

# SimpleLink™ Wi-Fi® CC3100, CC3200 UniFlash

This document describes the Uniflash utility functionalities and the general practice of using it with TI's SimpleLink CC31xx and CC32xx devices.

## Contents

1	Introduction .....	3
2	Installation .....	3
3	Prerequisites.....	4
4	Uniflash Block Diagram .....	5
5	Session Support .....	6
6	Secured File System Support.....	11
7	GUI Interface.....	11
8	CC3200 Support.....	18
9	Command Line Support .....	19
10	Configuration File Support .....	27
11	Image Creation and Programming .....	65
12	Troubleshoot/Debugging .....	82
13	Limitations .....	83
14	References .....	83
Appendix A	UART Connection .....	84

## List of Figures

1	Uniflash Block Diagram .....	5
2	Session Conversion .....	6
3	Saving a Session .....	9
4	Loading a Session .....	10
5	Loading a Session From a Bar .....	10
6	Loading a Session .....	11
7	Main Screen .....	12
8	Format .....	13
9	Get Version .....	14
10	Non-Secure Flashing .....	15
11	Servicepack Flashing .....	16
12	Add File.....	17
13	Deleting File.....	18
14	mcu_flashing.....	19
15	Device Role Configuration .....	40
16	Station Flow.....	41
17	Station Configuration .....	42
18	Ap Flow .....	43
19	Ap Configuration.....	44
20	P2p Flow .....	45
21	P2p Configuration .....	46
22	Http Server Flow .....	47

23	Http Server Configuration .....	48
24	Dhcp Server Flow .....	49
25	Dhcp Server Configuration .....	50
26	Mdns Flow.....	51
27	Mdns Main Screen .....	52
28	Mdns Configuration .....	53
29	Profiles Flow .....	54
30	Profiles Main Screen.....	55
31	Open Profile Screen .....	56
32	Wep Profile Screen .....	57
33	Wpa Profile Screen .....	58
34	P2p Profile Screen .....	59
35	Tls Profile Screen .....	60
36	TtlsMschapv2 Profile Screen .....	61
37	PeapPsk Profile Screen .....	62
38	Fast Profile Screen.....	63
39	Smartconfig Flow.....	64
40	Smartconfig Configuration .....	65
41	Add Command.....	69
42	Multiple Remove.....	70
43	Move Command.....	71
44	Export Button .....	72
45	General Setting Button .....	73
46	Programming Button.....	74
47	Programming GUI Flow .....	75
48	Import Session.....	76
49	General Settings.....	77
50	Add Format.....	77
51	Add Logger.....	78
52	Add ServicePack .....	78
53	Export.....	79
54	Debugging.....	82
55	UART Connection .....	84

#### List of Tables

1	Missing Title.....	12
2	Command Line Arguments .....	23
3	Configuration Files .....	80

#### Trademarks

SimpleLink is a trademark of Texas Instruments.  
 Wi-Fi is a registered trademark of Wi-Fi Alliance.  
 All other trademarks are the property of their respective owners.

## 1 Introduction

### 1.1 Overview

The Uniflash utility allows the end-users to communicate with the Simple Link device via a standard Universal Asynchronous Receiver/Transmitter (UART) interface. For UART connection, see [Appendix A](#). The Uniflash utility enables the following functionalities:

- GUI interface
- Flashing of files (including read back verification option)
- Service pack flashing
- Storage format
- Versioning reading for bootloader and chip ID
- CLI interface
- Automatic reset of the board with any operation. Automatic reset is applicable only to TI evaluation boards, CC3100 BoosterPack and CC3200 LaunchPad. Applies to Windows version only
- Building and Flashing of configurations files (a file composed from a collection of parameters)
- Preparing offline image for gang programming and Image Programming via UART (not using direct flash programmers)
- Linux support

## 2 Installation

You can download the latest UniFlash [here](#).

Once you have downloaded the UniFlash installer, you can install the UniFlash application using the provided executable. During installation, you need to choose where you would like to install the application.

---

**NOTE:** Up to version 3.2.0.000123, support for the CC3xxx SimpleLink devices was not part of the Uniflash main stream where all other platforms are supported. However, now that Uniflash has been upgraded to v3.3.x, CC3xxx SimpleLink is also included as one of the supported chipsets. To prevent confusion, v3.2.x still supports the latest CC3xxx SimpleLink (via SW upgrades).

---

### 2.1 Linux

UniFlash is a 32-bit application. Therefore, if you are on a 64-bit Linux OS distribution, you need to install the available 32-bit runtime libraries to be able to execute the UniFlash installer as well as run UniFlash after installation.

Since the dependencies between UniFlash and CCSv6 are similar. For CCS, see the [Linux Installation Instructions](#).

#### 2.1.1 Limitations

Linux version, v3.3 b67, has some limitations compared to the Windows version.

- No automatic reset of the connected target is supported (unlike Windows version)
- Using 'Verify' option (for files or image) results in much slower content reading than the Windows version
- It has been observed occasionally that communication with the target device is halted. It mainly happens during large content reading (with 'Verify' flag checked)

### 2.1.2 During Installation

The Linux installer also installs the drivers/permission files required to use the CC3100/CC3200 devices. But this process needs superuser/root access during installation, so it is recommended to use 'sudo' when invoking the installer.

