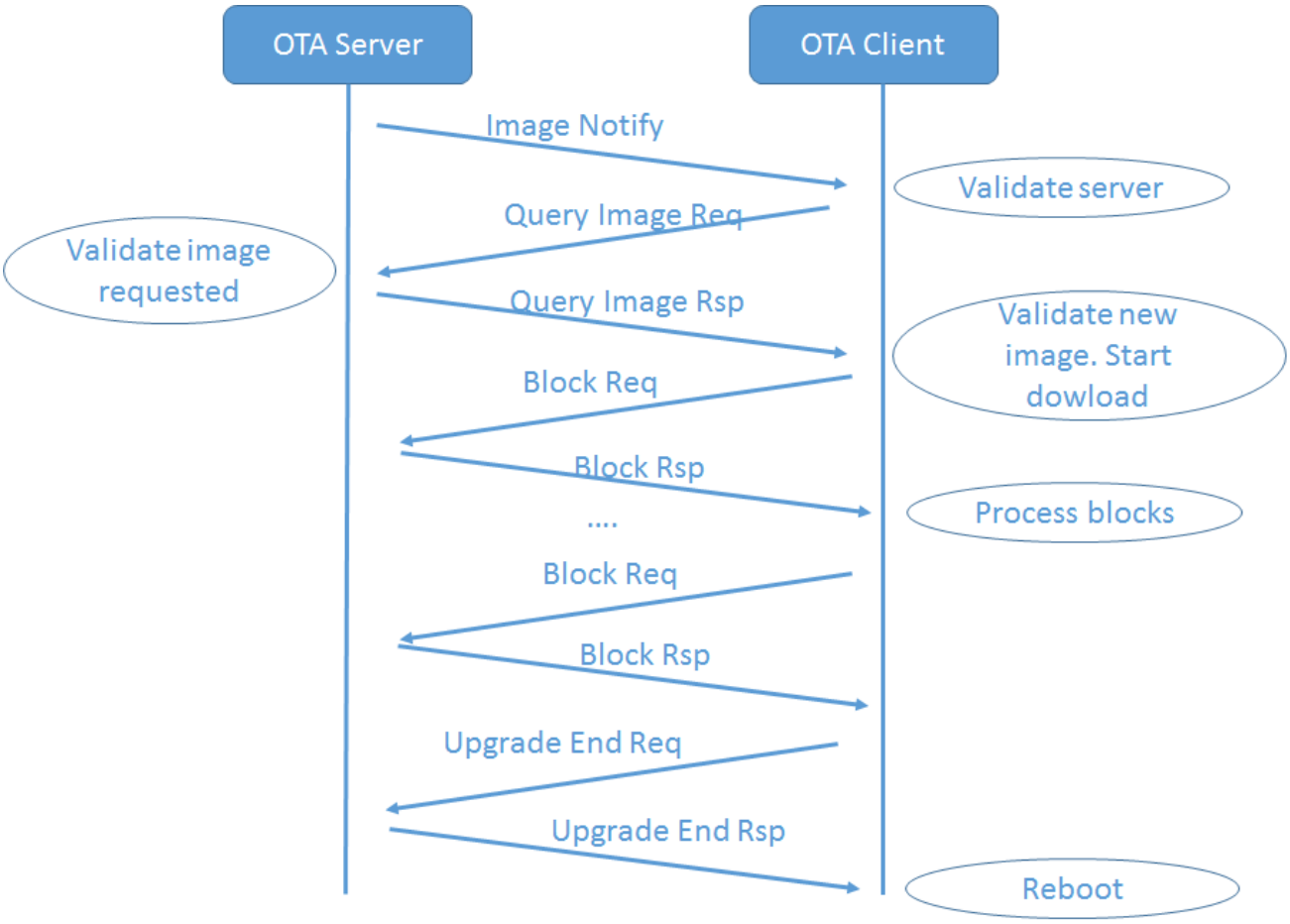

Figure 2. OTA Upgrade Diagram

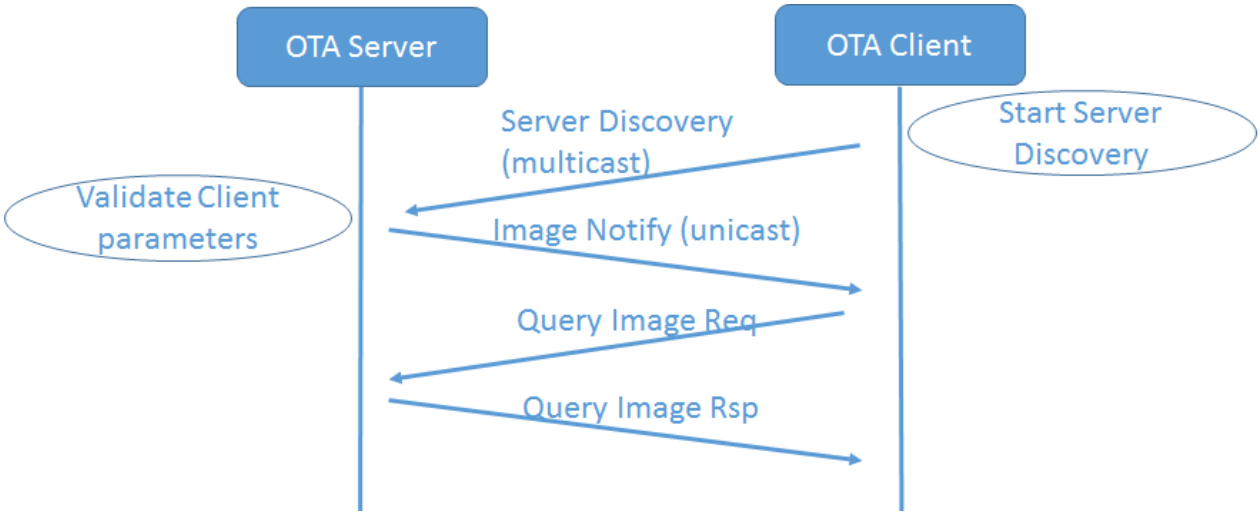

Figure 3. OTA Server Discovery Diagram

Chapter 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_ AT45DB041E_c – use this particular EEPROM device type to store the image in the external flash of the client for FRDM-KW41 boards. For other boards, check Eeprom.h;
- OTA_USE_NWK_DATA = TRUE – includes the OTA server data in a Service TLV in Network Data packets, so each board will always be aware of the server's short address.

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
