Application note

The BlueNRG-1, BlueNRG-2 BLE OTA (over-the-air) firmware upgrade

Introduction

This application note describes the BlueNRG-1, BlueNRG-2 “over-the-air” (OTA) firmware upgrade procedures running on top of Bluetooth low energy (BLE) stack provided with the BlueNRG-1, BlueNRG-2 systems-on-chip. First of all, it starts with some concepts related to "over-the-air" firmware upgrading process and then guides the user through all steps required to run some OTA firmware upgrading sessions.

BLE OTA firmware upgrade is supported on the following devices and BLE stack versions:

Table 1. **BLE OTA firmware upgrade**

BLE device	BLE stack version	OTA FW upgrade support
BlueNRG-1	1.x	Yes(1)
BlueNRG-1	2.0	No
BlueNRG-1	2.1 or later	Yes (2)
BlueNRG-2	1.x, 2.x	Yes

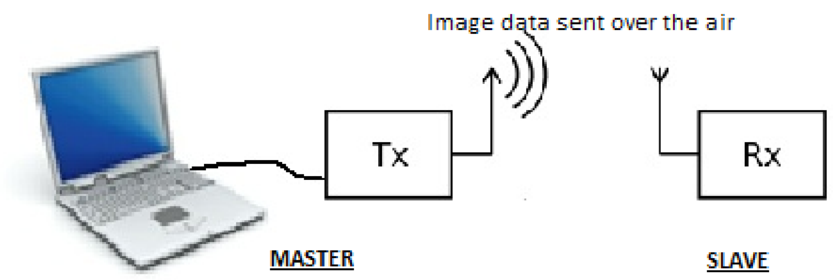
1. *It is provided on the BlueNRG-1 DK SW package supporting the BLE stack v1.x.*
2. *It is supported with the BLE stack v2.1 or later modular approach, basic configuration (BLE_STACK_CONFIGURATION=BLE_STACK_BASIC_CONFIGURATION):*
   - *No controller privacy*
   - *No LE secure connections*
   - *No master/central role*
   - *No data length extension*

Note: *The document content is valid both for the BlueNRG-1 and the BlueNRG-2 device. Any reference to the BlueNRG-1 device is also valid for the BlueNRG-2 device. Any specific difference is highlighted whenever it is needed.*

1 The concept of "over-the-air" firmware upgrade

"Over-the-air" (OTA) firmware upgrade is a protocol that allows a Bluetooth low energy slave device to receive a firmware image over-the-air from a Bluetooth low energy master device and write it into Flash memory. To put things in the right context for Bluetooth low energy technology, OTA firmware upgrade framework defines a service exposing its own characteristics, which can coexist with other services used by any given application running on the BLE stack. The BLE master is a combined system made of a BlueNRG-1, BlueNRG-2 development kit platform connected to a PC through USB. This BlueNRG-1, BlueNRG-2 platform is driven by the BlueNRG GUI. Thanks to this choice, a lot of resources are available from PC, specifically with regards to compilers for firmware image generation and memory space required to store images before they are deployed over-the-air for firmware upgrade.

Figure 1. BlueNRG-1, BlueNRG-2 OTA master and slave devices



2 OTA FW upgrade service description

The OTA FW upgrade service is addressed through the files OTA_btl.[ch] provided within the BlueNRG-1_2 DK SW package (Library\Bluetooth_LE\OTA folder).

A short description about the OTA FW upgrade service and its related characteristics follows:

- Btl OTA service: it is the FW upgrade service
  - aci_gatt_add_service(UUID_TYPE_128, &ota_service_uuid, PRIMARY_SERVICE, 10, &btlServHandle);
- Btl image characteristic : it contains some information about lower and higher bounds of free memory as suggested by the current application that includes the OTA FW upgrade service
  - aci_gatt_add_char(btlServHandle, UUID_TYPE_128, &ota_char_uuid, 8, CHAR_PROP_READ, ATTR_PERMISSION_NONE, GATT_NOTIFY_READ_REQ_AND_WAIT_FOR_APPL_RESP,16, 0, &btlImageCharHandle);
- Btl new image characteristic: it contains the base address and the size of the image that the master pretends to send over-the-air and the notification range requested to the slave for sending acks during OTA FW transfers
  - aci_gatt_add_char(btlServHandle, UUID_TYPE_128, &ota_char_uuid, 9, CHAR_PROP_READ| CHAR_PROP_WRITE|CHAR_PROP_WRITE_WITHOUT_RESP, ATTR_PERMISSION_NONE, GATT_NOTIFY_READ_REQ_AND_WAIT_FOR_APPL_RESP | GATT_NOTIFY_ATTRIBUTE_WRITE,16, 0, &btlNewImageCharHandle);
- Btl new image content characteristic: it contains a 16-byte block of firmware image data sent by the master (through a characteristic write command) along with some control information such as: block sequence number (2 bytes) and checksum for integrity check 1 byte)
  - aci_gatt_add_char(btlServHandle, UUID_TYPE_128, &ota_char_uuid, 20, CHAR_PROP_READ| CHAR_PROP_WRITE|CHAR_PROP_WRITE_WITHOUT_RESP, ATTR_PERMISSION_NONE, GATT_NOTIFY_READ_REQ_AND_WAIT_FOR_APPL_RESP | GATT_NOTIFY_ATTRIBUTE_WRITE,16, 0, &btlNewImageTUContentCharHandle);
- Btl expected image sequence number characteristic, through which the slave device notifies the master about the next block it expects and error conditions
  - aci_gatt_add_char(btlServHandle, UUID_TYPE_128, &ota_char_uuid, 4, CHAR_PROP_NOTIFY| CHAR_PROP_READ, ATTR_PERMISSION_NONE, GATT_NOTIFY_READ_REQ_AND_WAIT_FOR_APPL_RESP, 16, 0, &btlExpectedImageTUSeqNumberCharHandle);

Note: OTA FW upgrade service and characteristics proprietary UUIDs (128 bits) are defined within the file OTA_btl.c.

