CC3200 SimpleLink™ Wi-Fi® and Internet-of-Things Solution, a single-chip wireless MCU

CC3200 Over-The-Air (OTA) Update Application Note

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Over The Air (OTA) update is wireless delivery of new software updates and/or configurations to embedded devices and with the concept of **Wireless Sensor Network** and **Internet of Things**, OTA is an efficient way of distributing firmware updates or upgrades.

2 OTA Library Implementation

2.1 System Block Diagram

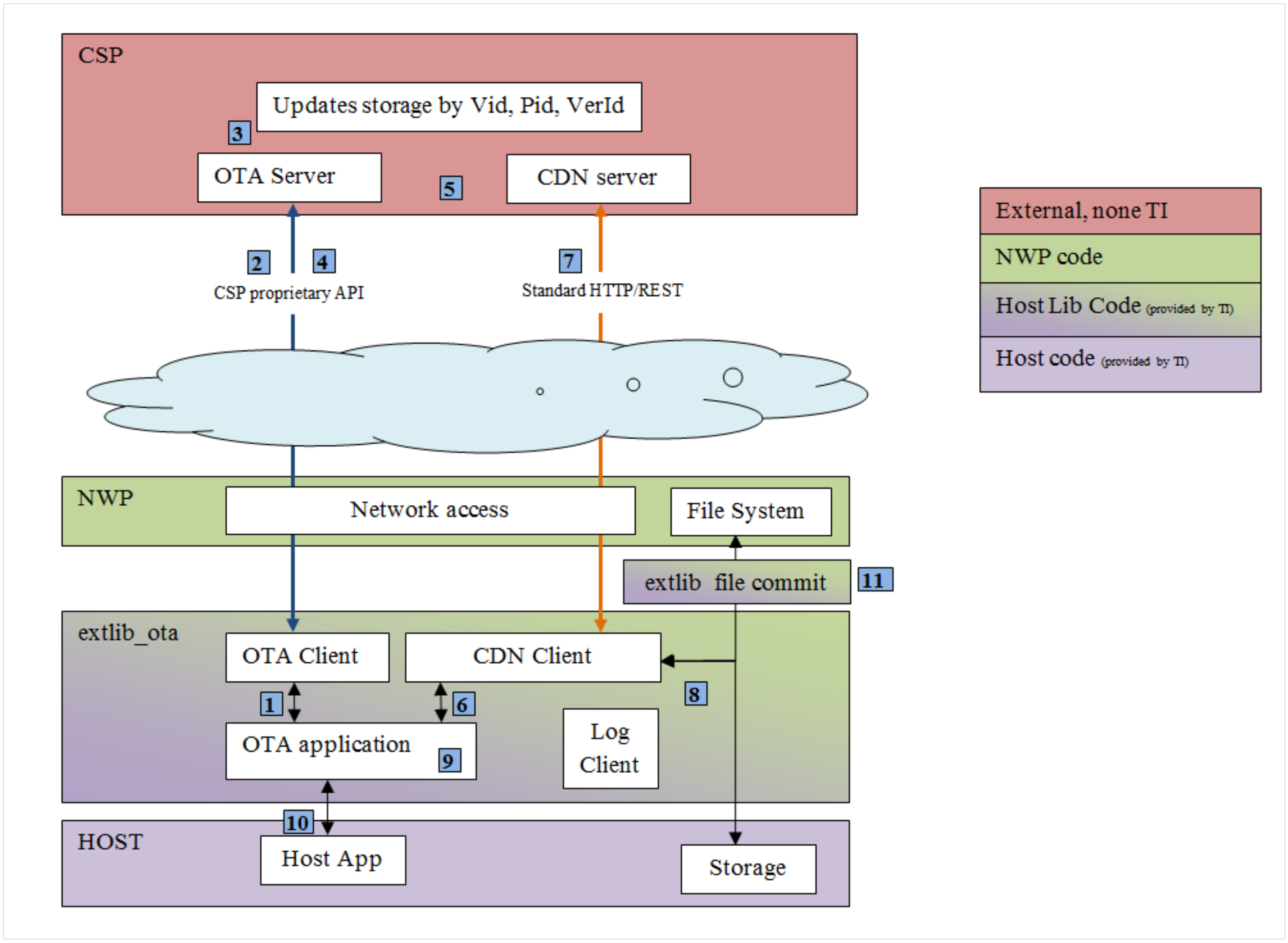


Figure 1 : System Block Diagram

2.2 Module Descriptions

2.2.1 Extlib_ota module

- Connects to the OTA server
- Downloads the list of updates depending on the HOST/NWP version
- Searches for update in the cloud directory /**Vid00_Pid00_VerAAXX**, this enables the vendor to put updates for a specific product and version
- Supports file name pattern: **faa_sys_filename.ext** , where aa is the flag for secured file, signature file, use external storage, reset NWP/MCU
- Downloads all update files and can store it on SFLASH or on an external storage.
- Instructs the host application on how to proceed (Reset MCU, reset NWP, …)
- Supports Non-Os time sharing and FSM - save progress info, return after every step
- Saves statistics into file "/sys/otastat.txt" and also uploads it onto the cloud
- Restrictions
  - File revert is only supported for MCU image file.
  - Uses 2 secured socket (CC3x00 support only 2 secured sockets)
  - Max of 16 files in each update

2.2.2 Extlib_file_commit (FLC) module

- Accesses the SFLASH file system (Open, Read, Write, Close)
- Manages the MCU image commit process:
  - Uses /sys/mcubootinfo.bin file to identify active image (1, 2) and image status (TESTING, TESTREADY, NOTEST).
  - Selects the next image to be updated
  - Allows testing the new image by setting TESTREADY and signaling reboot.
  - Commits the new image when indicated by host application.

2.3 High Level Flow

1. OTA App periodically calls the local OTA client to connect to the OTA server and check for updates.
2. OTA client send "update_check" request with vendor id , ask OTA server for a list of resources
3. OTA Server based vendor id sends back the list to resources to update
4. OTA client send "metadata" request with next resource id, asking for specific resource information
5. OTA server sends back the CDN domain and path to the resource
6. OTA app call CDN Client to download the resource to the File system (or to external storage)
7. CDN client, using HTTP requests, downloads the file in chunks into the File storage
8. Steps from 4 to 8 are repeated until each resource in the list is updated.
9. OTA returns DOWNLOAD_DONE to Host along with reset MCU and/or NWP flag.
10. Host activates the commit process.

