Realtek Ameba Z OTA upgrade

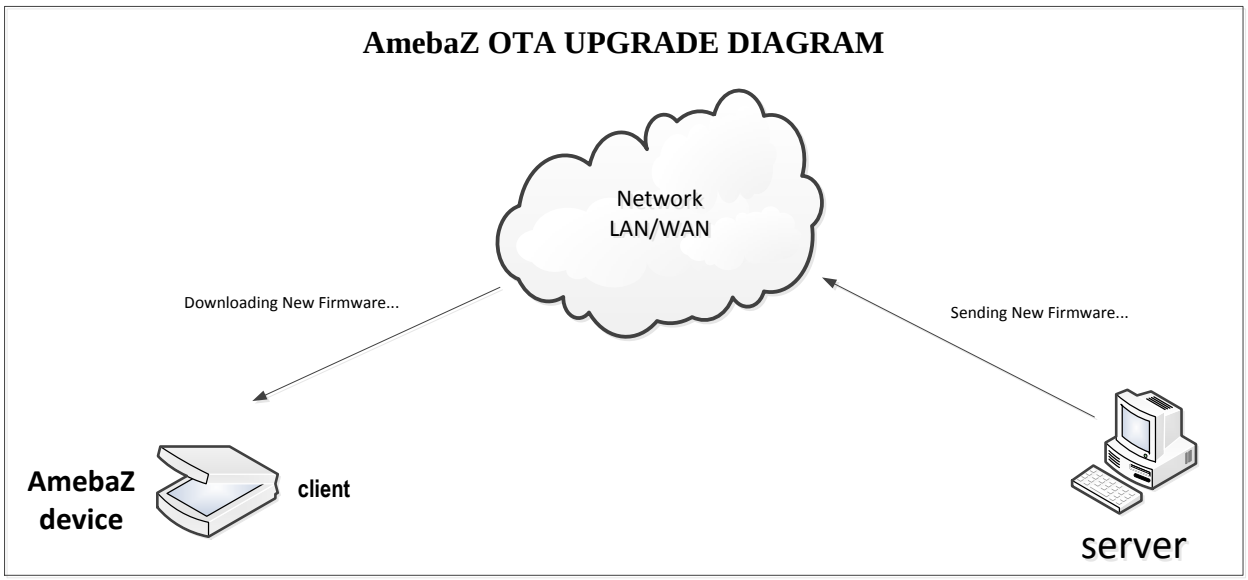
This document introduces how to update AmebaZ firmware over the air.

Table of Contents

1 Introduction 3
2 Firmware Image Output 4
3 File Format in OTA Upgrade 5
3.1 Firmware File Format 5
3.2 Firmware File Header Format 5
4 Firmware Upgrade over the Air 6
4.1 Overview 6
4.1.1 OTA Operation Flow 6
4.1.2 Boot Process Flow 7
4.1.3 Upgraded Partition 8
4.2 Implement OTA over Wi-Fi 8
4.2.1 OTA Using Local Download Server 8
4.2.2 Build OTA Application Image 8
5 OTA Upgrade from Local Server 12
5.1 Diagram 12
5.2 Steps for Local OTA Upgrade 12
5.3 OTA Firmware Swap Diagram 13
5.4 OTA Signature 14

1 Introduction

Over-the-air programming (OTA) provides a methodology of updating device firmware remotely via TCP/IP network connection.



2 Firmware Image Output

After building project source files in SDK, there are two binary files will be generated automatically for upgrade selection.

image2_all_ota1.bin or ***image2_all_ota1.bin-en*** <-------> OTA1

image2_all_ota2.bin or ***image2_all_ota2.bin-en*** <-------> OTA2

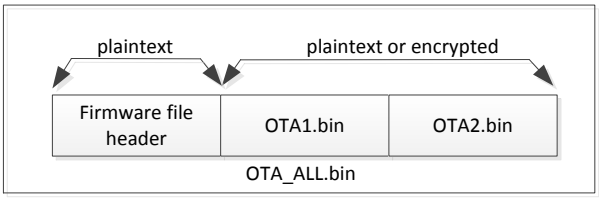
Then use image tool and these two images to generate the OTA image ***OTA_All.bin***.

