CC3100 Over-The-Air (OTA) Update

Application Note

Table of Contents

List of Figures

List of Tables

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.

OTA Library Implementation

System Block Diagram

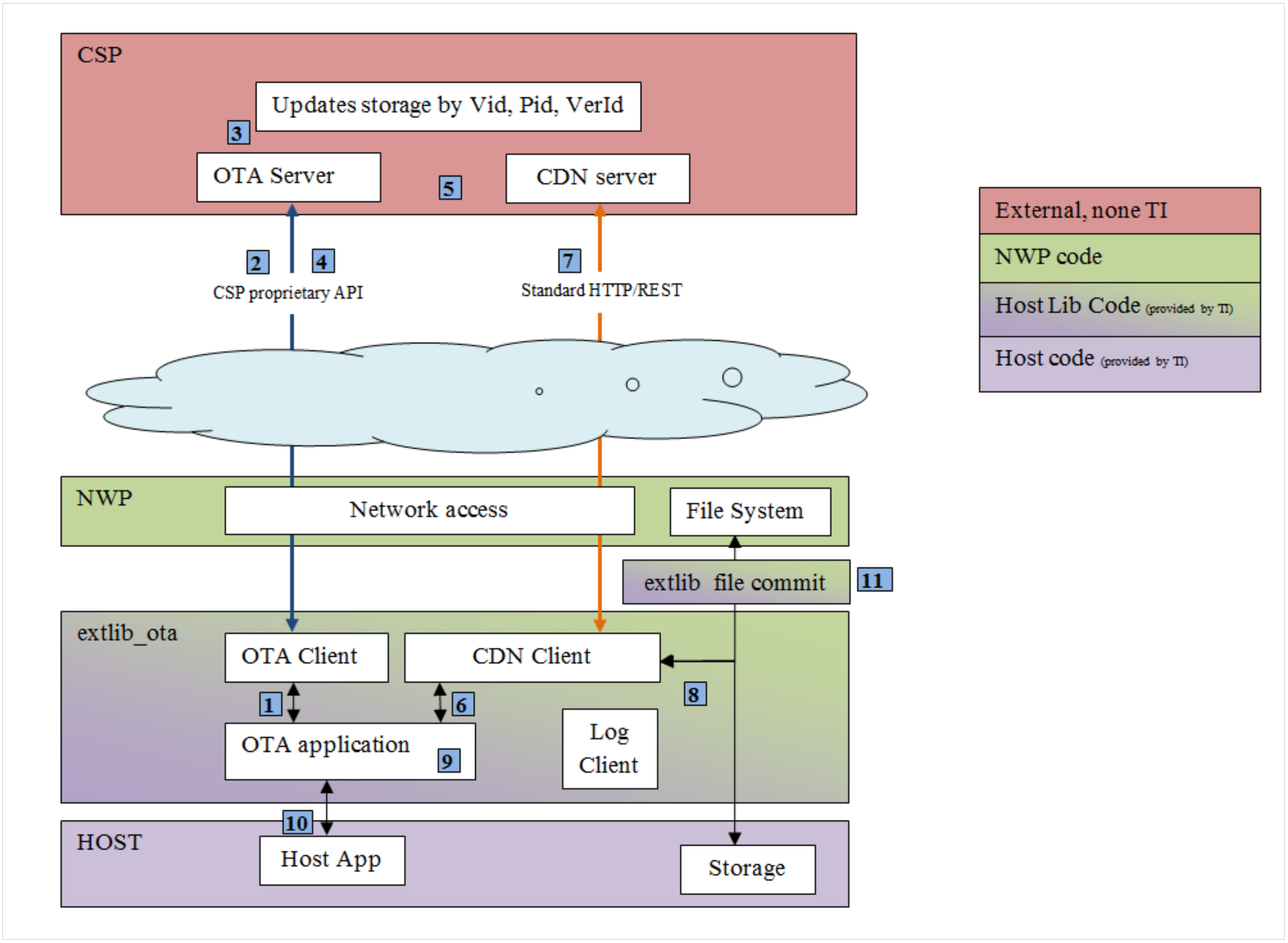


Figure 1 : System Block Diagram

Module Descriptions

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: **f*aa*_sys_filename.ext** , where *aa* is the flag for secured file, signature file, use external storage, reset NWP. Refer to 'File Naming Convention' section for details.
- 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
  - Uses 2 secured socket (CC3x00 support only 2 secured sockets)
  - Max of 16 files in each update

Extlib_file_commit (FLC) module

- Accesses the SFLASH file system (Open, Read, Write, Close)
- Manages the MCU image commit process (Valid for CC3200 only):
  - 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.