Alternately, you can run the provided scripts to install the drivers/permission files after installation (but it stills need root access). The script 'cc3x\_ftdi\_usb\_linux\_install.sh' is available in the <installDir>/install\_scripts/ directory.

### 2.1.3 Running UniFlash

When running UniFlash on Linux (either GUI or CLI), you will need to run it using sudo to get access to the USB/serial port.

Alternately, you can manually change the permission of the port after you plug in your device to your machine. Use 'chmod' to change the permission of the /dev/ttyUSBx port for your device. You will need to manually do this every time a new device is plugged into your machine.

Another alternative is to add the current user to the 'dialout' group to gain access to any serial port resource. For instructions on how to do this, contact to your network administrator.

### 2.1.4 C++ Dependencies

On some older distribution of Linux, the standard 32-bit C++ library that is included in the apt packages will not be compatible with UniFlash. Ubuntu 10.04 was an example of this that was noticed in our testing, which comes with libstdc++.so.6.0.13, but UniFlash requires a newer version (6.0.16 or newer). When UniFlash is started, it will fail to load the required libraries when a new session is created.

To work around this issue, a version of the required C++ library is shipped with the UniFlash install. To force UniFlash to load this version of the library, start the UniFlash GUI using the provided uniflashPre.sh in the <installDir>/eclipse/ directory. When this script is executed, the library will load and UniFlash GUI will start.

This issue only affects the GUI; users should be able to use CLI without running into this problem.

### 2.1.5 COM Port

When using UniFlash (in both the GUI and CLI), you need to specify the 'COM' port for communicating with your device. In the Linux version, you only need to provide the numerical part of the 'ttyUSBx' string associated with the port for your device. So if you want to connect to ttyUSB2, enter 2 as the COM port value instead of 'ttyUSB2'.

### 2.1.6 Porting Between Windows and Linux

UniFlash Configuration/Session files should be compatible between Windows and Linux, as long as the file paths (URLs) are updated to match the current folder structure of the new machine. You can use \${sessionDir} for your file paths if your user files are in the session directory to maintain cross platform compatibility. For more information on this topic, see [Section 5.3.3](#).

## 3 Prerequisites

### 3.1 Software

Uniflash package installed on windows OS. Windows 7 and Windows XP are supported.

### 3.2 Hardware

Platforms supported:

- CC3200LP (supports ES1.32 and above)
- CC3100BP (supports ES1.32 and above) - Note that CC31XXEMUBOOST kit is also needed for flashing

- Other evaluation boards are also supported as long as connected to the correct UART port

#### 4 Uniflash Block Diagram

The Uniflash utility should be kept simple and stateless. Since some information should be non-volatile, the Uniflash uses files for storage.

The following set of files is used by Uniflash:

- Template file (CC3xxx\_template.xml): this file lists the files on the target device along with file attributes. It can be considered as a filter between the tool and the target file system. This file holds the required information for all the files, user files, system files and configuration files (HTML, certificates, user files, and so forth). The template file is formatted as an XML file.
- Configuration file (CC3xxx\_cfg\_parameters.xml): this file lists all configurable parameters used as part of the configuration files support. All configurable parameters are also linked to the appropriate target files.
- Tokens file (CC3xxx\_token.xml): whenever authorization is required by the device, tokens should be used. Tokens are the keys for each file. Since the tool should be stateless, it is required to hold the tokens in a non-volatile repository, for example, the local hard disk or a remote server. Without the tokens file, it is not possible to communicate with an authenticated device. The only way to recover is by formatting the device. The Tokens file is formatted as an XML file.
- Image Configuration file (GangImageConfig.xml): this file is used for Image Creation and Programming. It lists all commands that compose the image and extracted and executed upon reset.
- Content of the file system: since the 'configuration file' holds only the offsets of specific parameters inside the configuration files (and not the entire content), the full content files must be provided to the tool.
- Session file (\*.usf): as described under Session Support, a session file includes all relevant files for interaction with the target device along with other specific configurations such as target interface, COM port number, device type, and so forth.

Figure 1 illustrates the structure.