- gUseInternalStorageLink_d=0 – use external flash for temporary storing the image sent over the air; otherwise uses the internal flash.
- gEraseNVMLink_d=1 – includes NVM firmware sectors in OTA file, and will overwrite those sectors when performing OTA update. When using .bin format, gEraseNVMLink_d must be set to 0 to preserve NVM data
- 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.

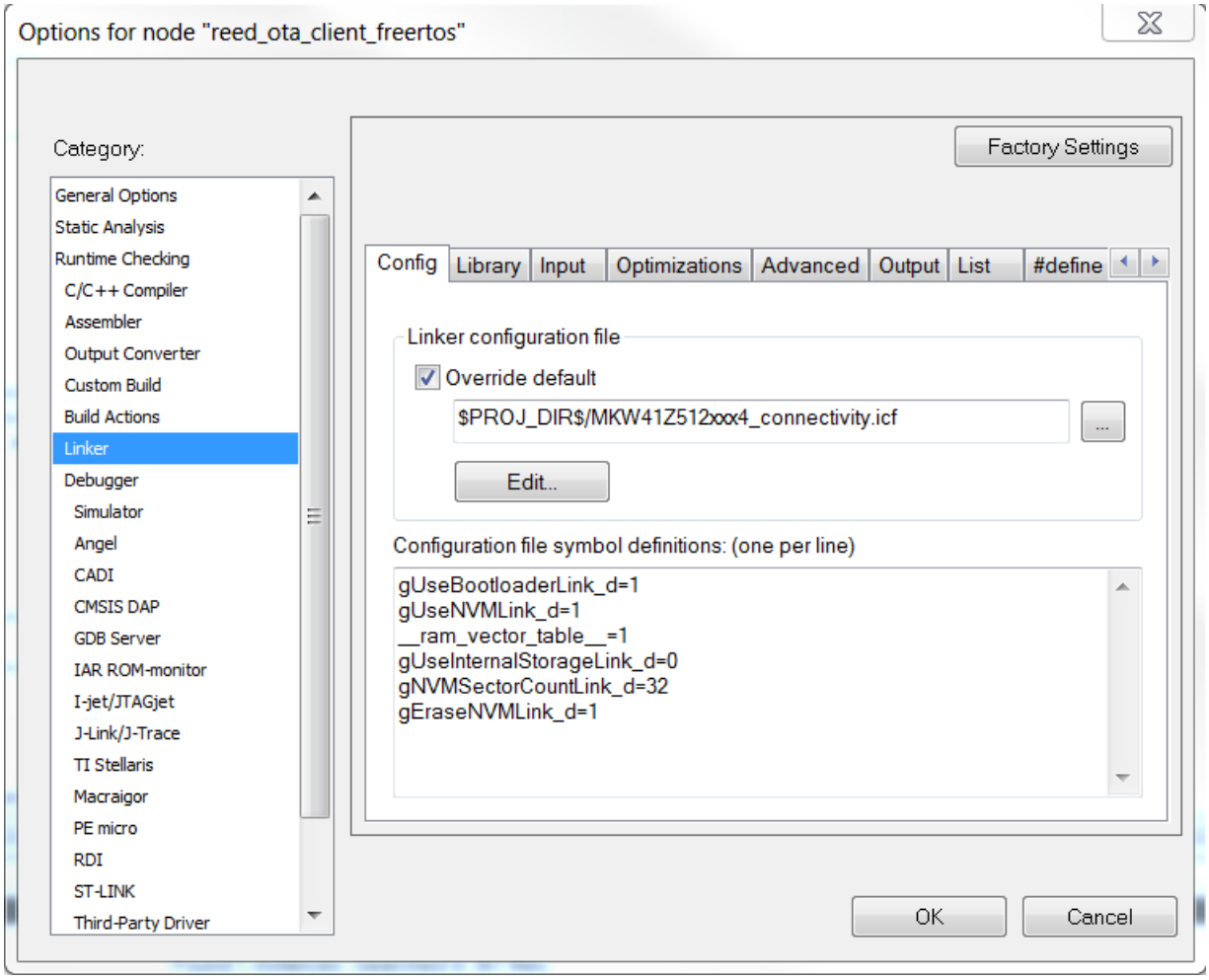

Figure 4. IAR Linker Configuration Options

In the case of MCU Xpresso IDE, the linker file for the ota client project is preconfigured with the appropriate settings.