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.

OTA Application State Machine

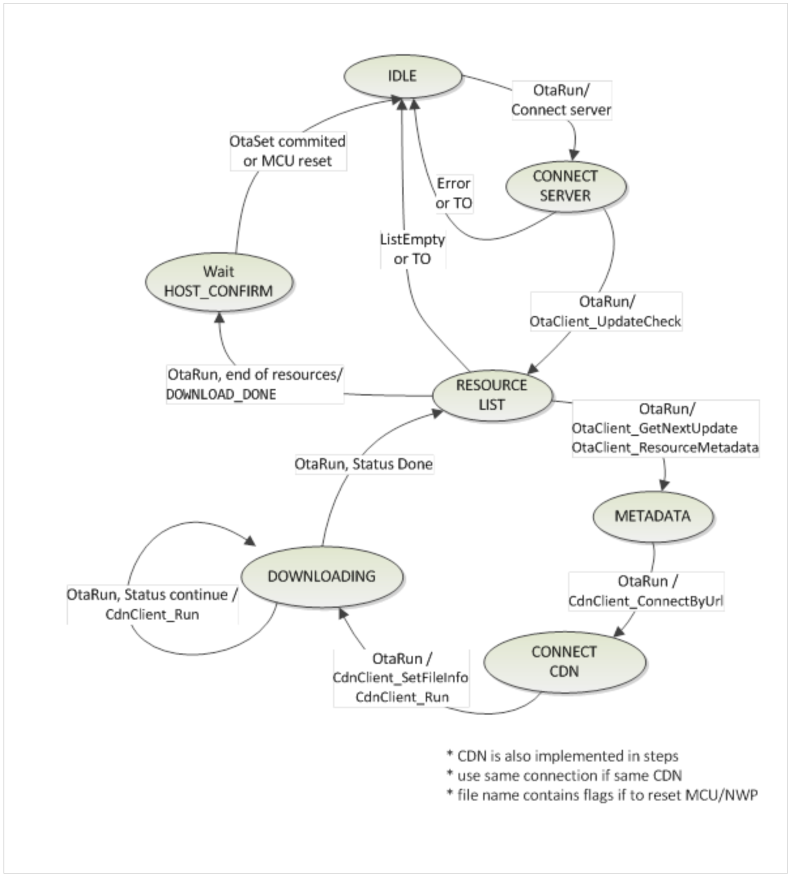


Figure 2 : OTA Application State Machine

Sequence Diagrams

OTA client/server sequence

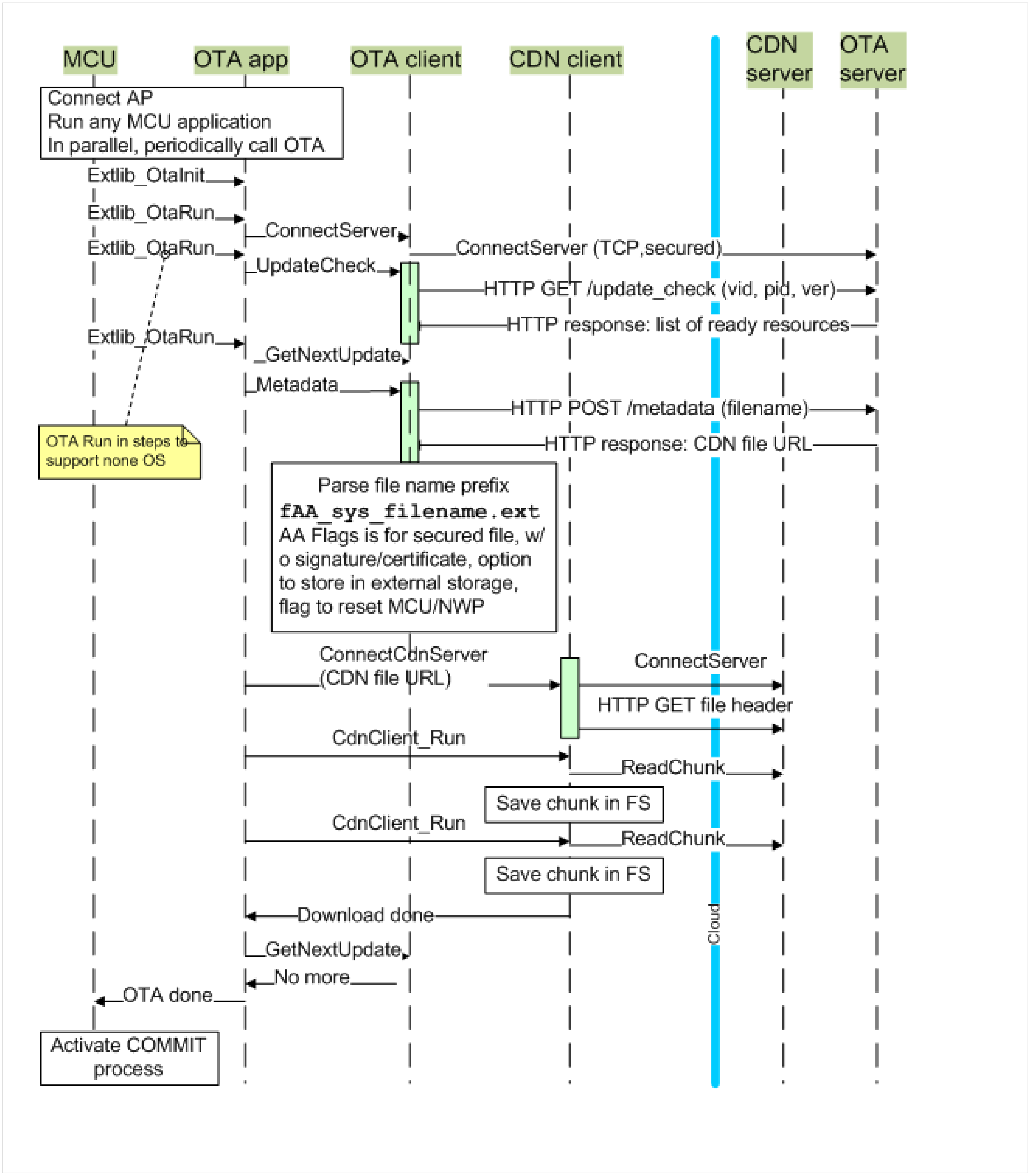


Figure 3 : OTA client/server sequence

Example OTA Update Application

This application focuses on showcasing CC3100'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 ota sample file version and Service Pack version to a macro **OTA_VENDOR_STRING** defined in otaconfig.h file.

Assuming the current ota sample file on device has version number 01 and service pack running on the device holds the version number **2.1.0.12.31.1.1.0.5.1.0.3.20** and **OTA_VENDOR_STRING** is defined as **Vid01_Pid01_Ver,** the application constructs the *VendorString* by appending the 4th byte (shown in red) to the macro i.e. **Vid01_Pid01_Ver0112.** This folder on Dropbox should contain all the files that need to be updated. If left empty OTA library assumes a NO_UPDATE condition.

This application checks for the update every 10s in folder based on the logic mentioned in above paragraph.

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

Usage

Setting up OTA Application for MSP430F5529LP with CC3100 SDK

Basic Setup

1. Copy *simplelink_extLib* directory under the root directory of the SDK, for example: C:\ti\CC3100SDK_1.0.0\cc3100-sdk
2. Copy *ota_sample_app* directory under the examples directory, for example: C:\ti\CC3100SDK_1.0.0\cc3100-sdk\example
3. Copy *flc_lib*, *ota_lib* and *ota_sample_app* project directories under 'platform\msp430f5529lp\example_project_ccs' for example C:\ti\CC3100SDK_1.0.0\cc3100-sdk platform/msp430f5529lp/example_project_ccs
4. Create and setup Dropbox account (see sections below).