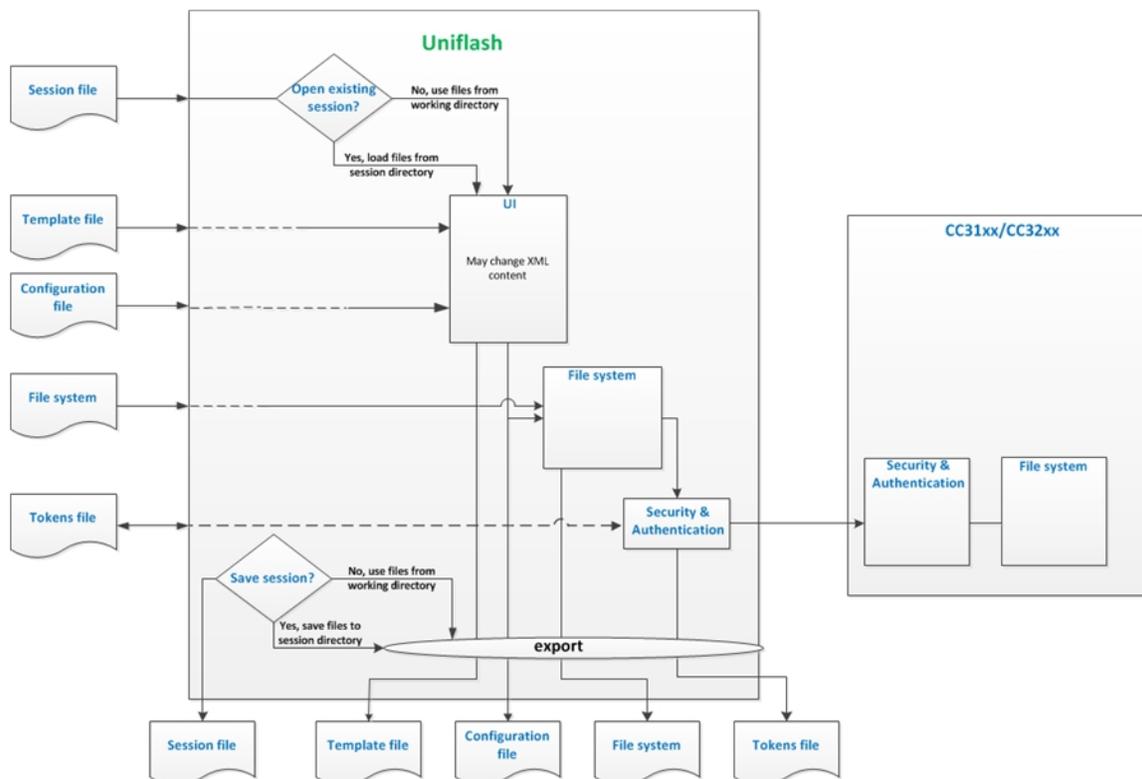


Figure 1. Uniflash Block Diagram

## 5 Session Support

### 5.1 Overview

Sessions can be viewed as working directories where all device specific files are located. It is important to note that each device keeps its own files and each device would most likely have a separate session.

It is also possible to keep different sessions for a single device. In this mode, each session represents a different configuration of the device. For example, one session for device upgrade (service pack update) and another for user files flashing.

### 5.2 Session Conversion

Session concept was first introduced in version 3.2.0.00019. However, since the database has been completely modified, it is not possible to convert session files created in version 3.2.0.00019 to work on version 3.2.0.00120.

Please note that converting from version 3.2.0.0065 to 3.2.0.00120 is possible and done automatically. When trying to open a session created in version 3.2.0.0065, a conversion message would appear asking to confirm. Clicking OK, results in converting to version 3.2.0.00120. Also note that once conversion is completed, the same session cannot be opened anymore by version 3.2.0.0065.

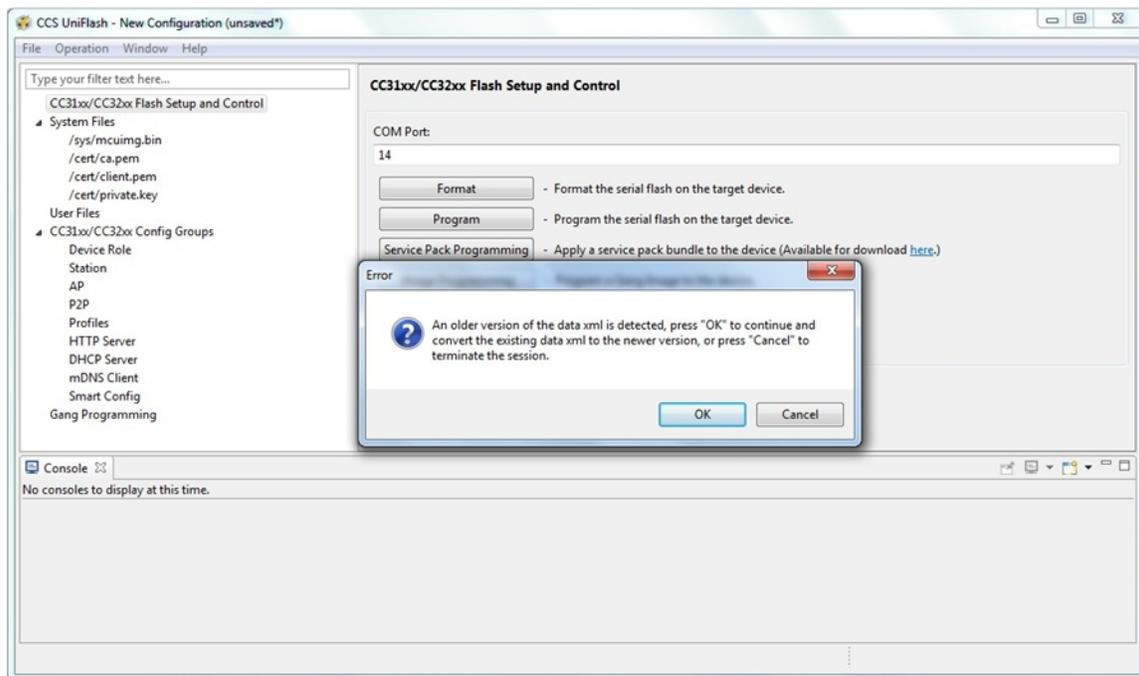


Figure 2. Session Conversion

## 5.3 Creating a Session

### 5.3.1 Session Content

Session files are saved as \*.usf files (UniFlash Session File) and includes information about the interface, device, and the session that you want to use. Configurations can be saved and opened in UniFlash to preserve and restore settings, and to maintain a different set of files for different devices.