2.4 OTA Application State Machine

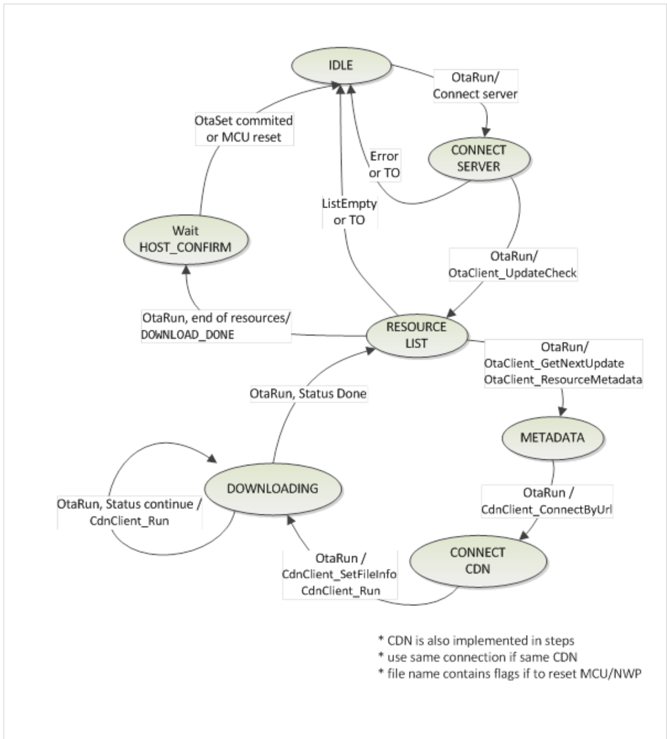


Figure 2 : OTA Application State Machine

2.5 Sequence Diagrams

2.5.1 OTA client/server sequence

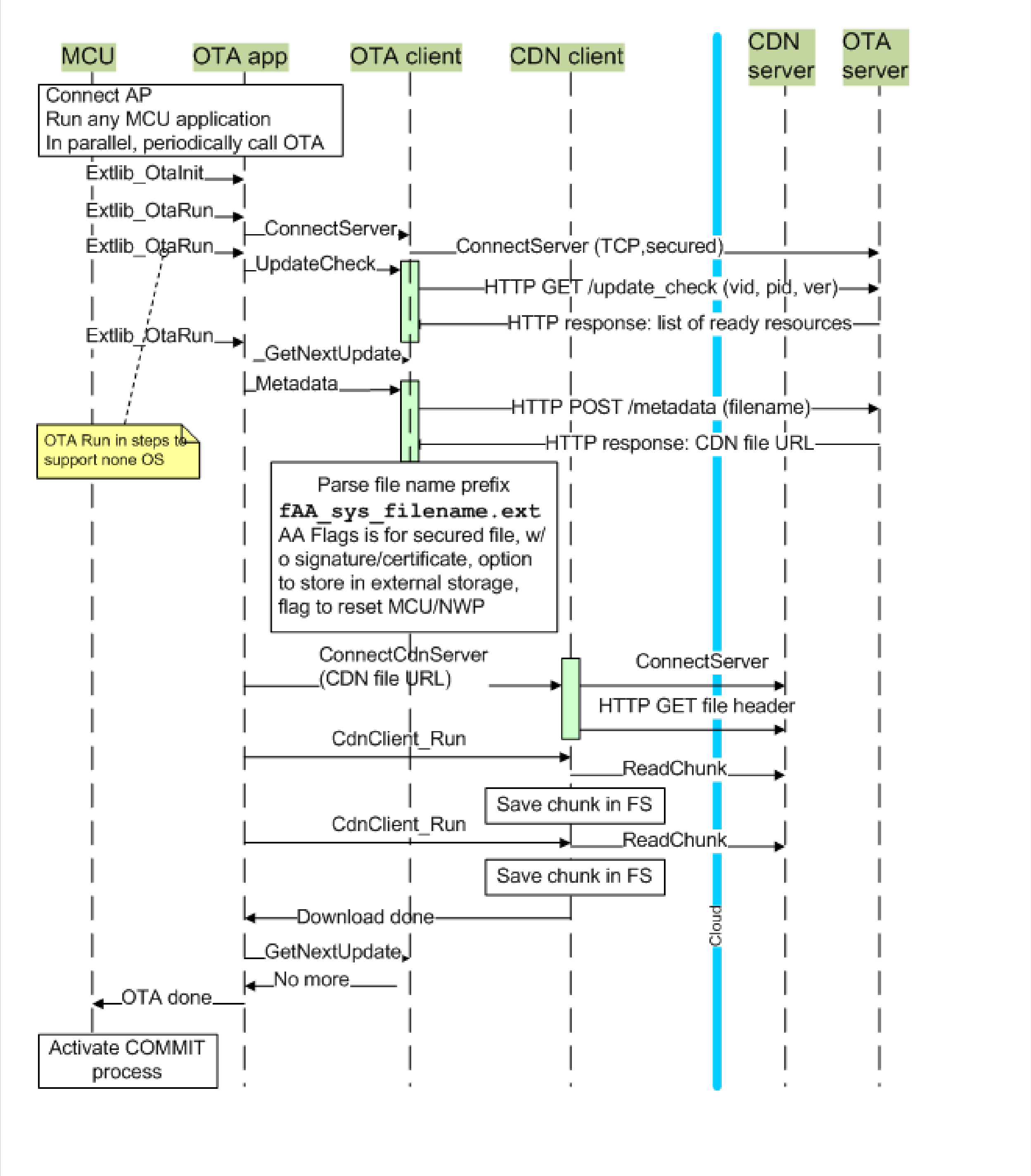


Figure 3 : OTA client/server sequence

2.5.2 OTA and MCU Commit sequence – Success

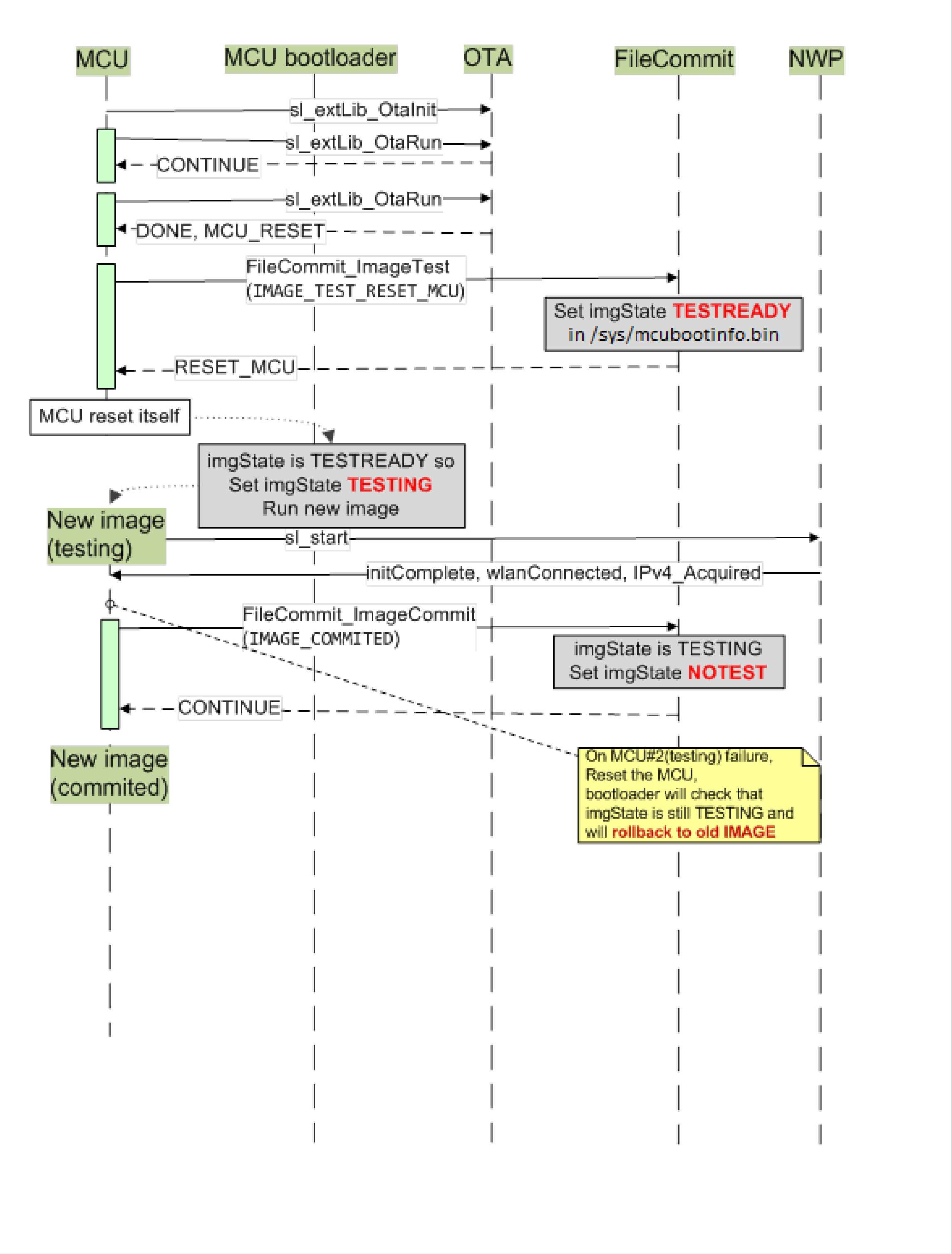


Figure 4 : OTA and MCU Commit sequence - success

Features characteristic data and other information are subject to change.