Flashing

1. Open Uniflash tool for CC3xxx
2. Mount CC3100 booster pack on CC31XXEMUBOOST board.
3. Format the sFlash.
4. Program the service pack.

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_Pid01_VerXXYY. Refer to "*Configuring the application for new Dropbox account*" section for details.

Configuring the application for new Dropbox account

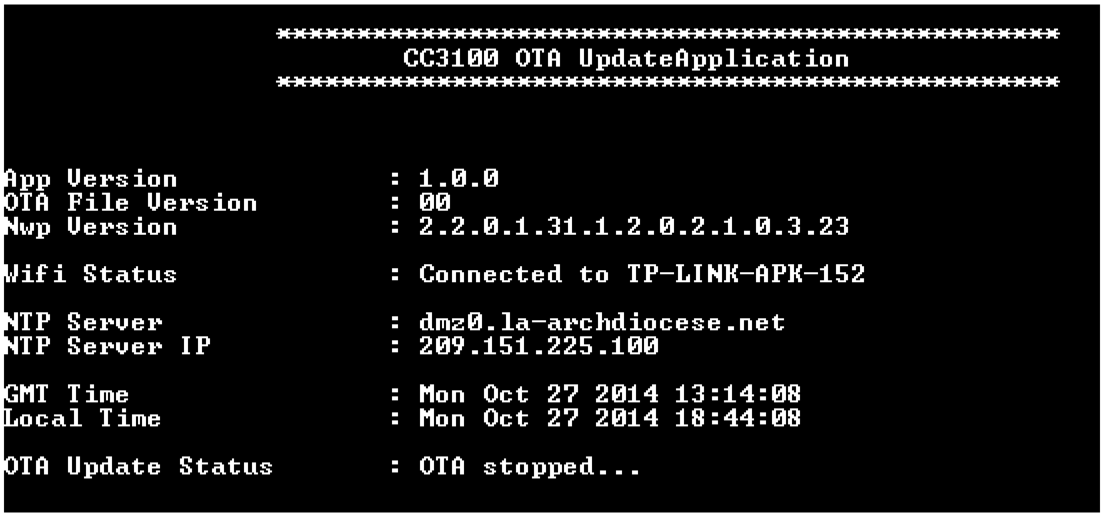
1. Open otaconfig.h
2. Update the following Parameters
    a. OTA_SERVER_REST_HDR_VAL - Set this to Dropbox App token generated in the previous steps

3. Upload the servicepack, ota sample file “**f08_otaSampleFile.txt**” and other user file into “Vid01_Pid01_VerXXYY” folder on Dropbox server where XX is the ota sample file version on serial flash (00 should be used if file is not flashed) and YY is the 4th byte of the NWP version.