Session consists of all relevant files for interaction with the target device along with other specific configurations such as target interface, COM port number, device type, and so forth. When a session is created, the following files can be found under the root session directory

- ./templates: includes all XML files with their default value as follows:
  - CC3xxx\_cfg\_parameters.xml
  - CC3xxx\_template.xml
  - CC3xxx\_token.xml
  - GangImageConfig.xml
- ./filesystem: includes all binary template files with their default value. The binary template files are relevant for configuration files only. By default, no binary template files are created. Only upon modifying parameters, the respective file is created. Possible template files are:
  - macadd.bin
  - ipcfg.ini
  - mode.cfg
  - devname.cfg
  - pmcfg.ini
  - smartconfigkeys.cfg
  - stacfg.ini
  - ap.cfg
  - p2p.cfg
  - dhcprsv.cfg
  - htprsv.cfg
  - mdns.cfg
  - pref.net

Session files are saved as \*.usf files (UniFlash Session File) and includes information about the interface, device, and the session that you want to use. Configurations can be saved and opened in UniFlash to preserve and restore settings, and to maintain a different set of files for different devices.

Note that upon saving a session, for example, as configname.usf, a directory named configname\_session is created in the same folder. This should make it portable in that as long as the file and folder are sent together, the user should be able to open it in UniFlash.

An example of configuration file content is as follows:

```
#Tue Apr 28 09:14:08 IDT 2015
capacity=1MB
com=21
interface=CC31x Flash Connections
device=CC31x Flash Devices
secure=true
alert=true
CFG_GRP_ID|DHCP_Server=false
CFG_GRP_ID|Station=true
CFG_GRP_ID|mDNS_Client=false
CFG_GRP_ID|Device_Role=true
CFG_GRP_ID|Profiles=false
CFG_GRP_ID|AP=false
CFG_GRP_ID|Profile7=false
CFG_GRP_ID|Profile6=false
```

```
CFG_GRP_ID|Profile5=false
CFG_GRP_ID|Profile4=false
CFG_GRP_ID|Profile3=false
CFG_GRP_ID|mDNSClient5=false
CFG_GRP_ID|Profile2=false
CFG_GRP_ID|mDNSClient4=false
CFG_GRP_ID|Profile1=false
CFG_GRP_ID|mDNSClient3=false
CFG_GRP_ID|mDNSClient2=false
CFG_GRP_ID|mDNSClient1=false
CFG_GRP_ID|P2P=false
CFG_GRP_ID|HTTP_Server=false
CFG_GRP_ID|Smart_Config=false
```

### 5.3.2 Creation Procedure

There are two ways to create a session:

- One way is from the top status bar by 'File → New Target Configuration'.
- The other way is from the Quick start guide by clicking 'New target configuration'.

When users choose to create a new configuration, a temporary session is created with the default template and token xml files. Upon creation, “unsaved\*” header information is stated on the top frame of Uniflash to indicate that the session is temporary. Users are free to modify this session to their liking and execute operations on their device. Users can also create a new configuration again to purge all of the current changes and start with the default template again.

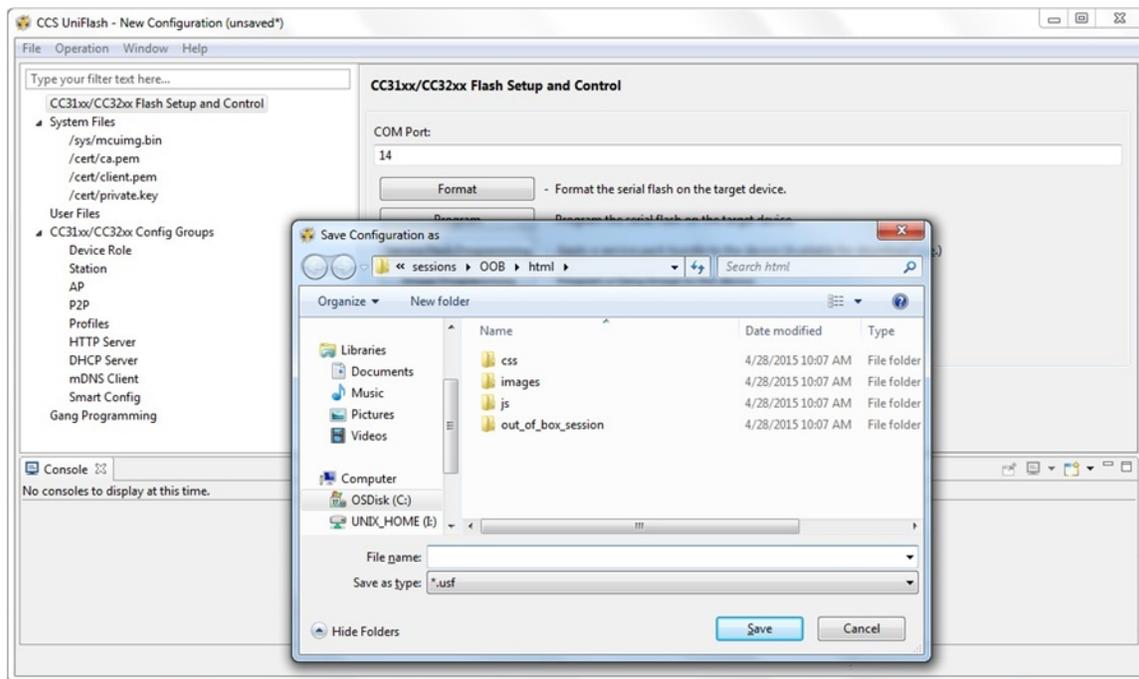
### 5.3.3 Relative Path Support

Relative paths support is essential for session ease of use. This feature makes the session feature agnostic to where the files are actually located. For example, the SDK is released with build in session configurations files (\*.ucf). These files are created in advance so users do not need to create it from scratch. However, if users decide not to use the default installation directory for the SDK, all absolute paths on the template XML files would fail. This is where relative path support comes into play. The template XML supports the special tag “\${sessionDir}” in the URL variables. This special tag is set according to the session itself (and thus is different per session) and links to the location of the directory named configname\_session (that is created automatically).