Chapter 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:

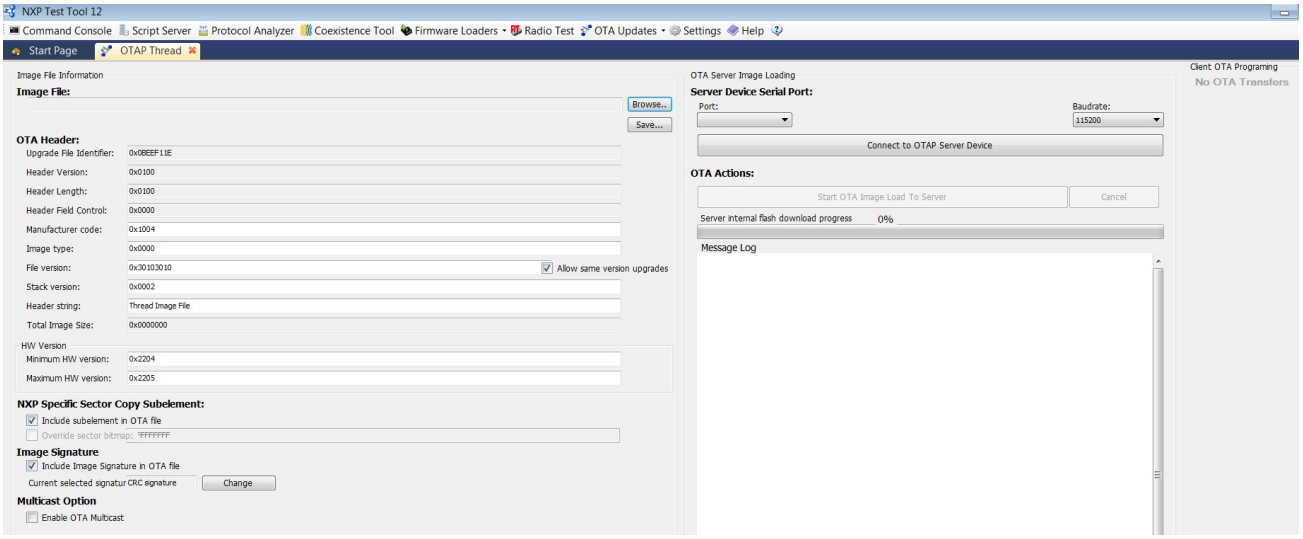

Figure 5. Test Tool overview

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.

Chapter 5
Running a unicast OTA scenario

1. Select the demo applications from <*install folder* >\boards\<*board_type* >\wireless_examples\thread :
   a. OTA Server: hcd_ota_server
   b. OTA Client:
      i. reed_ota_client (frdmkw41z boards only)
      ii. end_device_ota_client
2. Validate configurations:
   a. On the server side. Please make sure the following defines are set accordingly:
      i. In *config.h* : #define gEnableOTAServer_d 1
      ii. In app_thread_config.h: #define THR_SERIAL_TUN_ROUTER 0
   b. On the client side.
      i. Please open *config.h* make sure the following defines are set accordingly: #define gEnableOTAClient_d 1
      ii. For IAR IDE, right click project -> options -> linker -> config and add the appropriate settings as presented in Figure 4.
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. Load the OTAP Bootloader binary onto the client board. The linker files for the two project are configured so that the bootloader and wireless application will coexist on different areas of the flash. The process for writing the bootloader on the board is the following:
   a. Go to <*install_folder* >boards\frdmkw41z\wireless_examples\framework\bootloader_otap\bm\iar and open the project.
   b. Compile it and download it on to the board.
5. 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. 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.