2.1 OTA firmware upgrade transactions

In this section the steps about OTA firmware upgrade are dealt with:

1. Once the master and slave device running the OTA FW upgrade service are set, a discovery procedure needs to be fulfilled in order to allow the two devices to be connected. Discovery is achieved listening to advertisements coming from the devices within the radio range (active scan) and selecting the ones containing the OTA FW upgrade service UUID (128 bits) within the scan response.
2. Further, the name of the selected device is read from the advertising message to be utilized by master in order to enhance the slave identification procedure.
3. After connection, the master sends ‘ACI_GATT_DISC_CHAR_BY_UUID’ commands in order to read all OTA FW upgrade characteristic handles.
4. The master reads the Btl image characteristic through a ‘ACI_GATT_READ_CHAR_VALUE’ command to be aware of free space on the target slave device Flash memory.
5. Based on the information carried out at the previous step, the master selects a proper image to send over-the-air. The candidate image (placing somewhere on the master as *.bin file) has to fit within the target free Flash range.

6. Once the selection is done, the master sends a 'ACI_GATT_WRITE_CHAR_VALUE' in order to write image base address, size and notification range into the Btl new image characteristic and reads it back through 'ACI_GATT_READ_CHAR_VALUE' in order to verify it.
7. The master writes into the Btl expected image sequence number characteristic descriptor to enable slave notifications for image block sequence numbers and errors. Once the slave receives this command it sends back a notification.
8. The image transfer begins. The master sends the image in blocks of (N*16) bytes through a sequence of 'ACI_GATT_WRITE_WITHOUT_RESP' commands (one for each (N*16)-bytes block). Each time a new write command lands on the target slave it writes the new block of data within the Btl new image content characteristic. Each (N*16)-bytes block comes along with a 2-byte long sequence number and one byte long checksum field in order to check for the sequencing and message integrity at destination. Every time the slave internal buffer gets filled up by (N*16)-byte blocks, it downloads into Flash memory. Once the slave has completed to manage the internal buffer of (N*16)-byte blocks, it sends a notification message back to the master providing the block number of the next expected block. It might notify Flash write errors and Flash verify errors as well on latest blocks, in which case the bootloading session should stop under the assumption that we deal with issues on the destination device Flash.

   N = 1 for the BlueNRG-1, BLE stack v1.x and BLE stack v2.1 or later, BlueNRG-2, BLE stack v2.0.

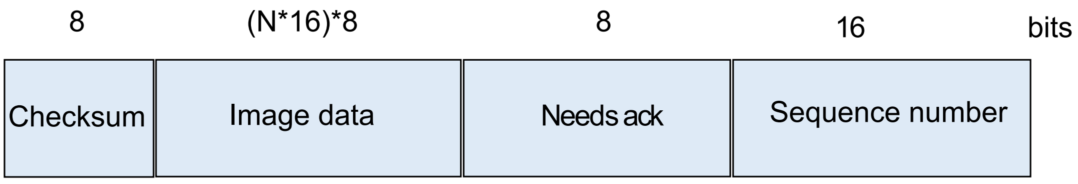
   N = ((OTA_ATT_MTU_SIZE - 3 - 4)/16) for the BlueNRG-2, BLE stack v2.1 or later with data length extension enabled for OTA FW upgrade process.

   On the BlueNRG-2, BLE stack v2.1 or later with data length extension feature (max. PDU length = 251 bytes) and increased ATT_MTU size (> 23 bytes), N is tailored to get an OTA packet size which fits, as much as possible, within the max. allowed BlueNRG-2 OTA client ATT_MTU size (OTA_ATT_MTU_SIZE). This allows the data size to be increased and transferred on each BLE OTA packet at link layer level and the OTA FW upgrade procedure to be speed up.

   The BLE stack v2.1 HCI_LE_SET_DATA_LENGTH() and ACI_ATT_EXCHANGE_MTU() APIs are used in order to set the data length extension feature on both OTA client transmitter and OTA server receiver sides, and agree on the max. supported ATT_MTU sizes.

   The OTA FW packet structure used by the ACI_GATT_WRITE_WITHOUT_RESP command is the following:

   Figure 2. OTA_packet_structure



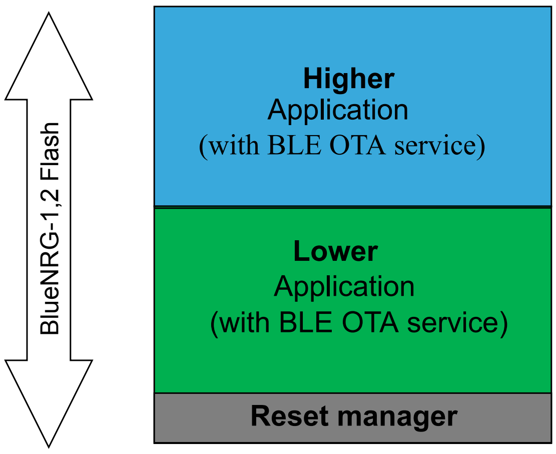
   Where:
   – Checksum byte is used for packet integrity check
   – Image data bytes contain the (N * 16) bytes of input file to be transferred
   – Needs ack byte indicates when OTA client expects a notification from slave during the OTA FW transfer (1: if notification is expected; 0 if no notification is expected)
   – Sequence number bytes are used to indicate the expected image sequence number value used during the OTA FW transfer
9. If the number of bytes downloaded successfully on the destination device Flash is equal to the initially provided image size information, the OTA FW upgrade procedure writes a specific application validity tag on a reserved entry of each application interrupt vector table in order to allow the OTA reset manager to properly identify the address of the new valid application.
10. The application validity tag stored on the interrupt vector table allows OTA FW upgrade session failures to be handled by avoiding to jump to a not valid application (the control is always transferred to the last valid application).