2.5.3 OTA and MCU Commit sequence – Rollback

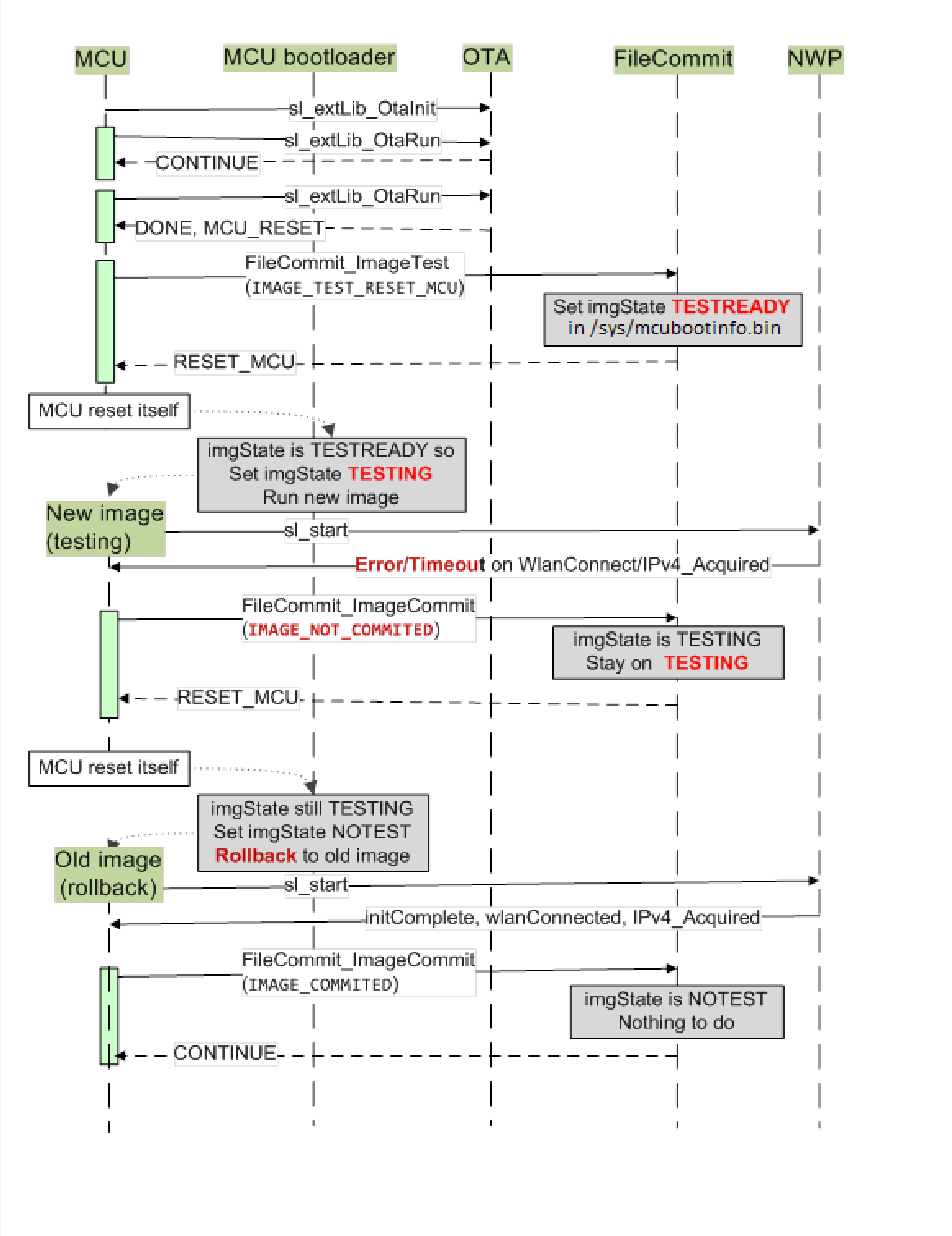


Figure 5 : OTA and MCU Commit sequence – rollback

3 Example OTA Update Application

This application focuses on showcasing CC3200's ability to receive firmware update and/or any related files over the internet enabled Wi-Fi interface. The example uses Dropbox API App platform to store and distribute the OTA update files.