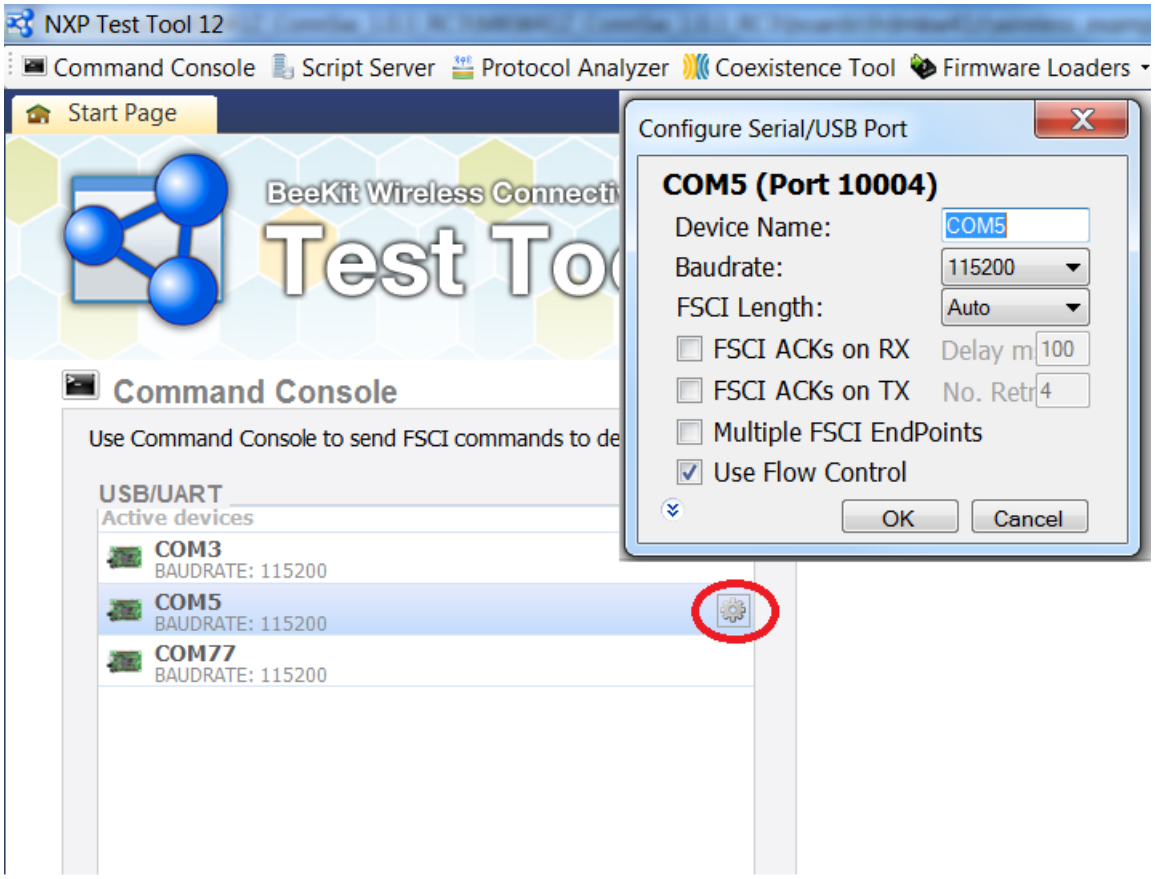

Figure 6. Test Tool Command Console

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

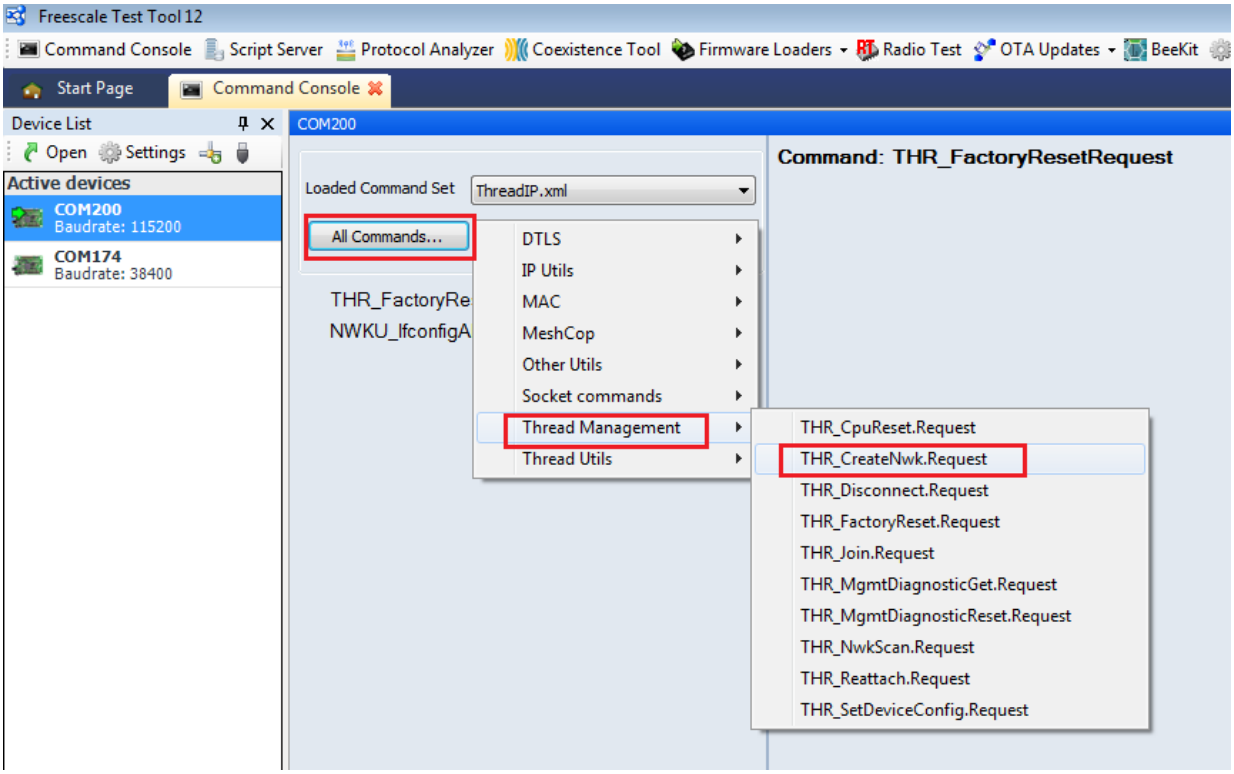

Figure 7. THR_CreateNwk.Request

c. Use *MESHCOP_StartCommissioner.Request*commandto start the commissioner:

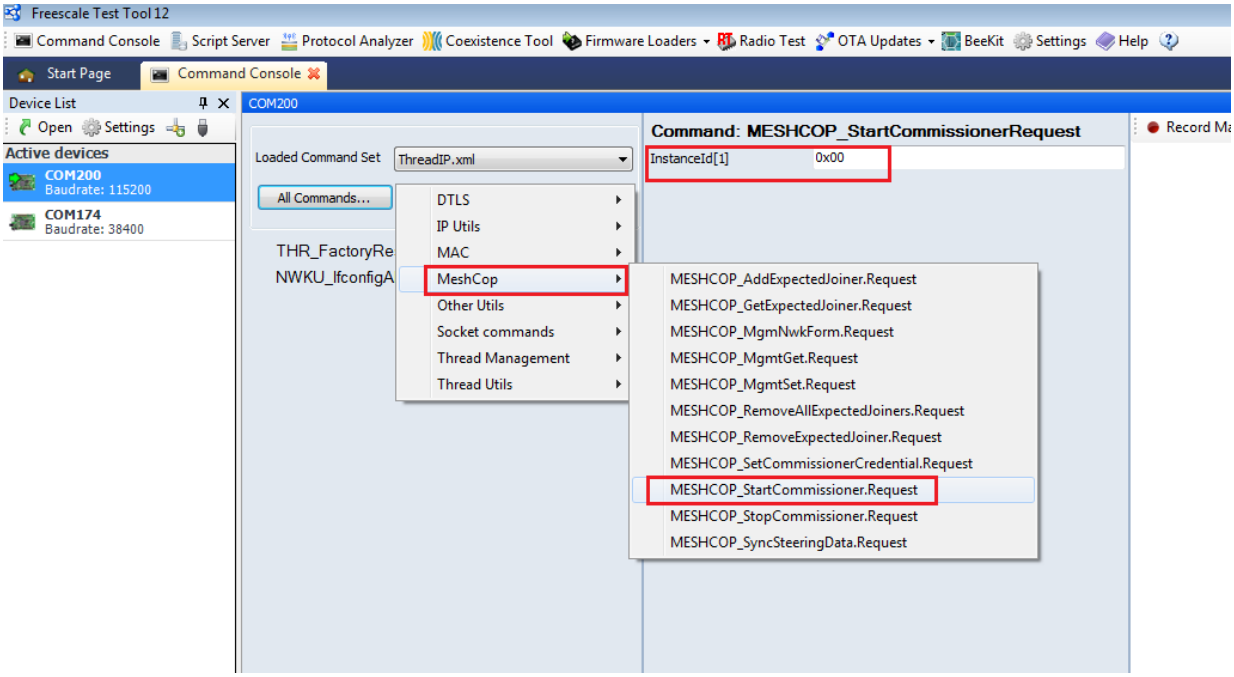

Figure 8. MESHCOP_StartCommissioner.Request

d. Add expected joiner and synchronize steering data. In this scenario we will accept any EUI64 address.