Refer to: AN0112 Realtek Ameba-Z Image Tool user manual

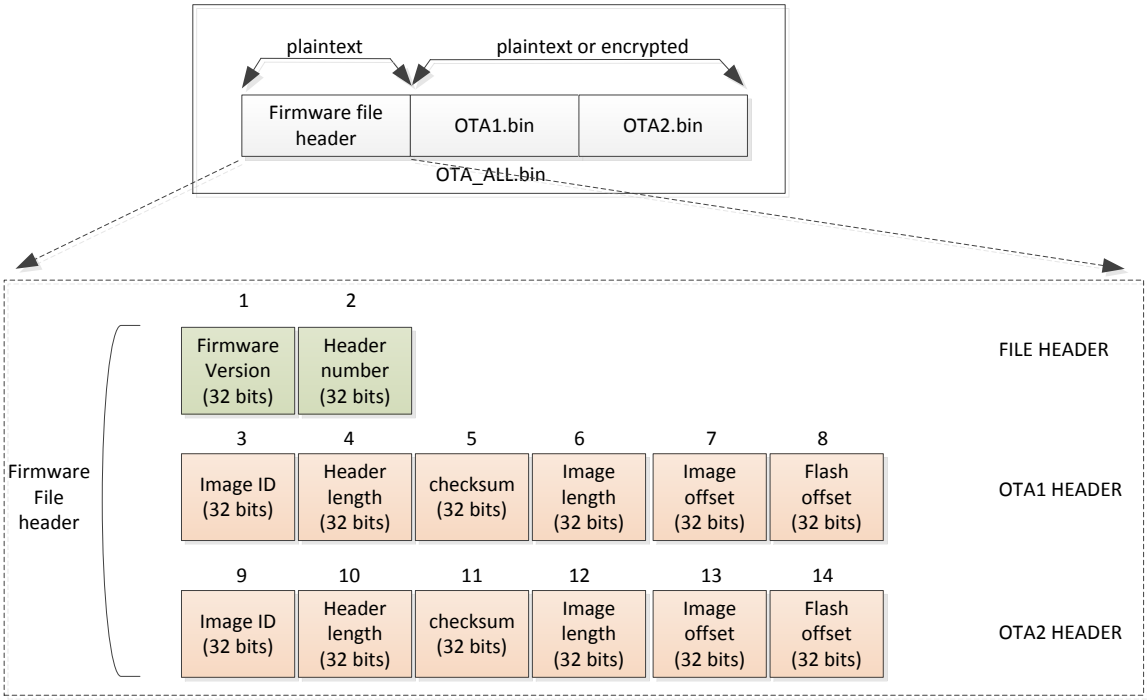
3 File Format in OTA Upgrade

3.1 Firmware File Format

This figure below shows the general firmware file format when AmebaZ downloads new firmware from server.