An APP on Dropbox API platform can be looked at as a network accessible drive where user contents are arranged as a tree of files and/or folders. The OTA library expects a folder at the top level which is pointed to via VendorString (the folder name on Dropbox), set during OTA initialization. This top level folder should contain the files to be updated directly and no folders. The OTA library also puts some restriction on the file names (see File Naming Convention for OTA on Dropbox section). File(s) with other name pattern will be rejected.

The VendorString can be constructed in a variety of ways and as an example this application constructs it by appending the Service Pack version to a macro **OTA_VENDOR_STRING** defined in otaconfig.h file.

Assuming the current service pack running on the device holds the version number **2.1.0.87.31.0.0.4.1.1.5.3.3** and **OTA_VENDOR_STRING** is defined as **Vid01_Pid00_Ver01,** the application constructs the VendorString by appending the 4th byte (shown in red) to the macro i.e. **Vid01_Pid00_Ver0187.** This folder on Dropbox should contain all the files that need to be updated. If left empty OTA library assumes a NO_UPDATE condition.

3.1 Source Files briefly explained

- NON-OS – Directory holding non-os based implementation of the application
    - main.c - Contains the core logic for the application
    - net.c - Wrapper function implementation for required SL_HOST APIs
    - otaconfig.h - Contains OTA server configuration details
    - pinmux.c - Auto generated file pin muxing on CC3200 LP boards
    - task.c - Implements non-os tasking

- OS – Directory holding os based implementation of the application
    - main.c - Contains the core logic for the application
    - net.c - Wrapper function implementation for required SL_HOST APIs
    - otaconfig.h - Contains OTA server configuration details
    - pinmux.c - Auto generated file pin muxing on CC3200 LP boards

3.2 Usage

3.2.1 Setting up OTA Application for LP with CC3200 SDK

3.2.1.1 Basic Setup

1. Copy simplelink_extLib directory into the root directory of the SDK, for example: C:\ti\CC3200SDK\cc3200-sdk
2. Copy “ota_update” and “application_bootloader” directory into the examples directory, for example: C:\ti\CC3200SDK\cc3200-sdk\example.
3. Create and setup Dropbox account (see sections below).

3.2.1.2 Flashing

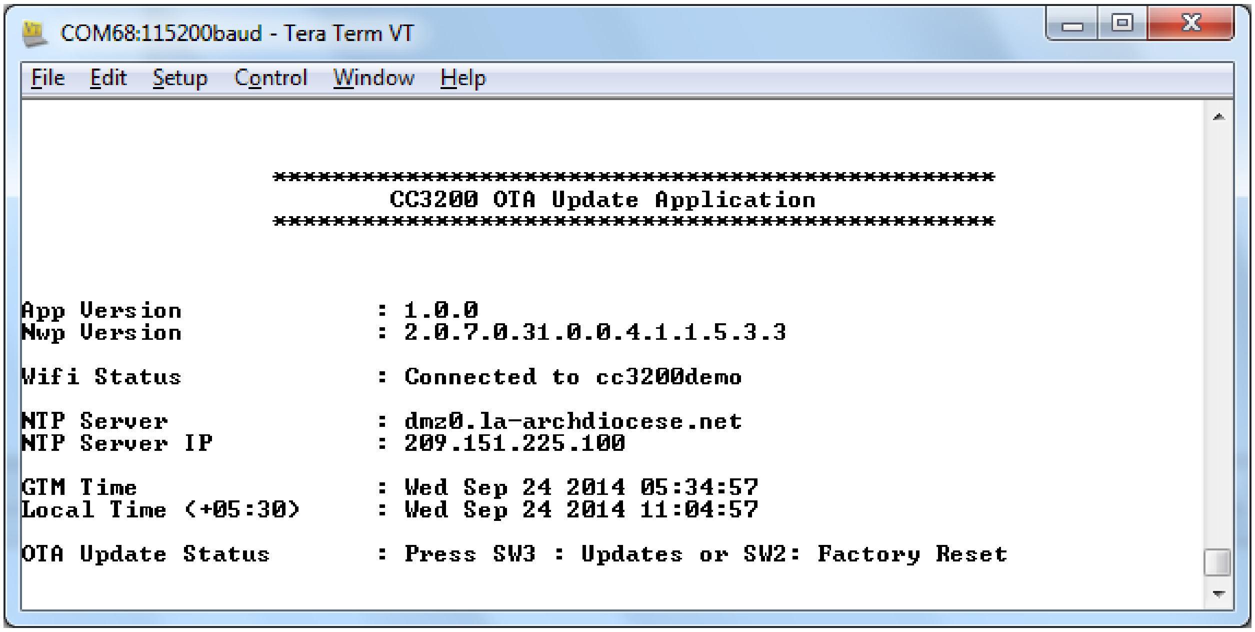
1. Open Uniflash tool for CC3xxx
2. Make sure LP is in UARTLOAD mode i.e. **SOP2** jumper mounted.
3. Format the sFlash.
4. Program the service pack.
5. Go to File->New Configuration. Choose the connection and board and click ok
6. Set “/sys/mcuimg.bin” URL to point “application_bootloader.bin”
7. Select “Erase”, “Update” and “Verify”
8. Add a new file and rename it to “/sys/mcuimg1.bin”. If no factory default is required rename it to “/sys/mcuimg2.bin”. See “application_bootloader” documentation for more details on file names.
9. Set the COM port to LP’s com port number
10. Press Program

3.2.1.3 Running

1. Remove SOP2 jumper
2. Connect to COM port via Tera-term or Hyper Terminal with following configuration
   a. **Port**: Enumerated COM port
   b. **Baud rate**: 115200
   c. **Data**: 8 bit
   d. **Parity**: None
   e. **Stop**: 1 bit
   f. **Flow control**: None
3. Press reset.