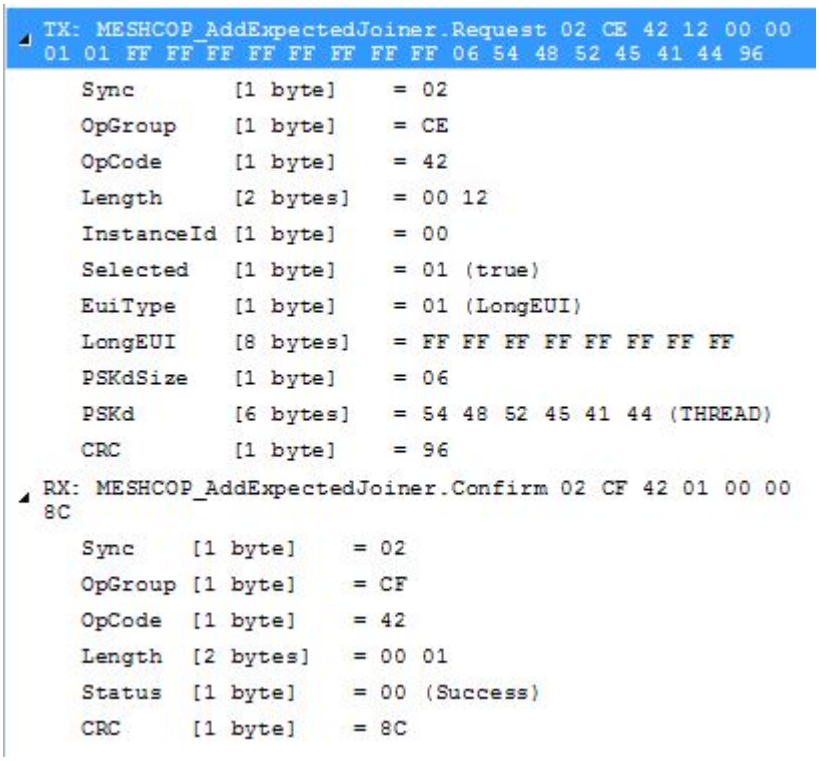

Figure 9. Add Expected Joiner Log

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 = 0x6
- PSKd = THREAD (the same used by OTA Client - default configuration)

MESHCOP_SyncSteeringDataRequest:

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

6. 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
7. 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.

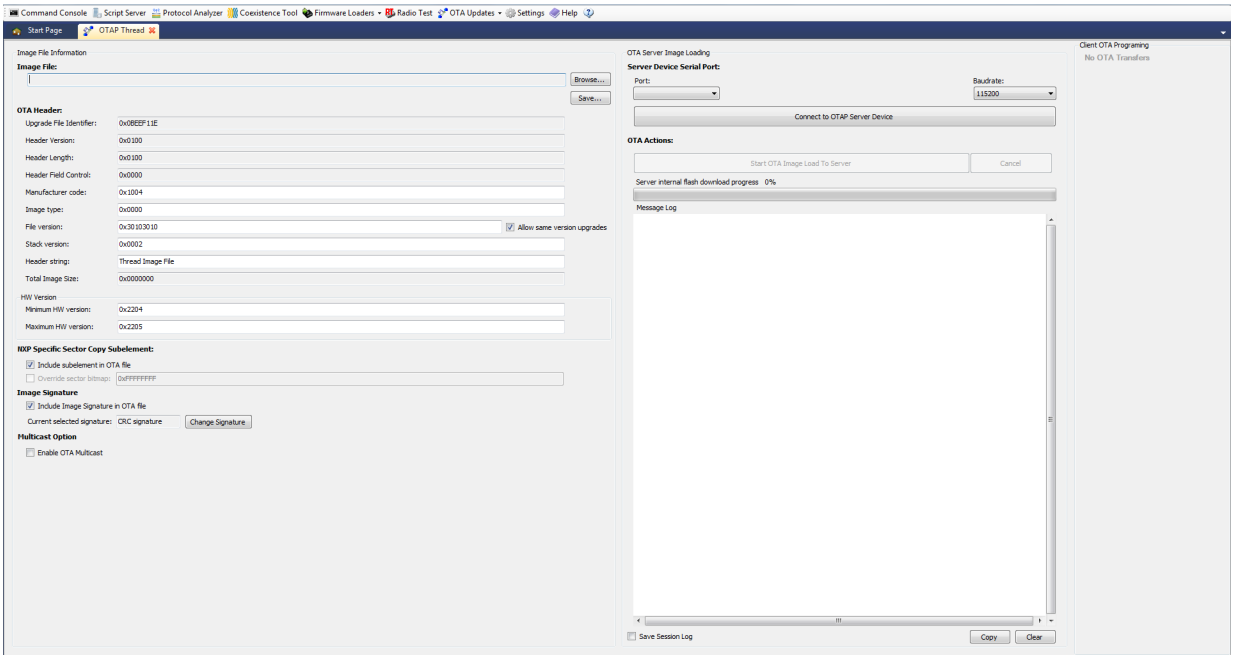

Figure 10. Thread OTAP view

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. It should be noted that USBKW41Z board does not have external flash and thus can work only in dongle mode.