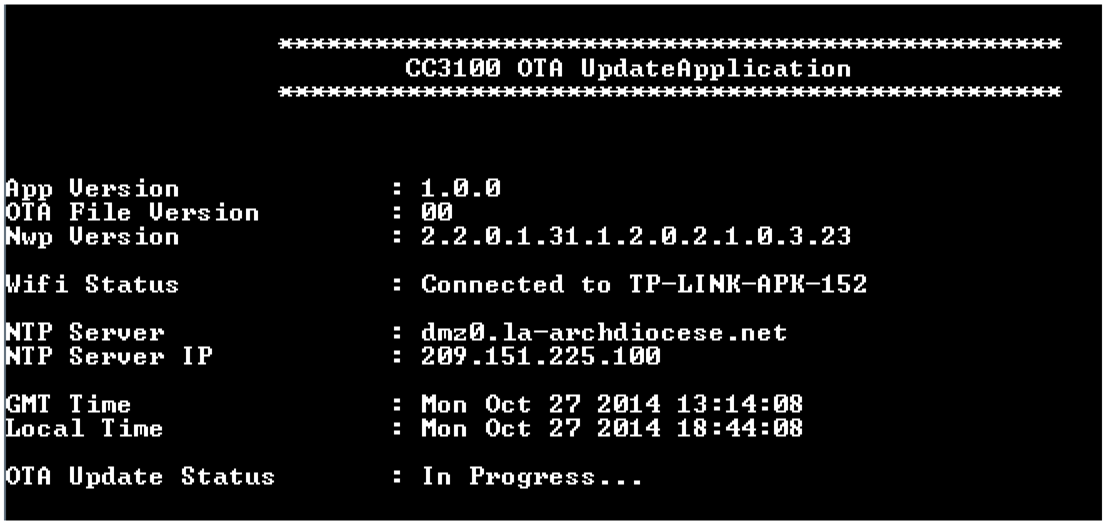
Running

1. Mount the CC3100 Booster-pack on MSP430F5529LP.
2. Connect to COM port via Tera-term or Hyper Terminal with following configuration
    a. **Baud rate**: 9600
    b. **Data**: 8 bit
    c. **Parity**: None
    d. **Stop**: 1 bit
    e. **Flow control**: None

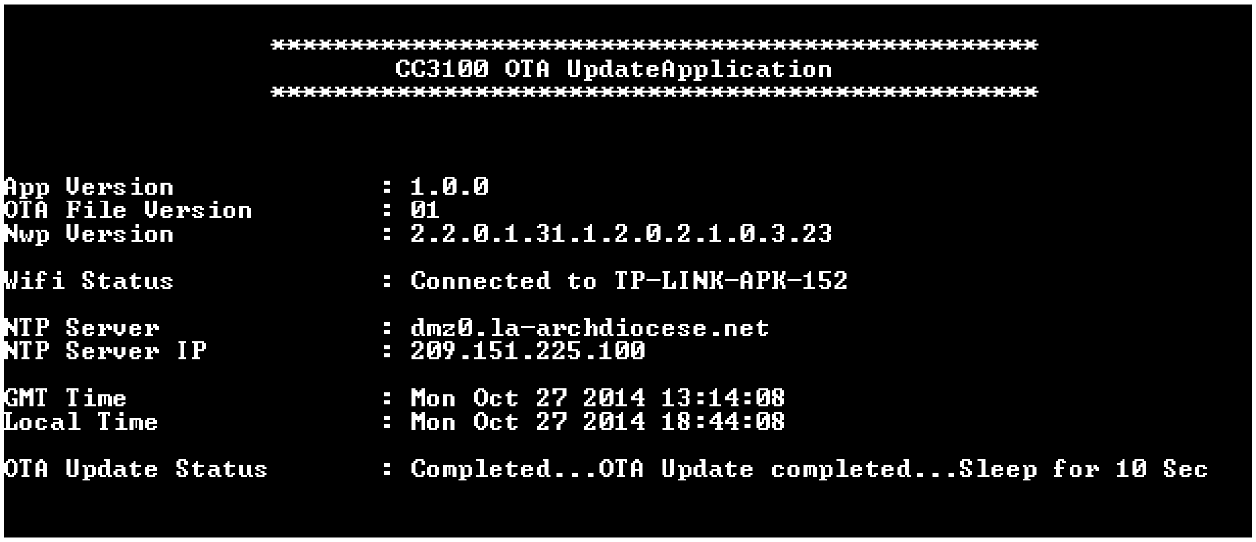
3. Open **sl_common.h** and change **SSID_NAME**, **PASSKEY** and **SEC_TYPE** per your access-point's properties.
4. Build ‘flc_lib’, ‘ota_lib’ library projects.
5. Build and run ‘ota_sample_app’ projects.
6. The application will download the new servicepack and files available at dropbox.
7. See the self-explanatory logs on the terminal-program's console.



```
                    ************************************************
                            CC3100 OTA UpdateApplication
                    ************************************************


App Version                : 1.0.0
OTA File Version           : 00
Nwp Version                : 2.2.0.1.31.1.2.0.2.1.0.3.23

Wifi Status                : Connected to TP-LINK-APK-152

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

GMT Time                   : Mon Oct 27 2014 13:14:08
Local Time                 : Mon Oct 27 2014 18:44:08

OTA Update Status          : OTA stopped...
```



```
                    ************************************************
                            CC3100 OTA UpdateApplication
                    ************************************************


App Version                : 1.0.0
OTA File Version           : 00
Nwp Version                : 2.2.0.1.31.1.2.0.2.1.0.3.23

Wifi Status                : Connected to TP-LINK-APK-152

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

GMT Time                   : Mon Oct 27 2014 13:14:08
Local Time                 : Mon Oct 27 2014 18:44:08

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