2.2 BLE OTA service firmware upgrade architecture

The BLE OTA service firmware upgrade architecture includes the following components:

- Reset manager
  - It should start after resetting and passing control to the latest updated and valid BLE application.
- Application with BLE OTA service (lower application)
  - It defines the OTA service in order to provide itself the OTA firmware upgrade capability.
  - When running, it allows new application image packets to be got over-the-air and stored in a specific Flash section (higher application storage area).
  - If the OTA FW upgrade process is successfully completed, a SW reset is generated to give control to the new valid higher application through the reset manager.
- Application with BLE OTA service (higher application)
  - It also defines the OTA service in order to provide itself the OTA firmware upgrade capability.
  - When running, it allows new application image packets to be got over-the-air and stored in a specific Flash section (lower application storage area).
  - If the OTA FW upgrade process is successfully completed, a SW reset is generated to give control to the new valid lower application through the reset manager.

Figure 3. **BLE OTA service FW upgrade architecture**

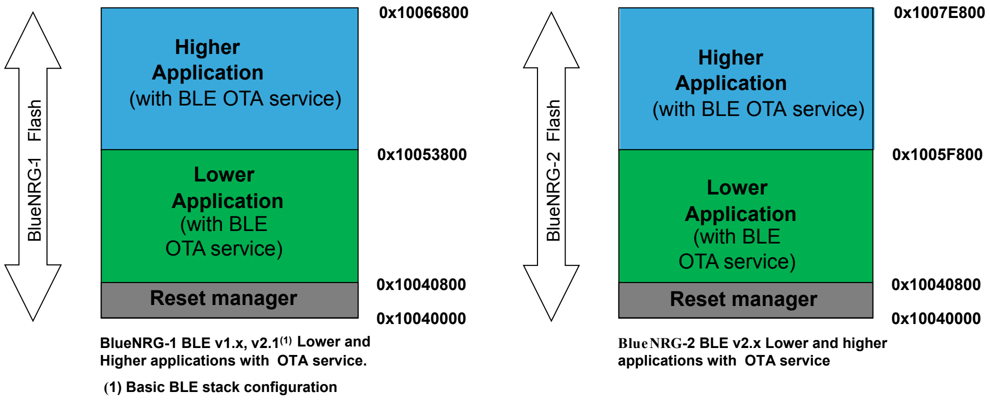

BlueNRG-1,2 Flash
Higher Application (with BLE OTA service)
Lower Application (with BLE OTA service)
Reset manager

2.2.1 The reset manager

The Flash base for BlueNRG-1, BlueNRG-2 starts at 0x10040000, so at the device reset, the micro starts executing application images sitting on that boundary. Once the OTA FW upgrade procedure writes a new image starting from a base address that is above the Flash base, a way to transfer the control towards the new application is needed, every time there is a device reset. For that purpose, every device, including the OTA FW upgrade service, needs at first, to be uploaded with an OTA reset manager starting from the Flash base address. At device reset, the OTA reset manager deals with jumping to the location of the last valid image that was successfully loaded by the OTA FW upgrade procedure, as indicated by the combinations of application validity tags stored on a reserved entry on each application interrupt vector tables, which are set after a successfully OTA FW upgrade session (refer to OTA_Check_Application_Tags_Value() function on OTA_ResetManager.c file). The figure below shows a typical Flash memory layout for BlueNRG-1, BlueNRG-2 devices with OTA FW upgrade service on-board. At the device initial setup, the OTA reset manager itself has to be uploaded at address 0x10040000 and then the lower application at address 0x10040800 (reset manager upper end).

The OTA reset manager application projects + header and source files are provided within the BlueNRG-1_2 DK SW package (BLE_OTA_ResetManager project).

Figure 4. Lower and higher applications with OTA service


BlueNRG-1 Flash
Higher Application (with BLE OTA service)
Lower Application (with BLE OTA service)
Reset manager
0x10066800
0x10053800
0x10040800
0x10040000
BlueNRG-1 BLE v1.x, v2.1(1) Lower and Higher applications with OTA service.
(1) Basic BLE stack configuration
BlueNRG-2 Flash
Higher Application (with BLE OTA service)
Lower Application (with BLE OTA service)
Reset manager
0x1007E800
0x1005F800
0x10040800
0x10040000
BlueNRG-2 BLE v2.x Lower and higher applications with OTA service

2.2.2 BLE application with BLE OTA service

In order to add the BLE OTA service provided within the file OTA_btl.c, steps below (IAR EWARM is taken as reference example) have to be followed:

1. On EWARM workspace, add, ST_OTA_LOWER_APPLICATION=1 or ST_OTA_HIGHER_APPLICATION=1 as preprocessor and linker options, respectively, to build a BLE lower or a higher application with the OTA service. In order to enable the data length extension support on OTA FW upgrade process for the BlueNRG2 device, BLE stack v2.1 or later, the OTA_EXTENDED_PACKET_LEN=1 must be added as preprocessor option. Further, the user application must be built with a modular configuration option which includes the data length extension feature.
2. On EWARM workspace, use the reference linker file to match the Flash layout described on the figure above (it is provided within the BlueNRG-1_2 DK SW package, OTA demo application folders).
3. On EWARM workspace, add reference to file OTA_btl.c and add path related to OTA_btl.h file.
4. On BLE user application, include the header file OTA_btl.h.
5. On BLE user application, call the OTA_Add_Btl_Service() API to add the OTA BLE service and related characteristics.
6. On BLE user application, add OTA service UUID to scan responses as follows: hci_le_set_scan_resp_data(18,BTLServiceUUID4Scan).
7. On BLE user application, aci_gatt_attribute_modified_event() event callback, add the call to the OTA_Write_Request_CB() API.
8. On BLE user application, aci_gatt_read_permit_req_event() event callback, add the call to aci_gatt_allow_read() API (to allow BLE stack to send a response).
9. On BLE user application, add the check if (OTA_Tick() == 1), in order to check when jumping to the new upgraded application by calling the function OTA_Jump_To_New_Application().
10. On BLE user applications, aci_hal_end_of_radio_activity_event() callback, add the call to the OTA_Radio_Activity() API (to synchronize FLASH write operations with radio activity).