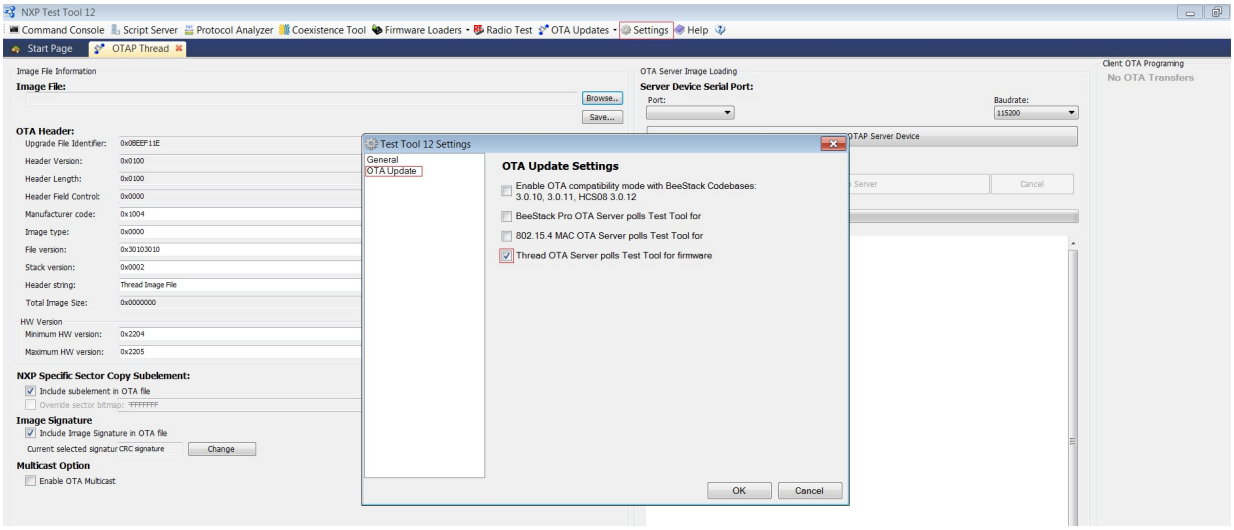

Figure 11. OTA Update settings

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.

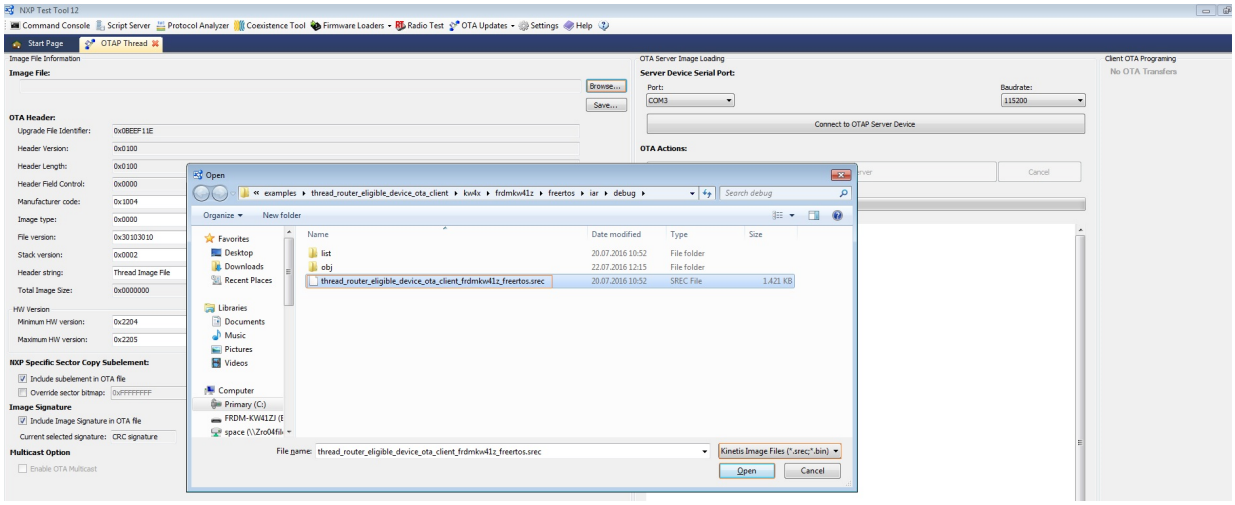

Figure 12. Selecting S-Record file type

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.

h. When deselecting the “Do not include NVM firmware sectors in OTA file” checkbox, the image used has to be compiled with gEraseNVMLink_d=0 in the linker tab of the project's option for any build (bin, srec, etc.). Otherwise, it will cause an invalid length of the image and the OTAP process will fail.

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 .

i. 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.

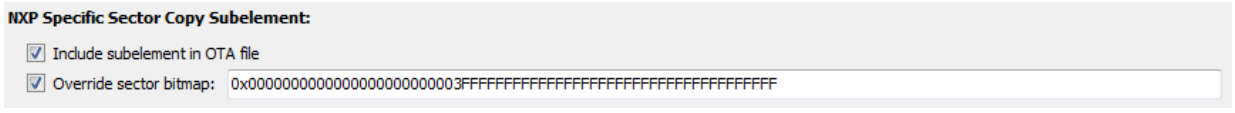

Figure 13. Override sector bitmap

j. 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.