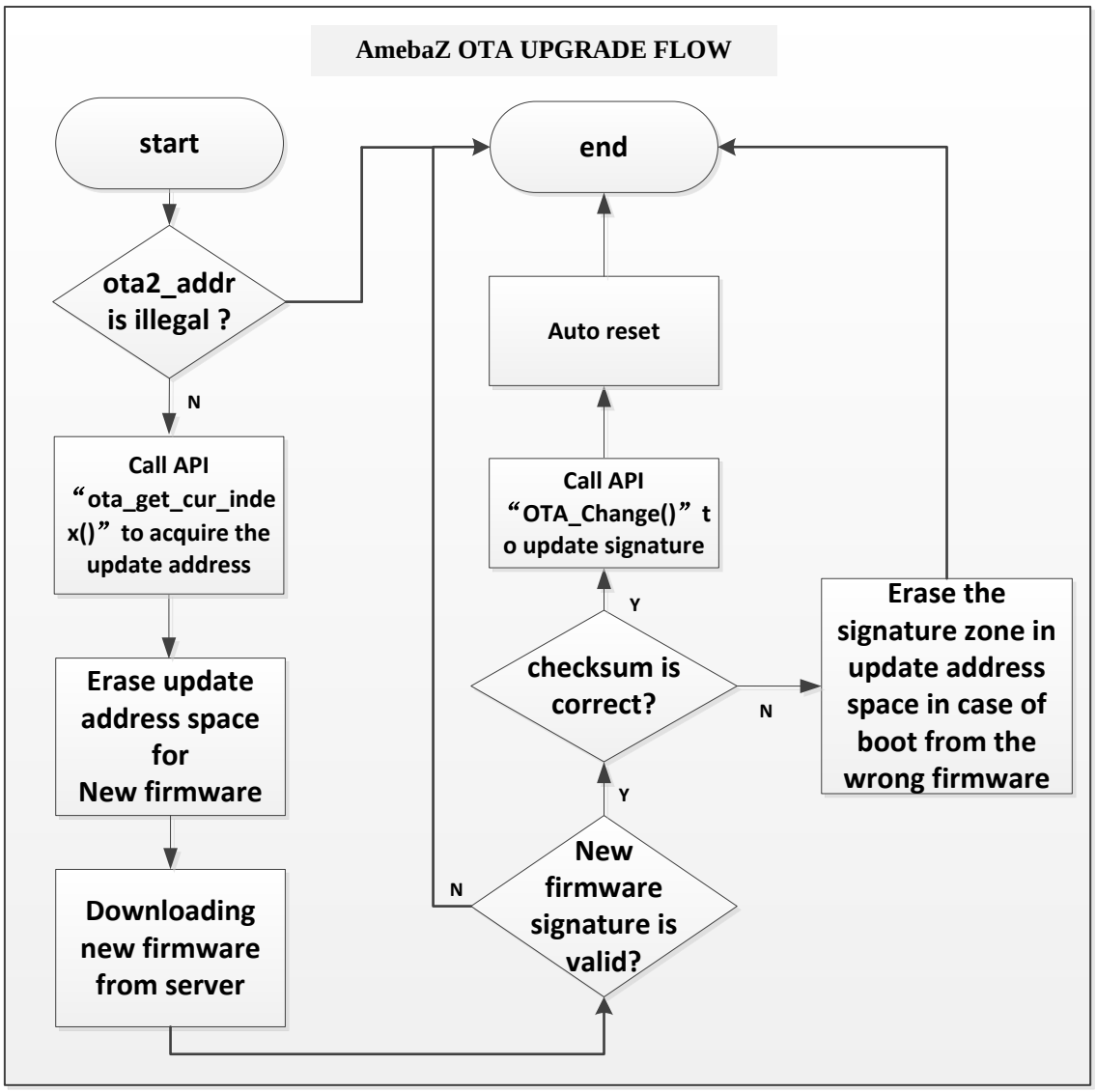
3.2 Firmware File Header Format



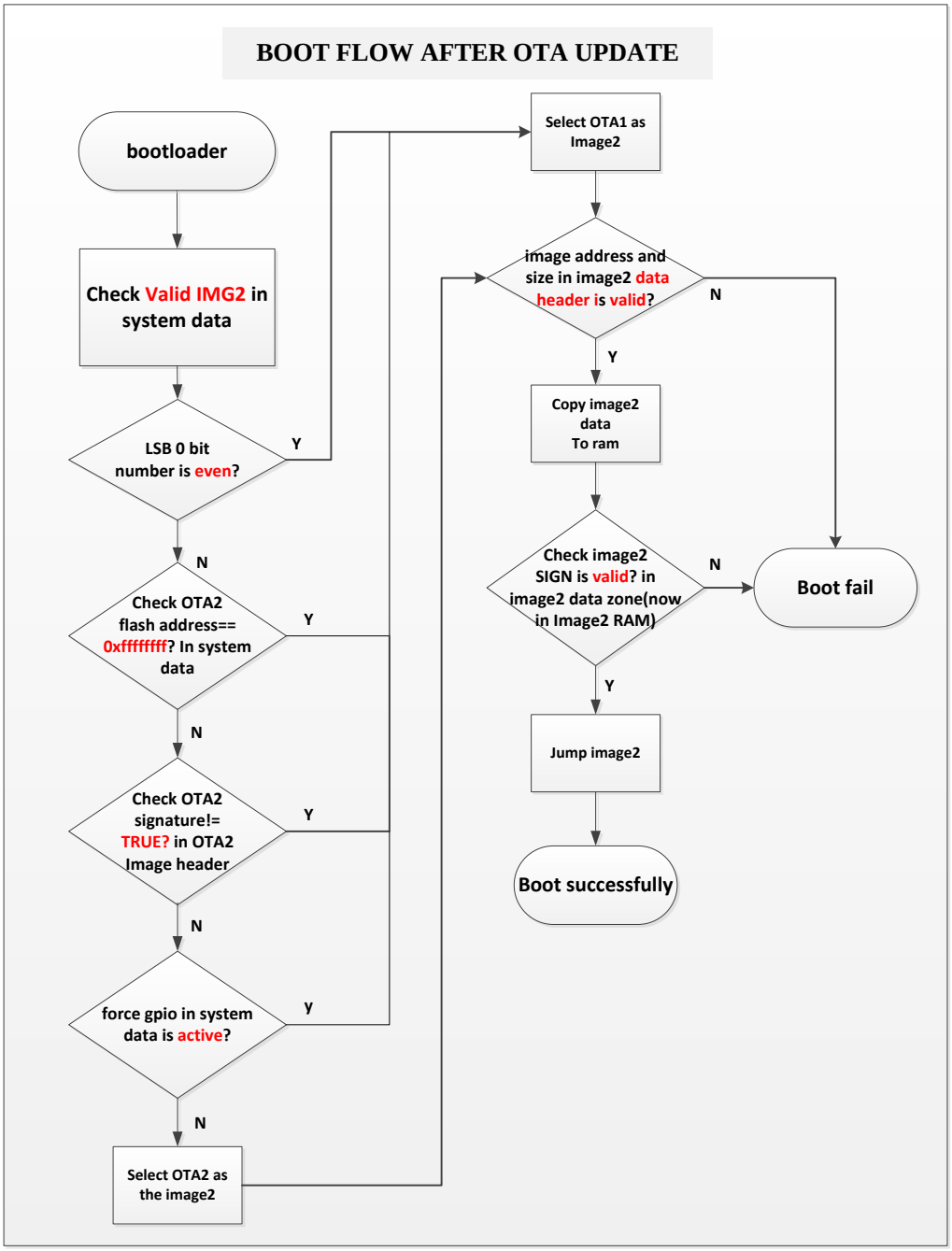
4 Firmware Upgrade over the Air

4.1 Overview

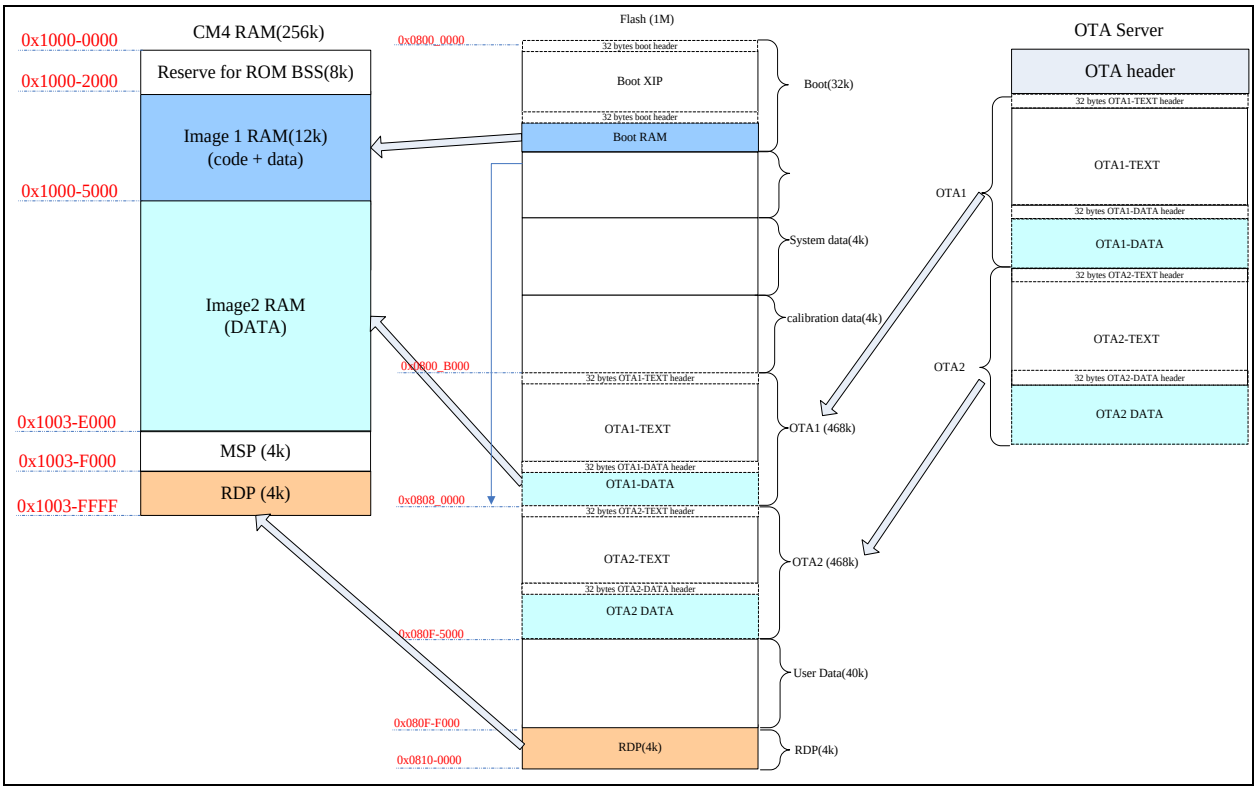
4.1.1 OTA Operation Flow



4.1.2 Boot Process Flow



4.1.3 Upgraded Partition



In AmebaZ OTA update procedure, OTA1-TEXT and OTA2-TEXT is swapped each other.

4.2 Implement OTA over Wi-Fi

4.2.1 OTA Using Local Download Server

The example shows how device updates image from a local download server. The local download server send image to device based on network socket.

Make sure both device and PC are connecting to the same local network.

4.2.2 Build OTA Application Image

Turn on OTA command

```
//project\realtek_amebaz_va0_example\inc\platform_opts.h
#define CONFIG_OTA_UPDATE      1
```

Define server type

// SERVER_TYPE = SERVER_LOCAL in *ota_rtl8710b.c.*

```
//in ota_rtl8710b.c
#define SERVER_LOCAL        1
#define SERVER_CLOUD        2
#define SERVER_TYPE         SERVER_LOCAL
#define  SINGLE_IMG_OTA_UPGRADE   1
```