Note: In order to clearly identify the required SW components to support the OTA FW upgrade process the #if ST_OTA_FIRMWARE_UPGRADE_SUPPORT preprocessor option should be used for the topics 5, 6, 7, 8, 9 on the list above.

i.e.

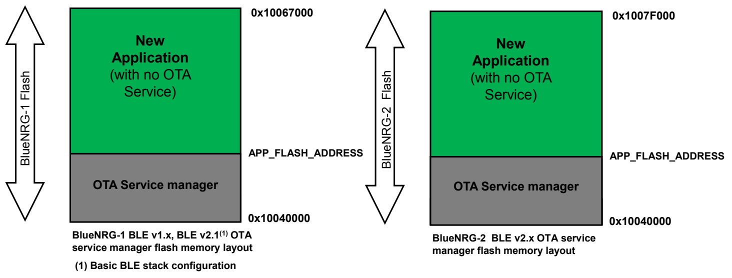
```
#if ST_OTA_FIRMWARE_UPGRADE_SUPPORT

        OTA_Add_Btl_Service();

#endif
```

A specific led is turned on during an OTA upgrade session (i.e. led DL3 on BlueNRG-1, BlueNRG-2 platforms).

2.3 OTA service manager framework

A simpler approach coming from the BLE OTA service architecture described in Figure 4. Lower and higher applications with OTA service, consists of using a basic OTA service manager application, which includes the BLE OTA service and its characteristics and the OTA reset manager capabilities.

Figure 5. OTA service manager Flash memory layout


BlueNRG-1 Flash
New Application (with no OTA Service)
OTA Service manager
0x10067000
APP_FLASH_ADDRESS
0x10040000
BlueNRG-1 BLE v1.x, BLE v2.1(1) OTA service manager flash memory layout
(1) Basic BLE stack configuration
BlueNRG-2 Flash
New Application (with no OTA Service)
OTA Service manager
0x1007F000
APP_FLASH_ADDRESS
0x10040000
BlueNRG-2 BLE v2.x OTA service manager flash memory layout

APP_FLASH_ADDRESS is defined as follows:

Table 2. APP_FLASH_ADDRESS

BLE device	BLE stack version	APP_FLASH_ADDRESS	Notes
BlueNRG-1	V1.x	0x10050800	OTA FW upgrade is not supported with the BlueNRG-1, BLE stack v2.0
BlueNRG-1	V2.0	Not Applicable	OTA FW upgrade is not supported with the BlueNRG-1, BLE stack v2.0
BlueNRG-2	v2.0	0x10055800	No modular configuration is supported on BLE stack v2.0
BlueNRG-1	v2.1 or later	0x10051000	BLE stack basic configuration
BlueNRG-2	v2.1 or later	0x10051800	BLE stack basic configuration with data length extension

The BLE OTA service manager architecture includes the following components:

- OTA service manager
  - It contains BLE OTA service and characteristics only and it provides the OTA firmware upgrade capability to any application. It also includes the OTA reset manager capability to pass control to the latest updated and valid applications.
- New application
  - It is the application image downloaded through the OTA service manager.
  - It does not require to include the OTA FW upgrade service.
  - In order to use this capability, the application has to activate the OTA service manager only (a specific OTA_Switch_To_OTA_Service_Manager_Application() API is provided within OTA_btl.c file).
  - It is placed at fixed Flash address.

2.3.1 OTA service manager application

The OTA service manager acts as a standalone OTA FW upgrade application providing the BLE OTA firmware upgrade capability to any application, which wants to use this feature, without including any BLE OTA service. It also includes the OTA reset manager capability to pass control to the latest updated and valid BLE application.

The OTA service manager application projects + header and source files are provided within the BlueNRG-1_2 DK SW packages (BLE_OTA_ServiceManager workspace).

In order to enable the data length extension support on OTA FW upgrade process for the BlueNRG2 device, BLE stack v2.1 or later, the OTA_EXTENDED_PACKET_LEN=1 must be added as preprocessor option on OTA service manager application.

Further on the BlueNRG2 device, BLE stack v2.1 or later, the OTA service manager application is built with the modular approach, OTA basic configuration (BLE_STACK_CONFIGURATION=BLE_OTA_BASIC_CONFIGURATION):

- No controller privacy
- No LE secure connections
- No master/central role
- Data length extension

2.3.2 New BLE application

BlueNRG-1, BlueNRG-2 applications, which want to use the OTA service manager capability, are only requested to address these basic steps:

1. On EWARM workspace, add ST_USE_OTA_SERVICE_MANAGER_APPLICATION=1 as preprocessor and linker options.
2. On EWARM workspace, use the reference linker file to match the Flash layout (it is provided within the BlueNRG-1_2 DK SW packages, demo application folder).
3. On EWARM workspace, add reference to file OTA_btl.c and add path related to OTA_btl.h file.
4. On new user application source file, include the header file OTA_btl.h.
5. On new user application source file, add some code to allow the application to launch the OTA service manager application. A simple button could be used to control the call to OTA_Jump_To_Service_Manager_Application(); (this API is provided within OTA_btl.c file).
6. On new user application, aci_hal_end_of_radio_activity_event() callback, add the call to the OTA_Radio_Activity() API (to synchronize FLASH write operations with radio activity).