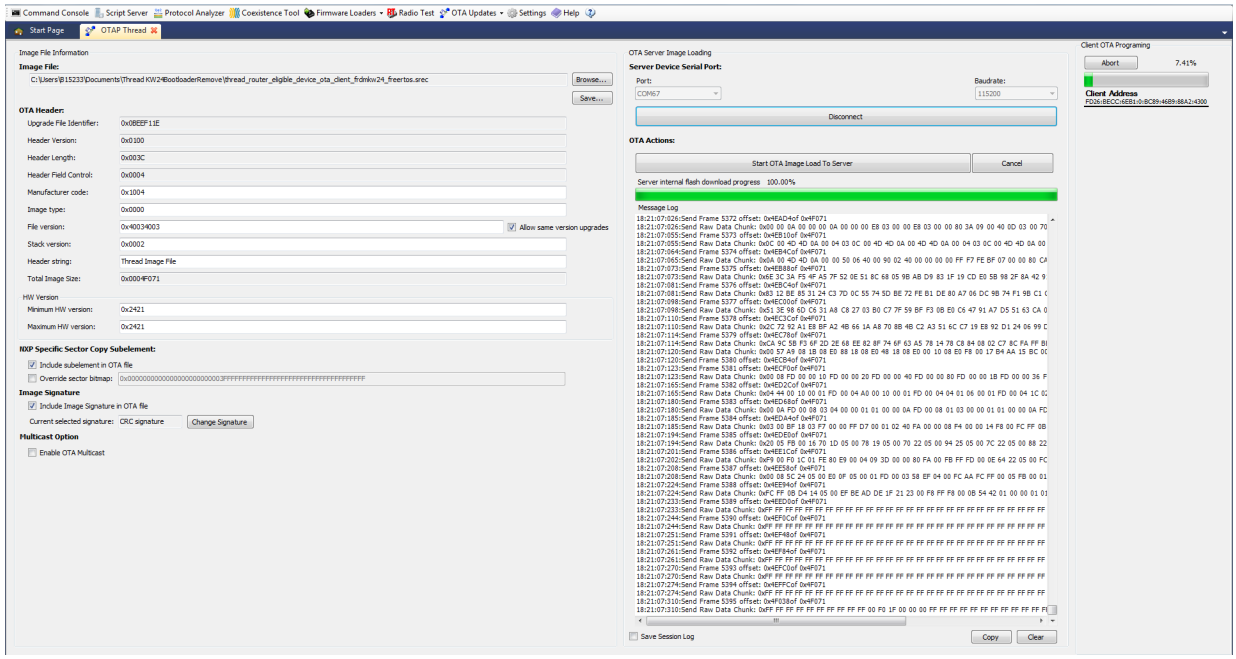

Figure 14. Running OTA procedure

k. 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.

Chapter 6
Running a multicast OTA scenario

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.

The multicast process begins and finishes when the main progress bar reaches 100%. After the multicast process finishes, each client board starts a unicast OTA process with the server, requesting the image chunks that failed to reach them during the multicast process. The status of the unicast transfers can be seen in the right side of the screen, as seen in the unicast OTA scenario. 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.

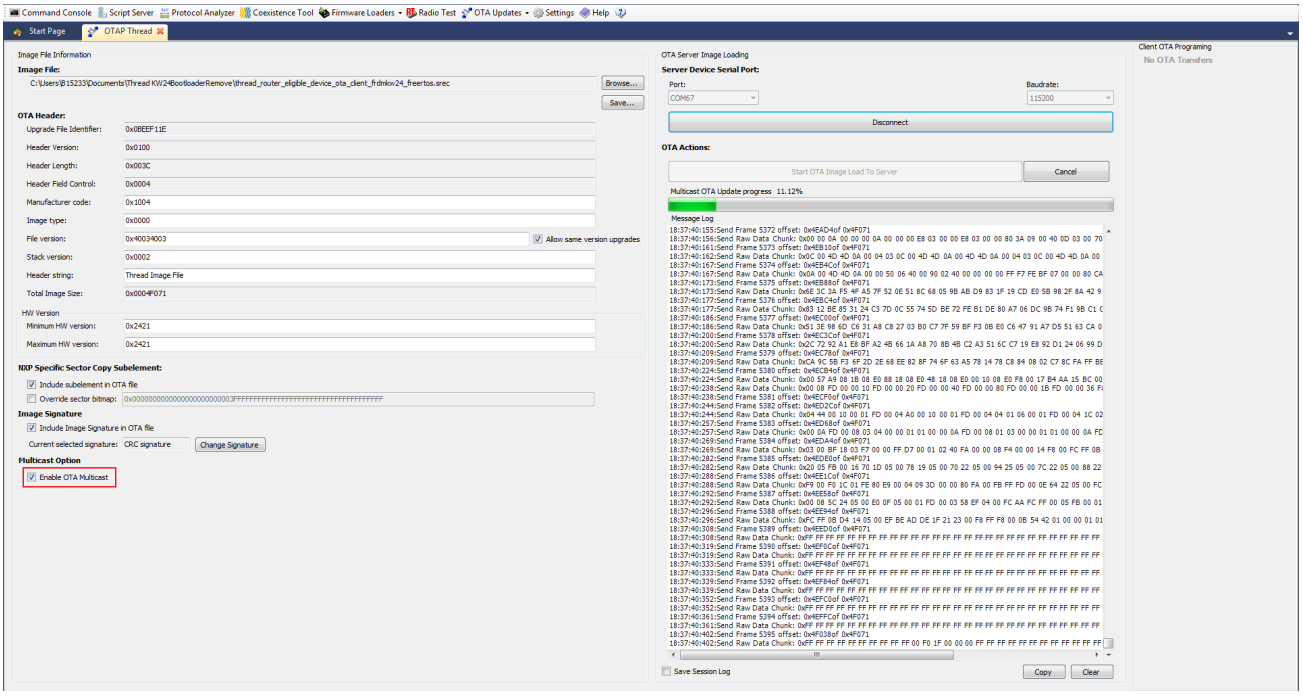

Figure 15. Running OTA procedure multicast

Chapter 7
Revision History

This table summarizes revisions to this document.

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
6	3/2017	Updates for Thread KW41 MCUX Release
7	01/2018	Updates for Thread KW41 Maintenance Release

Chapter 8
Copyright

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support