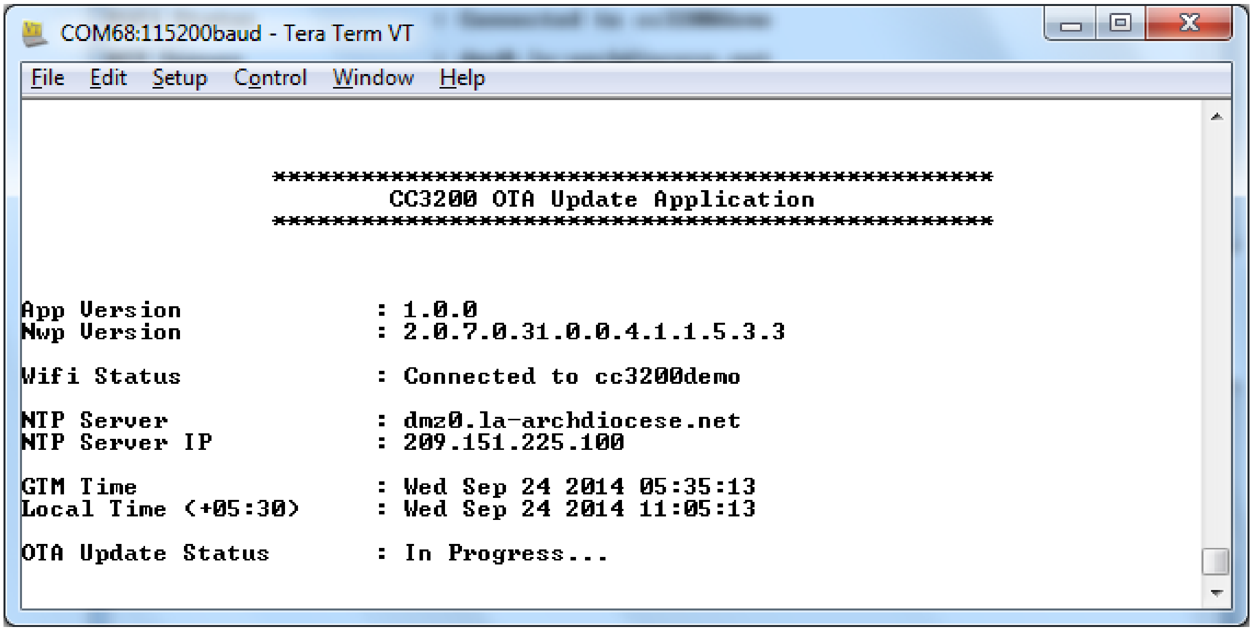
3.2.2 Triggering OTA Update

1. Wait for the Application to completely boot. Observer the app version


COM68:115200baud - Tera Term VT
File Edit Setup Control Window Help

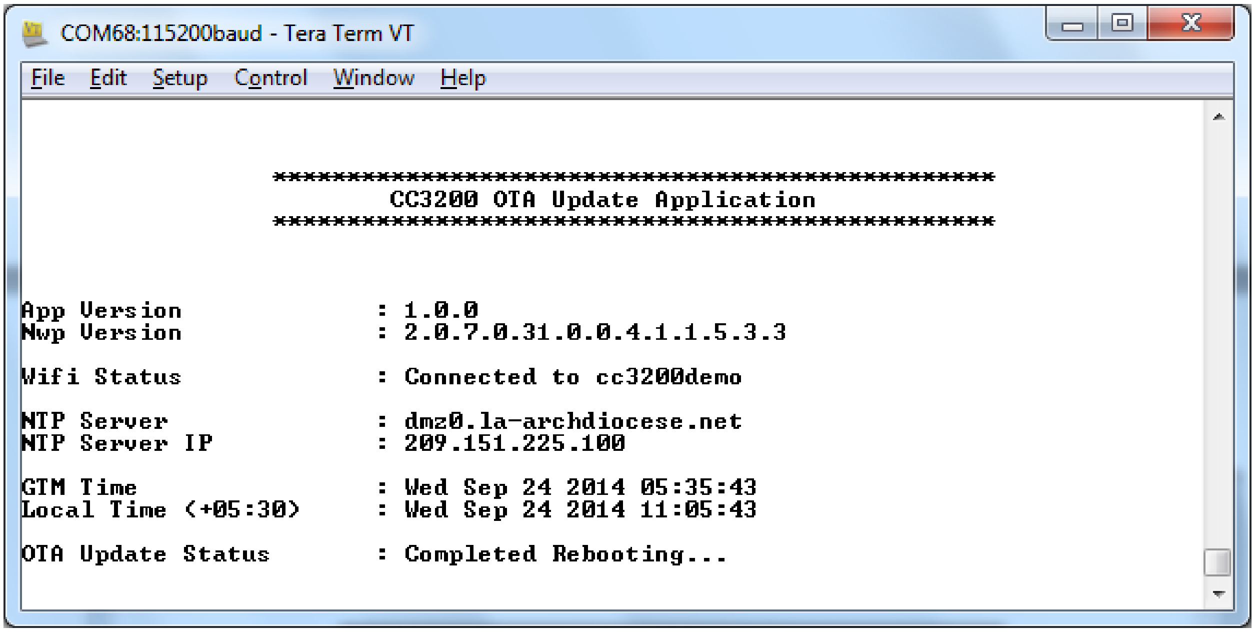
```
                 *************************************************
                         CC3200 OTA Update Application
                 *************************************************


App Version                  : 1.0.0
Nwp Version                  : 2.0.7.0.31.0.0.4.1.1.5.3.3

Wifi Status                  : Connected to cc3200demo

NTP Server                   : dmz0.la-archdiocese.net
NTP Server IP                : 209.151.225.100

GTM Time                     : Wed Sep 24 2014 05:34:57
Local Time (+05:30)          : Wed Sep 24 2014 11:04:57

OTA Update Status            : Press SW3 : Updates or SW2: Factory Reset
```

2. Press SW3 on LP to start the update. Observer the “OTA Update Status” is now “In Progress”


COM68:115200baud - Tera Term VT
File Edit Setup Control Window Help

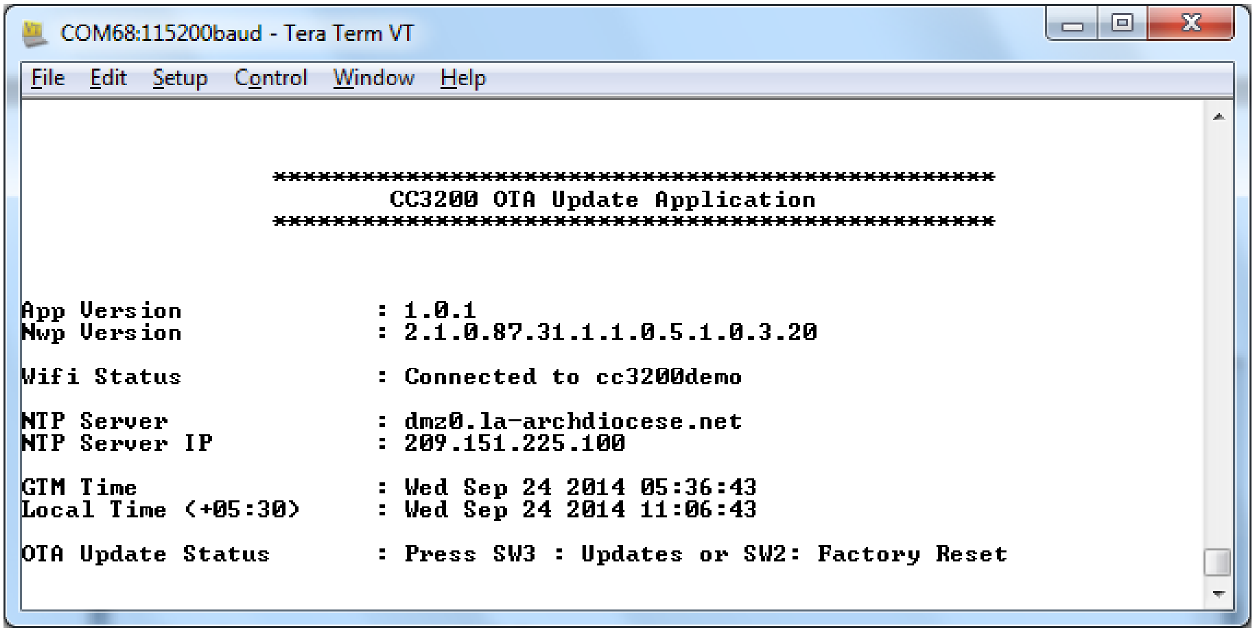
```
                 *************************************************
                         CC3200 OTA Update Application
                 *************************************************


App Version                  : 1.0.0
Nwp Version                  : 2.0.7.0.31.0.0.4.1.1.5.3.3

Wifi Status                  : Connected to cc3200demo

NTP Server                   : dmz0.la-archdiocese.net
NTP Server IP                : 209.151.225.100

GTM Time                     : Wed Sep 24 2014 05:35:13
Local Time (+05:30)          : Wed Sep 24 2014 11:05:13

OTA Update Status            : In Progress...
```