Note: *When using the OTA service manager the new application does not need to include any OTA FW upgrade service. Furthermore, the application must be always placed at a fixed base address.*

3 Hardware and software resources

Before describing how to setup BLE OTA firmware upgrade demonstration applications, follow the list of required hardware and software resources below (using the BlueNRG-2 device):

Table 3. **OTA FW upgrade HW and SW resources**

Resources
2 BlueNRG-2 platforms and related USB cables (i.e. STEVAL-IDB008Vx, x = 1,2) 1 ST-LINK/V2 tool
BlueNRG-1_2 DK x.x.x resources: • OTA reset manager and service manager • BLE sensor demo and BLE chat applications built to support OTA FW upgrade service or the OTA service manager approaches • BlueNRG-2 navigator PC application • BlueNRG-1 flasher PC application
BlueNRG graphical user interface (GUI) available on the STSW-BNRGUI SW package. OTA_Central.py script implementing the OTA FW upgrade client side can also be used (it is also available on the STSW-BNRGUI SW package).
PC with the following resources: Windows® XP SP3 or Windows® Vista or Windows®7 At least 128 Mbytes of RAM 2 x USB port 40 Mbytes of hard disk space available Adobe Reader 6.0 or later. IAR embedded workbench 7.70 or later (IAR EWARM is taken as reference toolchain)

4 Running the BlueNRG-1_2 DK SW packages OTA demo applications

The demonstration application projects "Sensor demo" and "BLE chat" come with specific workspaces allowing both of approaches to be exercised:

1. BLE application built to include the OTA FW upgrade service and use the OTA reset manager
2. BLE application built to use the BLE OTA service manager

A simple description of the available project workspaces with OTA firmware upgrade features for the sensor demo BLE application as follows:

Table 4. **Sensor demo IAR workspaces with OTA FW upgrade capabilities**

Sensor demo application project workspace	Description
LowerApp_OTA	Application configuration including the OTA FW upgrade service with “Lower Application” memory layout (refer to Figure 4. Lower and higher applications with OTA service and Section 2.2 BLE OTA service firmware upgrade architecture)
HigherApp_OTA	Application configuration including the OTA FW upgrade service with “Higher Application” memory layout (refer to Figure 4. Lower and higher applications with OTA service and Section 2.2 BLE OTA service firmware upgrade architecture)
Use_OTA_ServiceManager	Application configuration using the “OTA Service Manager” memory layout (refer to Figure 5. OTA service manager Flash memory layout and Section 2.3 OTA service manager framework)

A simple description of the available project workspaces with OTA firmware upgrade features for the BLE chat application as follows:

Table 5. **BLE_Chat demo IAR workspaces with OTA FW upgrade capabilities**

BLE_Chat application project workspace	Description
Server_LowerApp_OTA	Server application configuration including the OTA FW upgrade service with “Lower Application” memory layout (refer to refer to Figure 4. Lower and higher applications with OTA service and Section 2.2 BLE OTA service firmware upgrade architecture)
Server_HigherApp_OTA	Server application configuration including the OTA FW upgrade service with “Higher Application” memory layout (refer to refer to Figure 4. Lower and higher applications with OTA service and Section 2.2 BLE OTA service firmware upgrade architecture)
Server_Use_OTA_ServiceManager	Server application configuration using the “OTA Service Manager” memory layout (refer to Figure 5. OTA service manager Flash memory layout) and Section 2.3 OTA service manager framework).

Note:
- *OTA_Central.py is a python script, which implements the OTA FW upgrade client side, in line with the OTA FW upgrade architecture described on Section 2.2 BLE OTA service firmware upgrade architecture.*
- *BLE Beacon application also provides a workspace example (Use_with_OTA_ServiceManager) to use the Beacon application with the OTA service manager.*

4.1 BLE application built to include the OTA FW upgrade service

A step-by-step description about how to run the BlueNRG GUI OTA bootloader tool using the BlueNRG-2 sensor demo application built with the OTA service support follows (STEVAL-IDB008Vx (x=1,2) is used as reference platform and IAR, EWARM as reference toolchain):

1. Plug two BlueNRG-2 STEVAL-IDB008Vx (x=1,2) platforms to USB connectors on a PC.
2. Select one of the STEVAL-IDB008Vx (x=1,2) kit platform and connect to the STLINK tool.
3. Open EWARM IAR tool.
4. Open the BLE_OTA_ResetManager.eww IAR project, and select project, download, erase memory to erase the device Flash.
5. Build and download the related OTA reset manager application image on the selected platform (pre-built image is available on Firmware\BLE_Examples folder and it can be loaded through BlueNRG-2 navigator PC tool or BlueNRG-1 flasher tool).
6. Open the BLE_SensorDemo.eww IAR project, LowerApp_OTA workspace and build and download the related application image on the selected platform (pre-built image is available on Firmware\BLE_Examples folder and it can be loaded through BlueNRG-2 navigator or flasher tools).
7. [*Optional*] Get a “BlueNRG app for smartphones (IoS or Android)” to discover, connect and play with the sensor demo application just loaded on the BlueNRG-2 platform at step 6.
8. [*Optional*] If actions at step 7 are taken, disconnect the “BlueNRG app for smartphones (IoS or Android)” from the BlueNRG-2 platform. The lowest sensor demo application disconnected, starts advertising. This makes the application as a potential candidate for a BLE OTA firmware upgrade procedure through the BlueNRG-1 GUI bootloader tool or the OTA_Central.py script.
9. Program the second BlueNRG-2 kit platform with the required DTM application to be used with the BlueNRG GUI (pre-built DTM image DTM_UART.hex is available on Firmware\BLE_Examples folder and it can be loaded through BlueNRG-2 navigator or flasher tools).
10. Open the BlueNRG GUI on the PC and select the COM port related to the BlueNRG-2 platform configured on step 9, through the drop down “Port” and press ‘Open’.
11. Select ‘Tools’ - ‘OTA bootloader’ to open up the dialog containing OTA FW upgrade actions and press ‘Search for devices’.