Acquire the upgraded image2 address and verify this address

```
//in ota_rtl8710b.c
static u32 get_ota_address(u32 ota_target_index, u32 * new_addr, update_ota_target_hdr *
pOtaTgtHdr)
{
        IMAGE_HEADER *OTA1Hdr = NULL;
        uint32_t OTA1Len = 0;
        IMAGE_HEADER *FlashImgDataHdr = NULL;
        uint32_t ota2_addr;
        flash_t flash;

        /*read the OTA2 address from system data zone*/
        ota2_addr = HAL_READ32(SPI_FLASH_BASE, OFFSET_DATA);

        printf("ota2_addr = %x\n", ota2_addr);

        /*if the OTA2 address is not programmed in system data zone, the default OTA2
        address is used. This operation is just used in the local OTA update demo. For the
        cloud OTA upgrade based on this demo, this operation may not be used.*/
        if(ota2_addr == 0xffffffff) {
                ota_write_ota2_addr( &flash, OTA2_DEFAULT_ADDR);
                ota2_addr = HAL_READ32(SPI_FLASH_BASE, OFFSET_DATA);
        }

        if((ota2_addr%4096) != 0) {
                printf("\n\r[%s] ota addr in sys data space not 4k aligned 0x%x",
                 __FUNCTION__, ota2_addr);
                goto error;
        }

        if(ota_target_index == OTA_INDEX_2) {
                /* OAT2 address should not in OTA1 image & should 4K alignment */
```

```
                OTA1Hdr = (IMAGE_HEADER *)(OTA1_ADDR);
                OTA1Len = OTA1Hdr->image_size;
                FlashImgDataHdr =(IMAGE_HEADER*)((u32)OTA1Hdr+OTA1Len+
                                        IMAGE_HEADER_LEN);
                if ((ota2_addr < ((u32)FlashImgDataHdr + FlashImgDataHdr->image_size +
                                        IMAGE_HEADER_LEN)) && ((ota2_addr & 0xfff) == 0)) {
                        printf("\n\r[%s] illegal ota addr 0x%x", __FUNCTION__, ota2_addr);
                        goto error;
                }
                *new_addr = ota2_addr;
        } else {
                *new_addr = OTA1_ADDR;
                /* if OTA1 will be update, image size should not cross OTA2 */
                if(pOtaTgtHdr->FileImgHdr.ImgLen > (ota2_addr - *new_addr)) {
                        printf("\n\r[%s] illegal new image length 0x%x",
                                __FUNCTION__, pOtaTgtHdr->FileImgHdr.ImgLen);
                        goto error;
                }
        }

    /*if the flash offset parameter of the corresponding OTA index in file firmware header
    received from    server is needed in cloud OTA upgrade based on this demo, add code
    here*/
      #if 1
                if((*new_addr) != pOtaTgtHdr->FileImgHdr.FlashOffset){
                        printf("\n\r[%s] pOtaTgtHdr->FileImgHdr.FlashOffset = %p\n",
                                __FUNCTION__, pOtaTgtHdr->FileImgHdr.FlashOffset);
                        /*add code for cloud OTA upgrade use*/
                        #if 1
                                goto error;
                        #endif
                }
      #endif

        if(*new_addr == 0xFFFFFFFF) {
                printf("\n\r[%s] update address is invalid \n", __FUNCTION__);
                goto error;
        }
        return 1;
error:
        return 0;
}
```

Note:
The code below in function **get_ota_address()** is just used in local OTA upgrade demo in case of the system data zone being not programmed. This operation may not be necessary in the cloud OTA upgrade based on this local OTA upgrade demo.

```
        /*if the OTA2 address is not programmed in system data zone, the default OTA2
        address is used. This operation is just used in the local OTA update demo. For the
        cloud OTA upgrade based on this demo, this operation may not be used.*/
        if(ota2_addr == 0xffffffff) {
                ota_write_ota2_addr( &flash, OTA2_DEFAULT_ADDR);
                ota2_addr = HAL_READ32(SPI_FLASH_BASE, OFFSET_DATA);
        }
```