```
                  **************************************************
                          CC3100 OTA UpdateApplication
                  **************************************************


App Version               : 1.0.0
OTA File Version          : 01
Nwp Version               : 2.2.0.1.31.1.2.0.2.1.0.3.23

Wifi Status               : Connected to TP-LINK-APK-152

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

GMT Time                  : Mon Oct 27 2014 13:14:08
Local Time                : Mon Oct 27 2014 18:44:08

OTA Update Status         : Completed...OTA Update completed...Sleep for 10 Sec
```

File Naming Convention for OTA on Dropbox

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

/Vid*VV*_Pid*PP*_Ver*XXYY*/f*AA*_sys_filename.ext

The directory /Vid*VV*_Pid*PP*_Ver*XXYY*

- Vid*VV* – Vendor id number
- Pid*PP* – Product id number
- XX – OTA sample file version
- YY – Service Pack version

The filename f*AA*_sys_filename.ext

- f*AA* – 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_Ver0012/f43_sys_servicepack.ucf

Is for vendor id 01, product id 33, version 0012 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
Service Pack	f43_sys_servicepack.ucf
Service Pack signature	f00_sys_servicepack.sig

Limitations/Known Issues

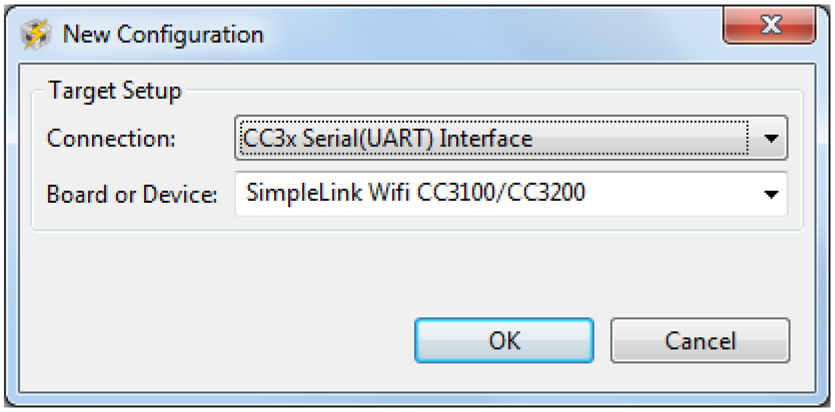
1. OTA cannot update a file to a newer file of same name if the size of the newer file is larger than the max size allocated to that file.
2. Rollback functionality for the service-pack is not supported.

Appendix

Generic Guidelines for Enabling OTA on CC3100/CC3200

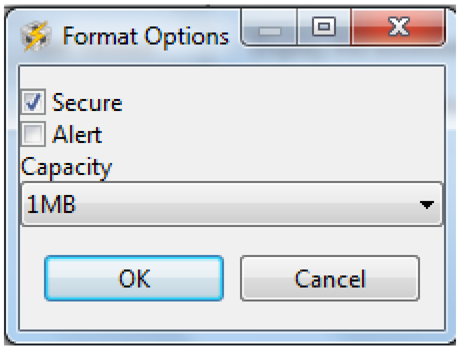
This section list down the recommended order of flashing, using UniFlash tool, to enable OTA on CC3100/CC3200 device(s):

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



Connection: CC3x Serial(UART) Interface
Board or Device: SimpleLink Wifi CC3100/CC3200

3. Format the storage with "*Secure*" checked and "*Alert*" unchecked.



☑ Secure
☐ Alert
Capacity
1MB

4. Update the service pack.

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

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:

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

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.