12. After ‘Search for devices’, the GUI starts the discovery process and gets back with some information about the address and application names of the devices running OTA FW upgrade service within the radio range. Once the previous process ends, the device list can be opened up through the combo box arrow below the ‘Search for devices’ button and the user can feel free to choose the device it intends to connect for the firmware upgrade process using the “Connect” button (application address and names are displayed). If user realizes he has connected the wrong device, he can just press the ‘Force Disconnection’ button and get back to the device selection within the combo box. After the device selection, the connection through ‘Connect’ button and reading of the related free memory range, user is requested to provide the new image file compiled with a base address and size fitting within the expected range on the slave device.

For that purpose, the user can open the SensorDemo IAR workspace HigherApp_OTA, and build the related BLE_SensorDemo_HigherApp_OTA.bin image (available in BLE_SensorDemo\EWARM\ BLE_SensorDemo_HigherApp_OTA\Exe folder) and contain a version of the sensor demo that is compiled with a base address equal to the highest address.

Figure 6. OTA process: OTA bootloader window

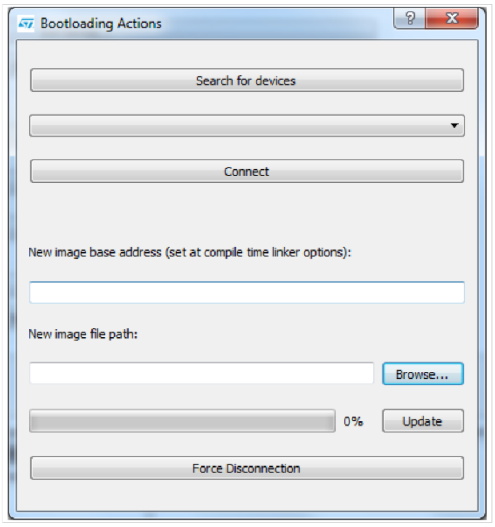
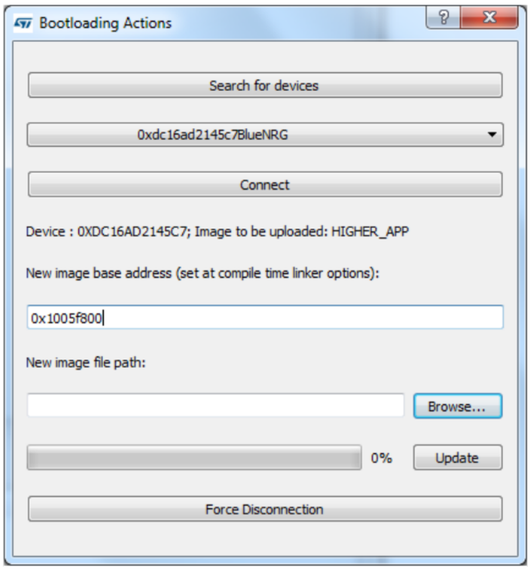


Figure 7. OTA process: search for devices and connect

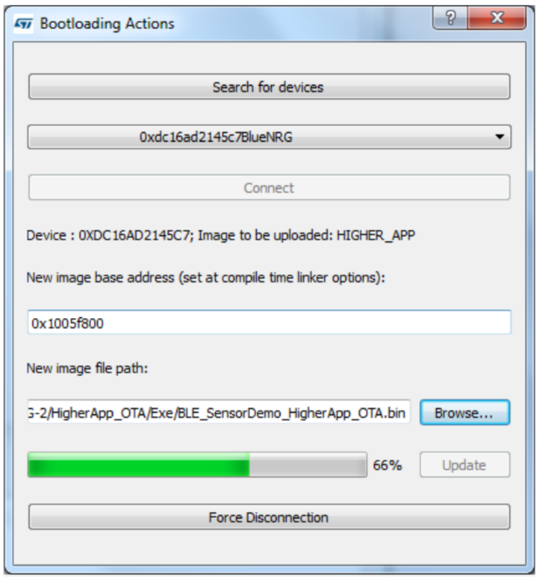


Device : 0XDC16AD2145C7; Image to be uploaded: HIGHER_APP

New image base address (set at compile time linker options): 0x1005f800

13. Once user browses to the selected application image built with the proper base address, he can press ‘Update’ button to start the OTA firmware upgrade process: a progress bar provides awareness for the time needed until process ends:

Figure 8. **OTA process: bootloader upgrade**



Device : 0XDC16AD2145C7; Image to be uploaded: HIGHER_APP

New image base address (set at compile time linker options):

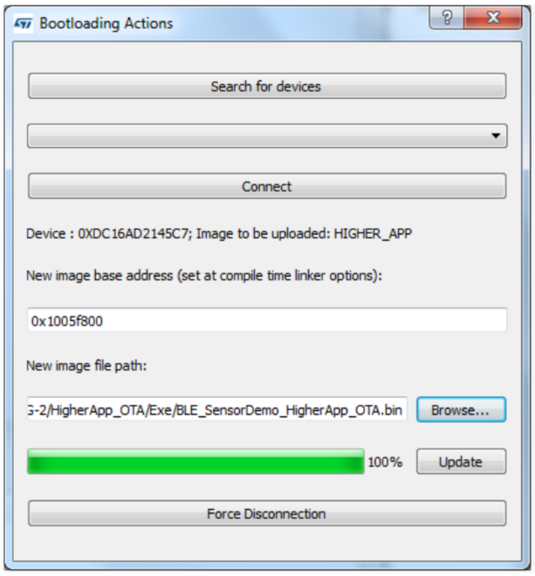
0x1005f800

New image file path:

3-2/HigherApp_OTA/Exe/BLE_SensorDemo_HigherApp_OTA.bin

14. On process completed the new application is launched.

Figure 9. **OTA process: bootloader upgrade completed**



Device : 0XDC16AD2145C7; Image to be uploaded: HIGHER_APP

New image base address (set at compile time linker options):

0x1005f800

New image file path:

3-2/HigherApp_OTA/Exe/BLE_SensorDemo_HigherApp_OTA.bin

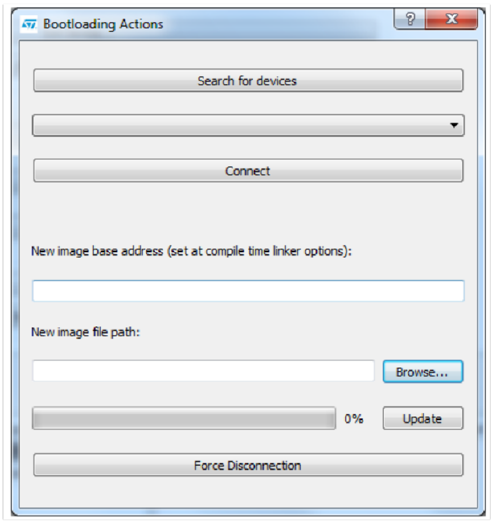
Note:
- *Similar procedure can be used for the BlueNRG-2 BLE chat demo application in order to verify the BLE OTA FW upgrade service functionalities (refer to the BLE_Chat\EWARM\BLE_Chat.eww, Server_LowerApp_OTA, Server_HigherApp_OTA workspaces).*
- *The OTA_Central.py script also provides the same OTA client functionalities as the BlueNRG GUI OTA bootloader tool.*

4.2 BLE application built to use the BLE OTA service manager

A step-by-step description about how to run the BlueNRG GUI OTA bootloader tool using the BlueNRG OTA service manager and SensorDemo, BLE chat workspaces addressing the OTA service manager configuration, as follows (STEVAL-IDB008Vx (x=1,2) is used as reference platform and IAR, EWARM ad reference toolchain):

1. Plug two BlueNRG-2 STEVAL-IDB008Vx (x=1,2) platforms to the USB connectors on a PC.
2. Select one of the kit platform and connect to the STLINK tool.
3. Open EWARM IAR tool.
4. Open the BLE_OTA_ServiceManager.eww IAR project, and select Project, Download, Erase memory to erase device flash.
5. Build and download the related application image on the selected platform (pre-built image is available on Firmware\BLE_Examples folder and it can be loaded through BlueNRG-2 navigator or Flasher tools).
6. Open the BLE_SensorDemo.eww IAR project, Use_OTA_ServiceManager workspace and build and download the related application image on the same platform (pre-built image is available on Firmware \BLE_Examples folder and it can be loaded through BlueNRG-2 navigator or Flasher tools).
7. [Optional] Get a "BlueNRG app for smartphones (IoS or Android)" to discover, connect and play with the sensor demo application just loaded on the BlueNRG-2 platform at step 6 .
8. [Optional] If actions at step 7 are taken, disconnect the "BlueNRG app for smartphones (IoS or Android)" from the BlueNRG-2 kit platform
9. Program the second BlueNRG-2 kit platform with the required DTM application to be used with the BlueNRG GUI (pre-built DTM image DTM_UART.hex is available on Firmware\BLE_Examples folder and it can be loaded through BlueNRG-2 navigator or flasher tools).
10. Open the BlueNRG GUI on the PC and select the COM port related to the BlueNRG-2 platform configured on step 9, through the drop down "Port" and press 'Open'.
11. If actions at steps 7 and 8 are taken, on BlueNRG-2 platform running the SensorDemo application, press the button PUSH1 in order to activate the OTA service manager; this action gives the control to the OTA service manager application which takes care of the OTA firmware upgrade process.
12. On BlueNRG GUI select 'Tools' -'OTA bootloader' to open up the dialog containing OTA FW upgrade actions and press 'Search for devices'.

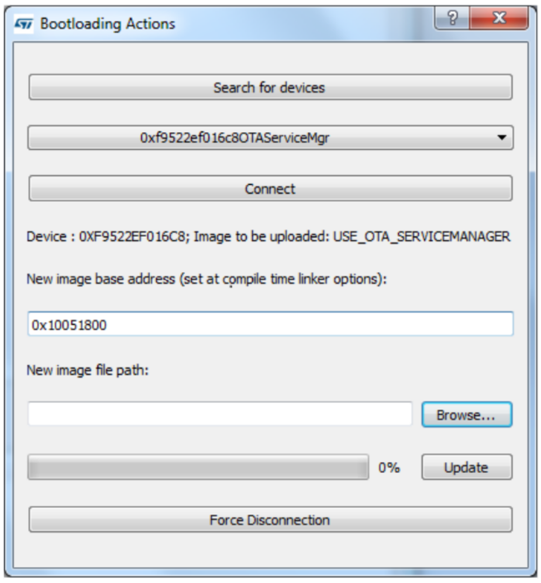
Figure 10. OTA service manager process: OTA bootloader window



13. After 'Search for devices', the GUI starts the discovery process and gets back with some information about the address and application names of the devices running OTA FW upgrade service within the radio range. Once the previous process ends, the device list can be opened up through the combo box arrow below the 'Search for devices' button and the user can find the device running the OTA "Service Manager" and press 'Connect'. If user realizes he has connected the wrong device, he can just press the 'Force Disconnection' button and get back to the device selection within the combo box. After the device selection, connection through 'Connect' button and reading of the related free memory range, user has to provide the new image file compiled with a base address and size fitting within the expected range on the BLE device:

– For that purpose, user can open the BLE Char IAR workspace Use_OTA_ServiceManager, and build the related BLE_Chat_Server_Use_OTA_ServiceManager.bin image (available in BLE_Chat\EWARM\ BLE_Chat_Server_Use_OTA_ServiceManager \Exe folder) containing a version of the BLE chat demo that is compiled with the required base address.