3. Wait for OTA to complete. Application will reboot itself


COM68:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
                 ************************************************
                         CC3200 OTA Update Application
                 ************************************************


App Version                : 1.0.0
Nwp Version                : 2.0.7.0.31.0.0.4.1.1.5.3.3

Wifi Status                : Connected to cc3200demo

NTP Server                 : dmz0.la-archdiocese.net
NTP Server IP              : 209.151.225.100

GTM Time                   : Wed Sep 24 2014 05:35:43
Local Time (+05:30)        : Wed Sep 24 2014 11:05:43

OTA Update Status          : Completed Rebooting...
```

4. Observer App version after reboot


COM68:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
                 ************************************************
                         CC3200 OTA Update Application
                 ************************************************


App Version                : 1.0.1
Nwp Version                : 2.1.0.87.31.1.1.0.5.1.0.3.20

Wifi Status                : Connected to cc3200demo

NTP Server                 : dmz0.la-archdiocese.net
NTP Server IP              : 209.151.225.100

GTM Time                   : Wed Sep 24 2014 05:36:43
Local Time (+05:30)        : Wed Sep 24 2014 11:06:43

OTA Update Status          : Press SW3 : Updates or SW2: Factory Reset
```

Note: Nwp version will be different from one show based on the service packs used.

3.3 Creating Dropbox API application

1. Create an account with Dropbox and login
2. Go to https://www.dropbox.com/developers/apps/create and choose “Dropbox API app”
3. Choose “Files and Datastores” and “Yes My app only needs access to files it creates”.
4. Provide a suitable name for the APP and click “Create APP” button
5. You will be redirected to Apps setting page. Scroll down to “**Generated access token**” and click generate. Copy and save the generated token.
6. Go to https://www.dropbox.com/home/Apps
7. Click on the application name
8. Create a new folder and name it “Vid01_Pid00_Ver00xx”. See below for more details on file naming. This folder will contain only Service pack of version yy.
9. Create another folder and name it “Vid01_Pid00_Ver00yy”. This folder will contain Application and other files corresponding to App version 01.
10. Create another folder and name it “Vid01_Pid00_Ver01yy”. This folder will contain only Service pack of version ZZ.
11. The rationale behind steps 8 to 10 is to ensure that Service pack gets updated first and then other files like MCU image are updated. To start with device has Application files of version 00 and service pack of version xx. In order to update the device to Application files of version 01 and service pack of version yy, copy new service pack yy into the folder “Vid01_Pid00_Ver00xx” and all other application files corresponding to version 01 into the folder “Vid01_Pid00_Ver00yy”. After completion of the update Application will start to point to “Vid01_Pid00_Ver01yy”

Note: xx represents the 4th byte of the service Pack version number currently running on the device and yy represents the 4th byte of the service pack contained in Vid01_Pid00_Ver00xx folder, xx can be equal to yy if there is no service pack update
Service Pack version sample: 2.1.0.**87**.31.0.0.4.1.1.5.3.3

3.4 Configuring the application for new Dropbox account

1. Open otaconfig.h
2. Update the following Parameters
   a. OTA_SERVER_REST_HDR_VAL - Set this to the token generated in the previous steps
   b. OTA_VENDOR_STRING - Set this to Vid01_Pid00_Ver01
   c. APP_VER_BUILD - 1
3. Compile and upload the .bin file into “Vid01_Pid00_Ver00xx” folder on Dropbox server
4. Rename it “f80_sys_mcuimgA.bin”.

5. Update the following parameters again
    a. OTA_VENDOR_STRING - Set this to Vid01_Pid00_Ver00
    b. APP_VER_BUILD - 0

6. Compile and upload the .bin file into “Vid01_Pid00_Ver01yy” folder on Dropbox server
7. Rename it “f80_sys_mcuimgA.bin”.

3.5 File Naming Convention for OTA on Dropbox

The files stored on the cloud should be in the following format

/VidVV_PidPP_VerRRXX/fAA_sys_filename.ext

The directory /VidVV_PidPP_VerRRXX

VidVV	– Vendor id number
PidPP	– Product id number
RR	– Application version
XX	– Service Pack version

The filename fAA_sys_filename.ext

- fAA – File Flags
  - f - File prefix
  - AA - File flags bitmap :
    - 01 - The file is secured
    - 02 - The file is secured with signature
    - 04 - The file is secured with certificate
    - 08 - Don't convert _sys_ into /sys/ for SFLASH
    - 10 - Use external storage instead of SFLASH
    - 20 - Reserved.
    - 40 - NWP should be reset after this download
    - 80 - MCU should be reset after this download

- sys optional and can be converted to /sys/ directory
- ext
    - signature - .sig, filename must be the name of the secured file
    - certificate - .cer, filename must be the name of the secured file

For example: Vid01_Pid33_Ver18/f43_sys_servicepack.ucf is for vendor id 01, product id 33, version 18 and secured file /sys/servicepack.ucf

Following table list the file names of fixed know image types:

Table 1 : OTA File Name

Image Type	OTA File Name
User Application binary	f80_sys_mcuimgA.bin
Service Pack	f43_sys_servicepack.ucf
Service Pack signature	f00_sys_servicepack.sig

4 Example Application Bootloader for OTA

CC3200 ROM boot-loader supports loading and executing one application image from sFlash storage. This application can be a secondary boot-loader (or application boot-loader) that can manage, load and execute more binary images from the sFlash storage. Such a use case would occur if the end application requires more than one executable binary and/or requires some extended features like roll-back in case of an update crash, where the secondary boot-loader keep track of best known image.