Define custom signature

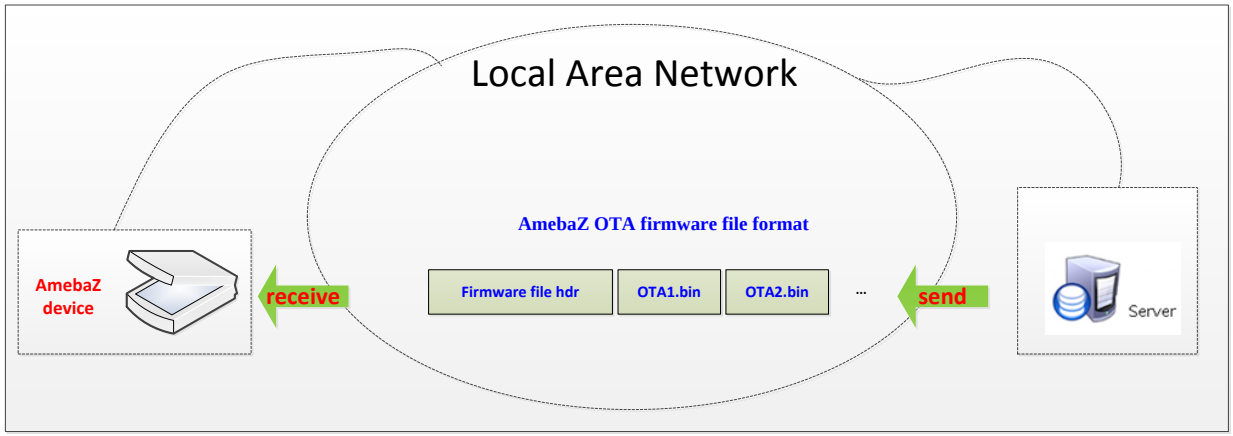
```
//in ota_rtl8710b.c
1. turn on the marco as follows:
#define CONFIG_CUSTOM_SIGNATURE 1
2. Define your own signature.

#if CONFIG_CUSTOM_SIGNATURE
/* -------------------------------------------------
 * Customized Signature
 * This signature can be used to verify the correctness of the image
 * It will be located in fixed location in application image:
 * (after 32B IMG2 header )
 * -------------------------------------------------*/
IMAGE2_CUSTOM_SIGNATURE
const unsigned char cus_sig[32] = "Customer Signature-modelxxx";
#endif 3. compare it while complete flashing.
static u32  verify_ota_checksum (u32 addr, u32 len, u8 * signature,
                                        update_ota_target_hdr * pOtaTgtHdr)
{
…
#if CONFIG_CUSTOM_SIGNATURE
        /* check custom signature */
        _memcpy(read_custom_sig, NewImg2Addr + IMAGE_HEADER_LEN, 32);
        if (strcmp((char*)read_custom_sig, cus_sig)) {
                goto update_ota_exit;
        }
```

```
#endif
...
}
```

5 OTA Upgrade from Local Server

Here is a demo for AmebaZ OTA upgrade from local server.

5.1 Diagram



5.2 Steps for Local OTA Upgrade

These steps can be followed to run this demo:

(1) Before run this demo, please make sure AmebaZ and server are in the same local area network.

(2) Generate image files for OTA upgrade use:
OTA_ALL.bin

(3) Copy the file into the Downloadserver folder
(SDK path: tools \DownloadServer):

(4) Edit start.bat file which is included in the path mentioned in step 3.
The name of the file copied in step 3 should be the same as that in start.bat file.
Modify Port = 8082, file =OTA_ALL.bin

```
@echo off
DownloadServer 8082 OTA_All.bin
set /p DUMMY=Press Enter to Continue ...
```

(5) Click the start.bat to execute Downloadserver program

(6) Reboot AmebaZ device and connect to AP

(7) Enter command: ATWO=IP[PORT].

```
# ATWO=192.168.0.20[8082]
[ATWO]: _AT_WLAN_OTA_UPDATE_

[MEM] After do cmd, available heap 28400


#
[ota_update_local_task] Update task start
```