Figure 11. **OTA service manager process: search for devices and connect**



Device : 0XF9522EF016C8; Image to be uploaded: USE_OTA_SERVICEMANAGER

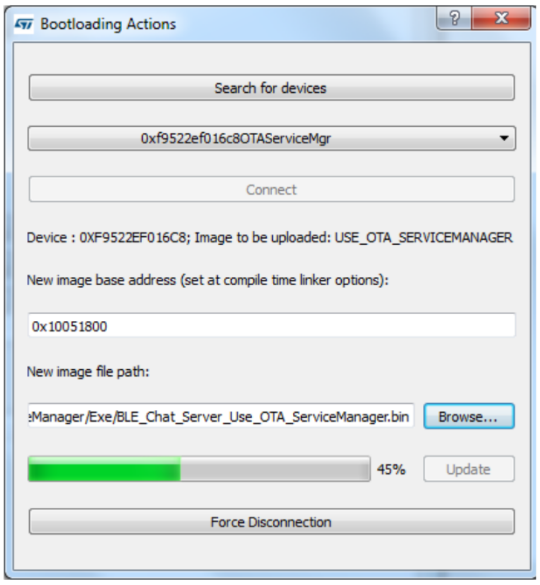
New image base address (set at compile time linker options):

0x10051800

New image file path:

14. User browses to the application image BLE_Chat\EWARM\ BLE_Chat_Server_Use_OTA_ServiceManager \Exe\BLE_Chat_Server_Use_OTA_ServiceManager.bin built with the proper base address, he can press 'Update' button to start the OTA bootloading process: a progress bar provides awareness for the time needed until the process completion.

Figure 12. **OTA service manager process: bootloader upgrade**



Device : 0XF9522EF016C8; Image to be uploaded: USE_OTA_SERVICEMANAGER

New image base address (set at compile time linker options):

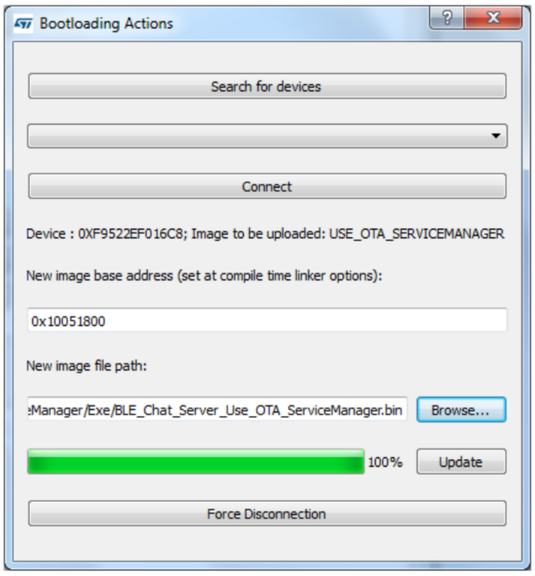
0x10051800

New image file path:

:Manager/Exe/BLE_Chat_Server_Use_OTA_ServiceManager.bin

15. When the process is completed, the new BLE chat application is launched.

Figure 13. OTA service manager process: bootloader upgrade completed



Device : 0XF9522EF016C8; Image to be uploaded: USE_OTA_SERVICEMANAGER

New image base address (set at compile time linker options):

0x10051800

New image file path:

:Manager/Exe/BLE_Chat_Server_Use_OTA_ServiceManager.bin

100%

Note: The OTA_Central.py script also provides the same OTA client functionalities as the BlueNRG GUI OTA bootloader tool.

5 List of acronyms

Table 6. List of acronyms used in this document

Term	Meaning
BLE	Bluetooth low energy
BTL	Bootloader
FW	Firmware
GUI	Graphical user interface
OTA	Over-the-air
SW	Software
USB	Universal serial bus

Revision history

Table 7. **Document revision history**

Date	Version	Changes
06-Jul-2016	1	Initial release.
02-Aug-2017	2	Added reference to the BlueNRG-2 device support and OTA FW upgrade support based on BLE stack version. Updated Figure 4: "OTA service manager Flash memory layout", Figure 6: "OTA process: search for devices and connect", Figure 7: "OTA process: bootloader upgrade", Figure 8: "OTA process: bootloader upgrade completed", Figure 9: "OTA service manager process: OTA bootloader window", Figure 10: "OTA service manager process: search for devices and connect", Figure 11: "OTA service manager process: bootloader upgrade" and Figure 12: "OTA service manager process: bootloader upgrade completed".
20-Jun-2018	3	Updated Table 1. BLE OTA firmware upgrade, and Section 2 OTA FW upgrade service description, Section 2.1 OTA firmware upgrade transactions, Section 2.3 OTA service manager framework, Section 4 Running the BlueNRG-1_2 DK SW packages OTA demo applications. Updated Figure 3. BLE OTA service FW upgrade architecture, Figure 4. Lower and higher applications with OTA service, Figure 5. OTA service manager Flash memory layout, , Figure 11. OTA service manager process: search for devices and connect, Figure 12. OTA service manager process: bootloader upgrade, and Figure 13. OTA service manager process: bootloader upgrade completed.
20-Sep-2018	4	Updated Section 2.1 OTA firmware upgrade transactions.
30-Nov-2018	5	Updated Figure 7. OTA process: search for devices and connect, Figure 8. OTA process: bootloader upgrade, Figure 9. OTA process: bootloader upgrade completed, Figure 11. OTA service manager process: search for devices and connect, Figure 12. OTA service manager process: bootloader upgrade, Figure 13. OTA service manager process: bootloader upgrade completed.

Contents

List of tables

List of figures