## 5.4 Saving a Session

Users can save their configuration by choosing the location and name of the configuration file (.usf will be added automatically if not provided). Once a configuration is created and saved, any further changes in the GUI will be automatically saved to this configuration (users do not need to choose to save again, unless then want to save to a different configuration). A directory with the template and token xml files is created at the location where the user chose to save the configuration file.

Note that if users do not save their configuration before closing UniFlash, the changes will be lost (although the actual files are still there in the temporary location).



**Figure 3. Saving a Session**

## 5.5 Loading a Session

There are two ways to load a session:

- One way is from the top status bar by 'File → Open Configuration'.
- The other way is from the Quick start guide by clicking on one of the recent sessions.

Choosing an existing .usf file, UniFlash will be populated with the same template, token and other GUI changes from the saved session. Once loaded, any changes that users perform in the GUI will automatically be saved to the same configuration.

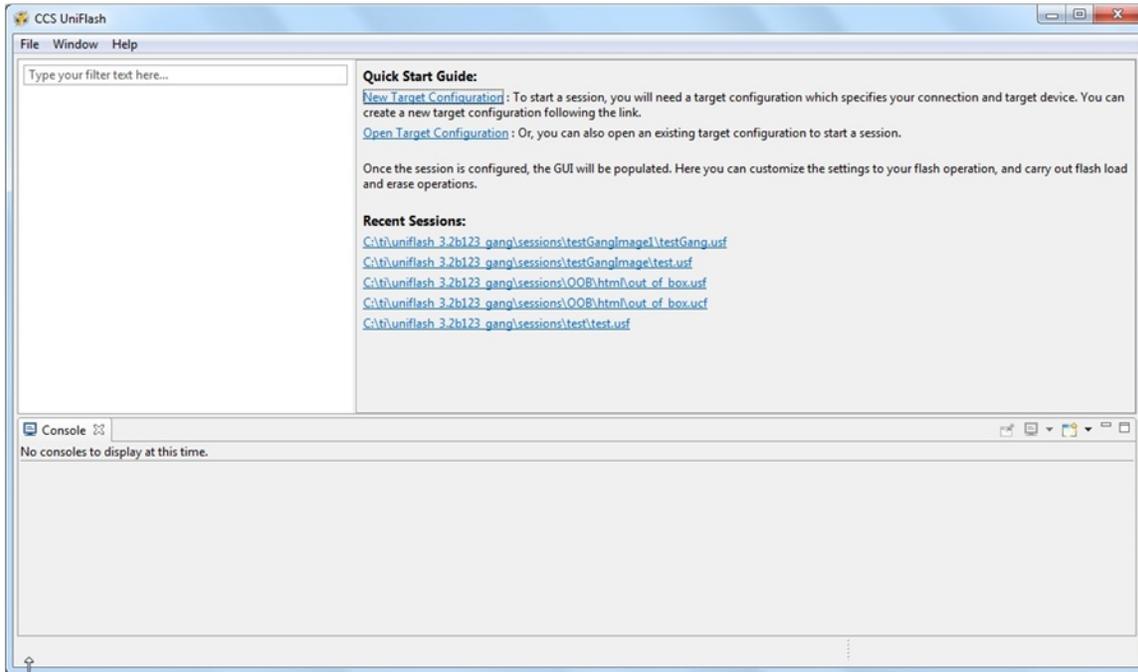


Figure 4. Loading a Session

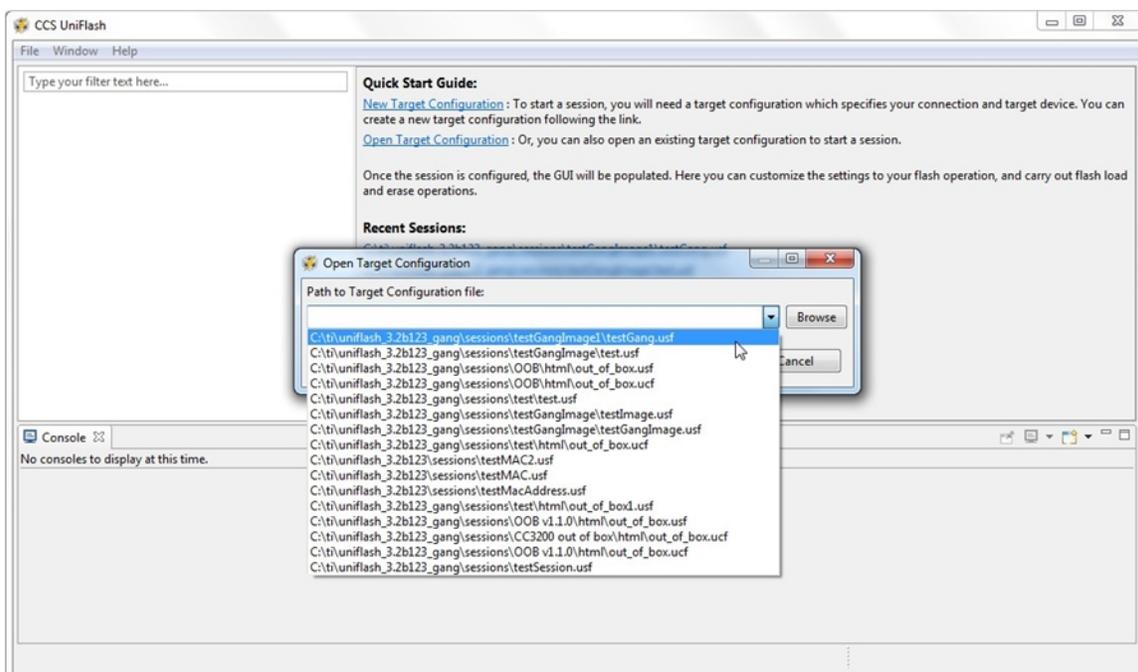


Figure 5. Loading a Session From a Bar

## 6 Secured File System Support

There are two types of CC3100 and CC3200 devices: production devices (PG1.33) and pre-production devices (ES1.32 or lower). Both devices are not considered secured devices.