This application focuses on showcasing a secondary boot-loader with a roll-back option. This implementation can manage up to three bootable images:

1. Factory Default (/sys/mcuimg1.bin) : A Factory default image
2. App Image 1 (/sys/mcuimg2.bin) : Application image
3. App Image 2 (/sys/mcuimg3.bin) : Application image

Application boot loader has to be programmed to SFLASH with the same file name as used to program standalone application i.e. "/sys/mcuimg.bin” and is responsible for selecting one of the application images and loading into RAM for execution.

The boot-loader on execute will auto relocate to the SRAM base area “0x20000000 to 0x20003FFF” before loading the application images; this ensures that actual application can still start at 0x20004000.

Application boot loader assists OTA update run time logic in testing, installation and if needed rollback of the newly downloaded image. It loads application images from SFLASH using simple link NVMEM API’s.

This implementation uses one additional boot info file to keep track of the latest and best known image, see “Application image selection and rollback logic” section for more details.

If the bootinfo file doesn't exists on the file system it is auto generated by this bootloader by scanning the file system for bootable images in the following order, if none exists, the booting is aborted:

1. Factory Default
2. App Image 1
3. App Image 2

If user wants to use all the three images, then delete the boot info file, if it exists, and flash the first application binary as /sys/mcuimg1.bin

In case user wants to use only 2 images (no Factory default) then delete the boot info file, if it exists, and flash the first application binary as /sys/mcuimg2.bin

See "Application image selection and rollback logic" section to understand the relation between the images.

4.1 Application Bootloader Memory Map

The application boot-loader is a concatenation of two binaries: A relocator and a boot manager. Following figure show the structure.

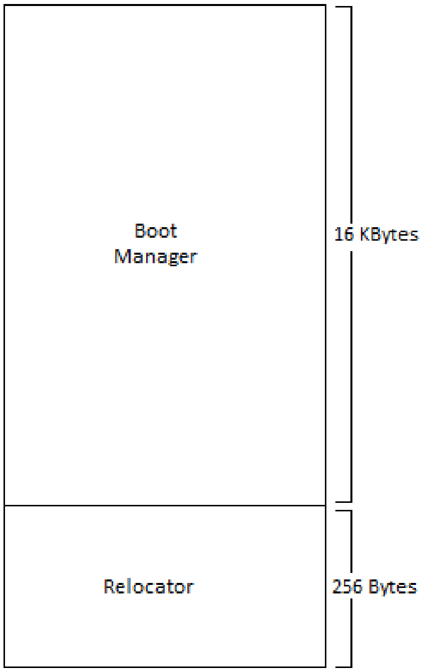


Figure 6 : Application Bootloader Memory Map

4.2 Application Image Selection and Rollback Logic

1. Read the file /sys/mcubootinfo.bin. This file has the below info

Table 2 : Application image selection and rollback logic

ACTIVE_IMAGE_FILE_NAME (sBootInfo_t.ucActiveImg)	File name of current active image, This can have the values 0 - /sys/mcuimg1 – Factory default image 1 - /sys/mcuimg2 2 - /sys/mcuimg3
OTAU_MCU_IMAGE_STATE (sBootInfo_t.ulImgStatus)	Status of the OTAU image 0xABCDDCBA - IMG_STATUS_NOTEST 0x56788765 - IMG_STATUS_TESTREADY 0x12344321 - IMG_STATUS_TESTING

If this file does not exit, the application bootloader creates it based on following priority order:

a. If /sys/mcuimg1 exists, active image is set to this else
b. If /sys/mcuimg2 exists, active image is set to this else
c. If /sys/mcuimg3 exists active image is set to this else
d. Booting is aborted

2. Reads the parameter "**ACTIVE_IMAGE_FILE_NAME**" and "**OTAU_MCU_IMAGE_STATE**" from this file /sys/mcubootinfo.bin.

3. If OTAU_MCU_IMAGE_STATE is equal to **IMG_STATUS_NOTEST**. Launch the contents of the file "**ACTIVE_IMAGE_FILE_NAME**" for execution.

4. If OTAU_MCU_IMAGE_STATE is equal to **IMG_STATUS_TESTING**, It means image previously tested has not been committed. Set the parameter "**OTAU_MCU_IMAGE_STATE**" to "**IMG_STATUS_NOTEST**" and launch the contents of the file "**ACTIVE_IMAGE_FILE_NAME**" for execution. This step essentially ensures rollback to previous image in case of new image test failure.

5. If **OTAU_MCU_IMAGE_STATE** is equal to **IMG_STATUS_TESTREADY**. Set the parameter "**OTAU_MCU_IMAGE_STATE**" to **"IMG_STATUS_TESTING**" and launch the contents of the file "NON_ACTIVE_IMAGE_FILE_NAME" for execution. **ACTIVE_IMAGE_FILE_NAME** and **NON_ACTIVE_IMAGE_FILE_NAME** would share the below relationship

Table 3 : Active and In-Active Images

Active Image	In Active Image
/sys/mcuimg1	/sys/mcuimg2
/sys/mcuimg2	/sys/mcuimg3
/sys/mcuimg3	/sys/mcuimg2

4.3 Source Files briefly explained

main.c	: Contains the core logic for application bootloader
startup_*.c	: Start up file for respective IDEs
udma_if.c	: UDMA driver wrapper API implementation

4.4 Usage

This application can be compiled using CCS and IAR. Project files are provides for each in their directories.

This application uses a “Tiny” version of the “simplelink” library. The following section list the steps for compiling this application

Note: The relocator can only be built on IAR. For other environment(s) use the pre-built binary provided

IAR

1. Open the simplelink work space from cc3200-sdk\simplelink\ewarm\simplelink.eww
2. Open Project->Edit Configurations and click “New”.
3. Name the new configuration as “NON_OS_TINY”
4. Set “Based on Configuration” to “NON_OS” and click ok.
5. Keeping “NON_OS_TINY” selected click ok again.
6. Go to project->Options->C/C++ Compiler->Optimizations->Level and set it to “Medium”
7. Go to preprocessor tab. Change macro from “SL_FULL” to “SL_TINY”.
8. Go to project->options->Library Builder and make sure the output file path is set to “$PROJ_DIR$\NON_OS_TINY\Exe\simplelink.a”.
9. Rebuild the project configuration.
10. Open cc3200-sdk\example\application_bootloader\ewarm\bootmgr.eww
11. Rebuild the project. This will generate application_bootloader\ewarm\application_bootloader.bin

CCS

1. Import the simple link project to CCS workspace
2. Right click the project. Go to Build Configurations->Manage and click New.
3. Name the new configuration as “NON_OS_TINY”
4. Set to copy setting from to “NON_OS” and click ok.
5. Right click the project. Go to Build Configurations->Set Active and select NON_OS_TINY
6. Rebuild the project
7. Import “bootmgr” from cc3200-sdk\example\application_bootloader\ccs
8. Rebuild the project.