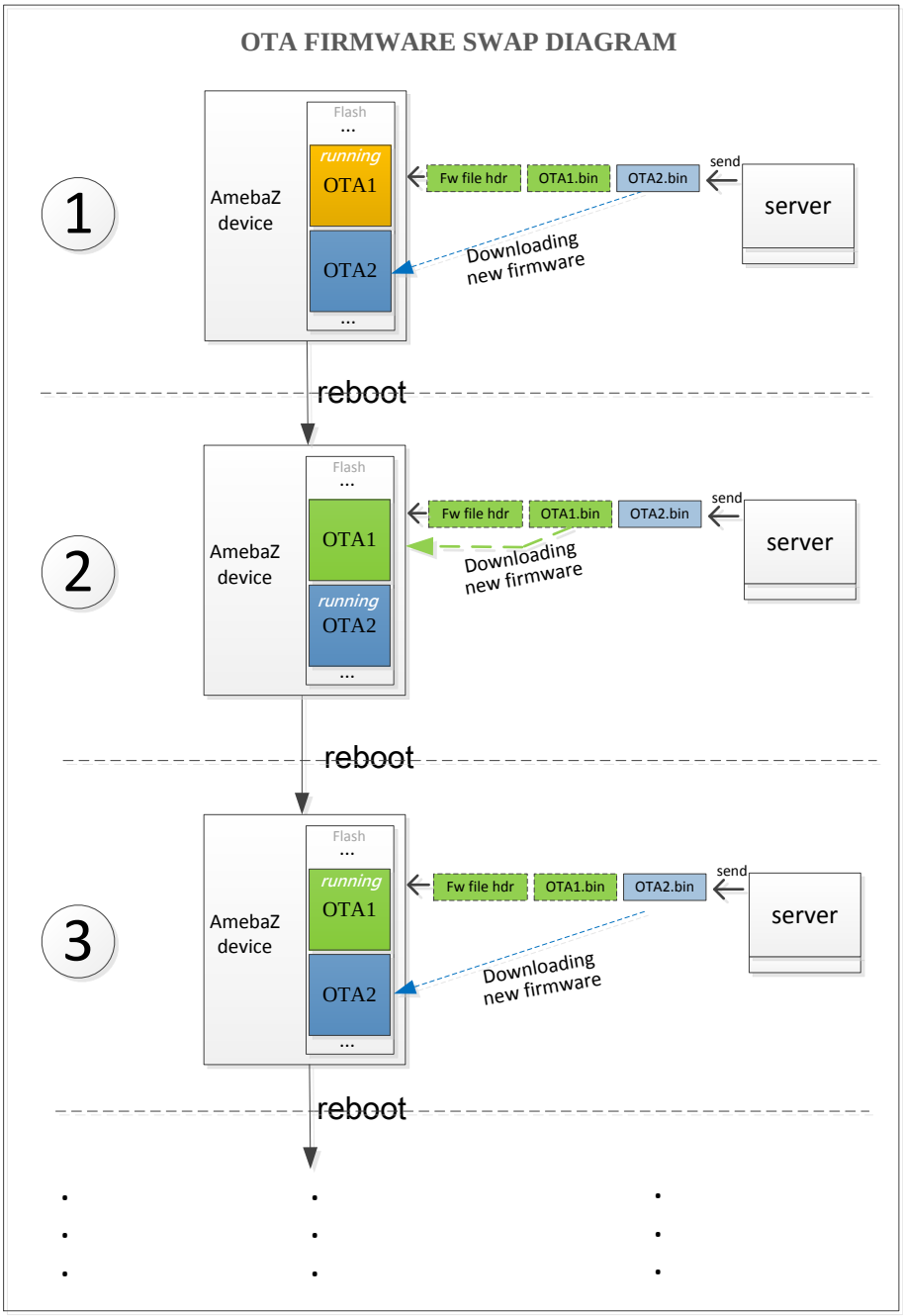
(8) After step 7, OTA upgrade procedure is started between AmebaZ and server. Here is the local download server success message:

```
Listening on port (8082)

Waiting for client ...
Accept client connection from 192.168.1.105
sending OTA_All.bin now...
Sending file...
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................
Total send 672024 bytes
```

(9) After finishing downloading image, Amebaz device will be auto-rebooted, and the bootloader will load new image 2 if it exists.

5.3 OTA Firmware Swap Diagram

Firmware swap diagram in OTA upgrade:



5.4 OTA Signature

To Clear or Recover OTA signature for verification via UART at command, please refer to AN0025.