Uniflash version v3.2.0.00123 or earlier had the option to format the device as secured (in case it is not) and also create secured files. However, secured file system support is deprecated from other Uniflash v3 versions since this feature is not supported on CC3100/CC3200.

Support for the secured file system was added for the next generation of SimpleLink Wi-Fi devices. For more information, go to [ti.com/simplelinkwifi](http://ti.com/simplelinkwifi).

## 7 GUI Interface

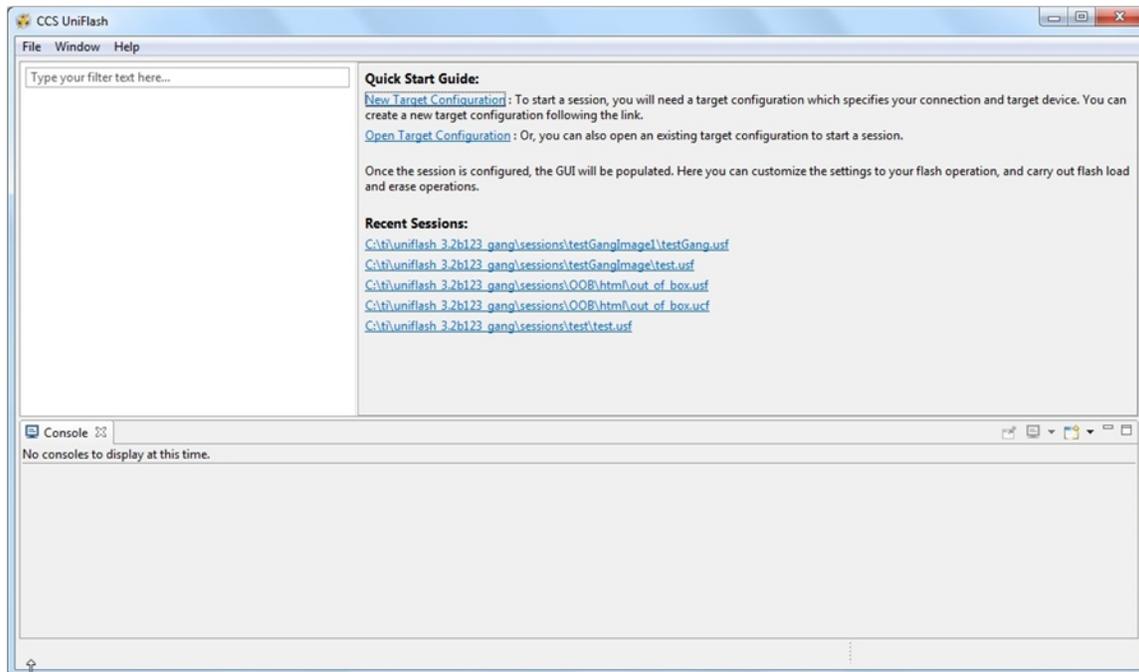
### 7.1 Main Screen

Upon running the Uniflash, the user is required to choose the target setup.

There are two ways the target setup can be configured:

- One way is from the top bar by file → New Configuration
- The other way is from the Via the Quick Start Guide on the main screen by clicking New target configuration.

Currently, the Connection is set to CC3x Serial (UART) Interface and the Board or Device is set to SimpleLink WiFi CC3100/CC3200.



**Figure 6. Loading a Session**

Loading the target configuration provides the Uniflash main screen.

The main screen is divided into three main sub screens:

- Uniflash main view: this is the upper left section on Main screen. All the presented data is parsed from the template XML and presented in a list form. The list is the list of target filenames.
- CC31x Flash Setup and Control: this is the upper right section on Main screen. It details the configurable options for each of the files chosen on the Uniflash main view. It also provides interface connection to the target device and buttons options (highlighted in next paragraphs).
- Console: this is a status window. Detailed information is printed during interaction with the device.