See “Example OTA Update Application” section for details to use this bootloader along with OTA Application.

5 Limitations/Known Issues

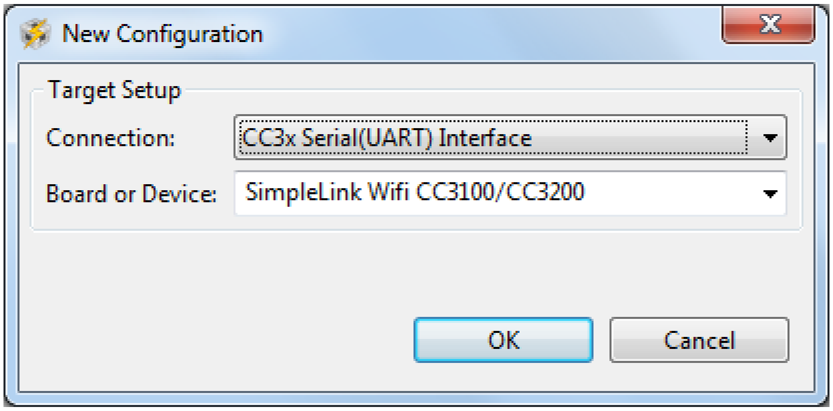
1. OTA cannot update an image to a newer image of same name if the size of the newer image is larger than the max size allocated to that image.
2. File revert is only supported for MCU image file.

6 Appendix

6.1 Generic Guidelines for Enabling OTA on CC3100/CC3200

This section list down the recommended order of flashing, using UniFlash tool, to enable OTA on CC3100/CC3200 device(s):

1. Open UniFlash tool for CC3200/CC3100.
2. Go to File->New Configuration and press ok with following option



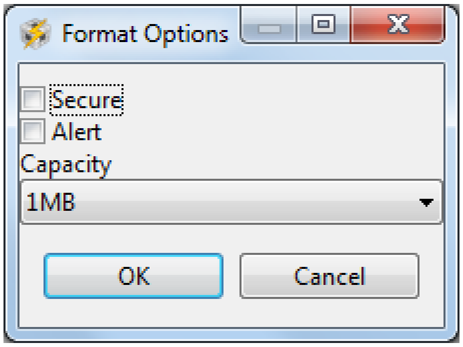
New Configuration

Target Setup

Connection: CC3x Serial(UART) Interface

Board or Device: SimpleLink Wifi CC3100/CC3200

3. Format the storage with “Secure” and “Alert” unchecked.



Format Options

□ Secure
□ Alert

Capacity: 1MB

4. Update the service pack.
5. Create a new file in the sessions file list using “Add File” option and rename it to “/sys/mcuimg1.bin”.
6. Create a new file in the sessions file list using “Add File” option and rename it to “/sys/mcubootinfo.bin”.

7. Set the following options for different files in the session

Table 4 : UniFlash File Options

File	URL	Options
/sys/mcuimg.bin	Application bootloader	Erase, Update, Verify
/sys/mcuimg1.bin	1st OTA Image	Erase, Update, Verify
/sys/mcubootinfo.bin	-	Erase

Note: For "/sys/mcuimg.bin", "/sys/mcuimg1.bin" or any other user file set the "MaxSize" to maximum size that will ever be required. OTA will fail to download any image with the same name and size greater that it's max size set during first time creation via UniFlash.

8. Save the configuration file
9. Press "Program" to program the file onto the device storage.

Refer to http://processors.wiki.ti.com/index.php/CC31xx_%26_CC32xx_UniFlash_Quick_Start_Guide for complete UniFlash quick start guide.

6.2 Porting OTA Library to other servers

Key Macros/Functions	For Dropbox	Descriptions
OTA_SERVER_NAME	api.dropbox.com	The server/domain name
OTA_SERVER_SECURED	1	If to use secure sockets
OTA_SERVER_REST_UPDATE_CHK	/1/metadata/auto/	REST API to get resource list
OTA_SERVER_REST_RSRC_METADATA	/1/media/auto	REST API to resource details
OTA_SERVER_REST_HDR	Authorization: Bearer	Authorization header
OTA_SERVER_REST_HDR_VAL		Authorization header value
LOG_SERVER_NAME	api-content.dropbox.com	Log server
OTA_SERVER_REST_FILES_PUT	/1/files_put/auto/	REST API to write files
OtaClient_UpdateCheck	http_build_request	Get the resource list
	json_parse_dropbox_metadata	
OtaClient_ResourceMetadata	http_build_request	Get per-resource details
	json_parse_dropbox_media_url	

This section lists down and describes the key parameters and functions that are server specific and are required to be re-implemented to port this library to a new server:

6.2.1 Server Info Structure

This structure holds the server related parameter like the domain name, authorization key, REST APIs, log server and vendor string. Following member variables are required to be initialized and passed to OTA Library as part of initialization.
server_domain: This holds the server name for the OTA server.
Eg: api.dropbox.com for Dropbox REST APIs

secured_connection: This holds if the connection to the OTA server and CDN server is secure or non-secure.

rest_update_chk: This defines the REST API for getting the list of resources from the server
Eg: /1/metadata/auto/

rest_rsrc_metadata: This defines the API for getting the details of each resource on the server
Eg: /1/media/auto

rest_hdr: This holds the additional HTTP headers (like authorization) for the server
Eg: Authorization: Bearer

rest_hdr_val: Holds the header value, like the access key.
Eg: BwPuaYu9AoAAABBBAAAAA-uhCfuTU_Jw54oBVgBCtZaMAsDfhTZcV8lLK7ruzD51r

log_server_name: Server name for logging the OTA logs
Eg: api-content.dropbox.com

rest_files_put: This holds the REST API for writing to the server
Eg: /1/files_put/auto/

log_mac_address: MAC address of the current device using the OTA library. This is used for logging

6.2.2 OTA Client Functions

OtaClient_UpdateCheck:

This function gets the list of updates from the OTA server. Internally, sends the **rest_update_chk** request and parses out the response to get the list of resources (files) available.

In case of Dropbox, json_parse_dropbox_metadata, parses the response.

OtaClient_ResourceMetadata

This function gets the details of requested resource including the resource path on CDN client and resource flags. Internally uses **rest_rsrc_metadata** to get the resource details

In case of Dropbox, json_parse_dropbox_media_url, parser the response.