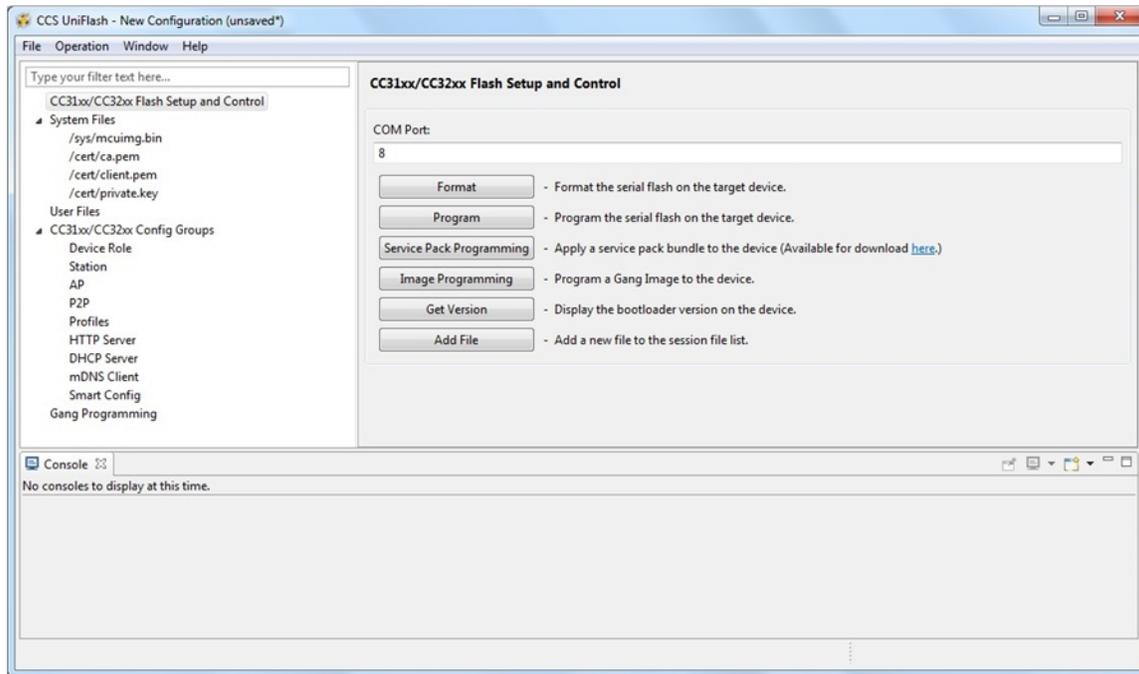


Figure 7. Main Screen

## 7.2 Target Connection

Connection to the target device is via UART interface. The user should configure the COM port number and the baud rate. Currently, no automatic detection of COM port is supported and the baud rate supported is 921,600 only.

The COM port number needs to be fetched from the 'Windows Device Manager'. Ideally, all FTDI chipsets on the evaluation boards are pre-flashed. However, FTDI chipsets that are not flashed for some reason, can still be used as Uniflash requires a simple COM port.

Table 1 describes all possible permutations:

Table 1. Missing Title

Evaluation Board	FTDI	#COM Ports	#COM Port for Flashing
CC3200 LaunchPad	Pre-flashed FTDI	1	The only COM port exposed
CC3200 LaunchPad	Non flashed FTDI	2	The upper COM port
CC3100 BoosterPack	Pre-flashed FTDI	2	The lower COM port
CC3100 BoosterPack	Non flashed FTDI	4	The 3rd COM port

In addition, the CC3200 LaunchPad can be made to work in two modes: functional mode and UART load mode.

UART load mode is to be used when flashing the device. This mode is selected by connecting "SOP2" jumper on J15 (denotes on PCB as '100:FLASH').

### 7.3 Format

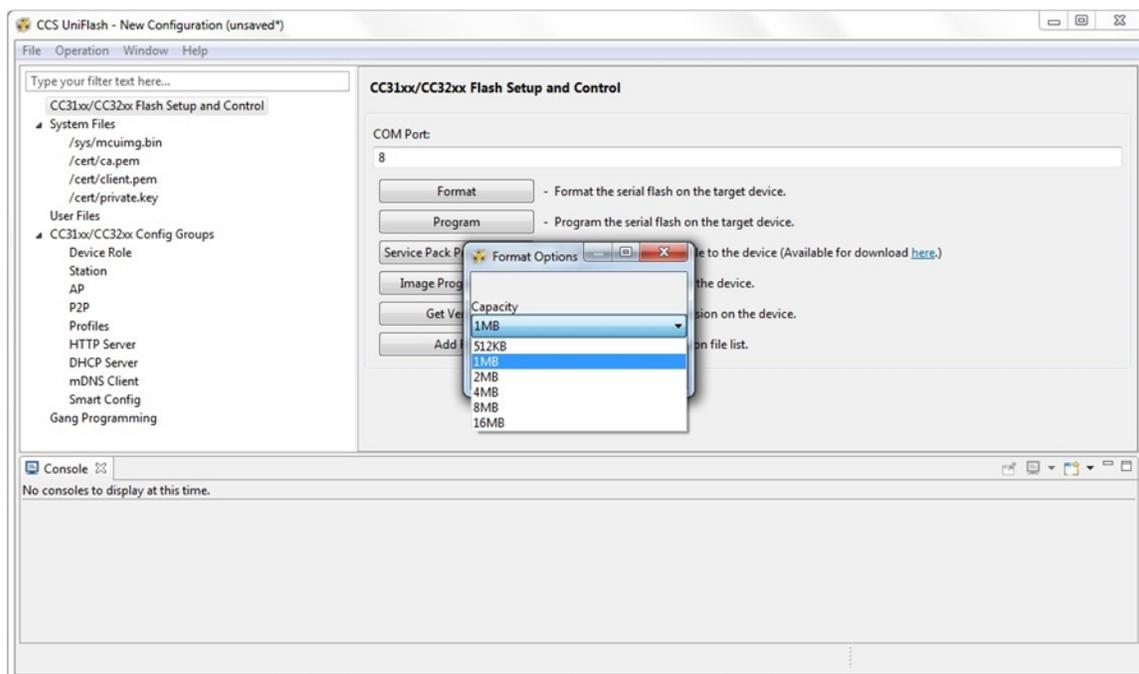
If you are using pre-production CC3200 Launch Pad with XCC3101GZ or XCC3200HZ and you are using Uniflash version v3.2.0.00123 or earlier, the device must not get formatted as secured. Uniflash version which is newer than v3.2.0.00123 does not have the option to set the format as secured.

The user is required to choose the capacity storage. It is the responsibility of the user to choose a capacity less than or equal the total size of the serial flash. The options are: [512KB, 1MB, 2MB, 4MB, 8MB and 16MB].

---

**NOTE:** Choosing a lower size than the actual size results only in not using the entire space (but no unexpected behaviors). However, choosing a higher size than the actual size may result in unexpected behaviors.

---



**Figure 8. Format**

## 7.4 Get Version

The version that can be fetched from the device is the bootloader version (as Uniflash interacts with the bootloader section). Additionally, the chipset version is fetched. The bootloader version is composed of four numbers, separated by periods.

The syntax is:

<SoftwareMajor.SoftwareMinor.SoftwareVersion.SoftwareSubVersion>

Note that the bootloader version fetched is the one that is located in ROM.

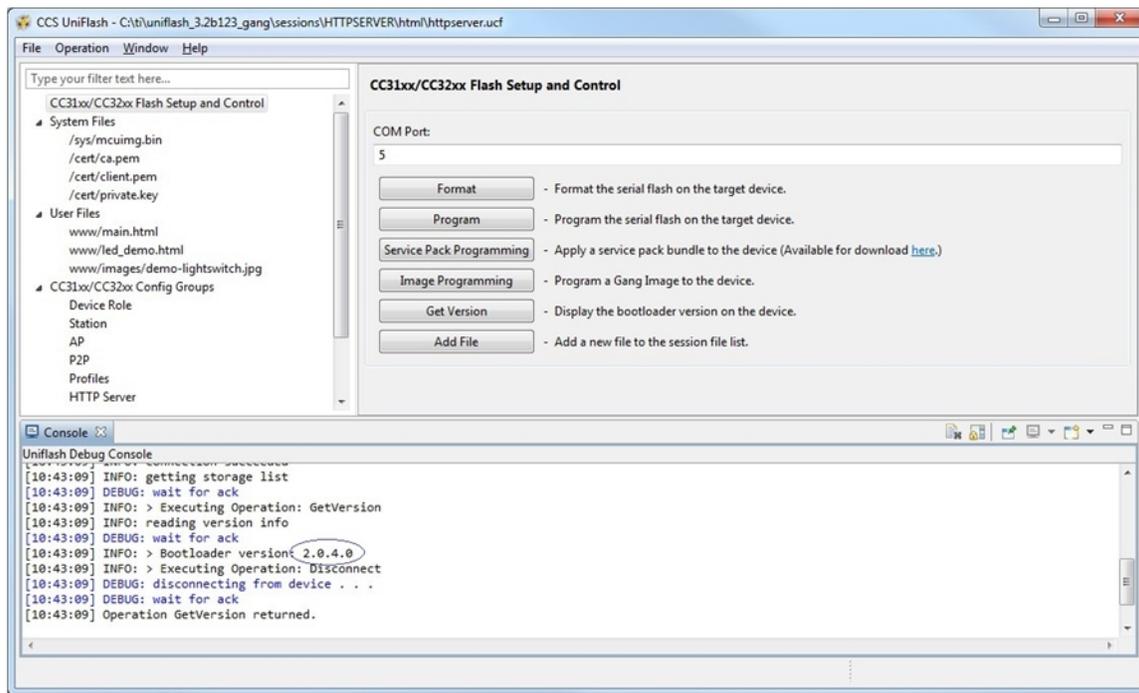


Figure 9. Get Version

As can be seen in Figure 9, invoking 'Get Version' results in version 2.0.4.0 printed on the 'Console' sub screen.

## 7.5 File Programming

### 7.5.1 General

Programming of the device is applied upon pressing the 'Program' button. Another way is from the top bar by clicking Operation->Program. Upon pressing the 'Program' button, Uniflash utility scans all target filenames and configuration groups listed and applies the followings (ordered):

- Each file with the 'erase' argument set to True is erased from the device
- Each file with the 'update' argument set to True is flashed to the device
- Each file with the 'verify' argument set to True is verified by reading it back from the device

Note that the files are scanned top-down as appears on the Uniflash main view screen.

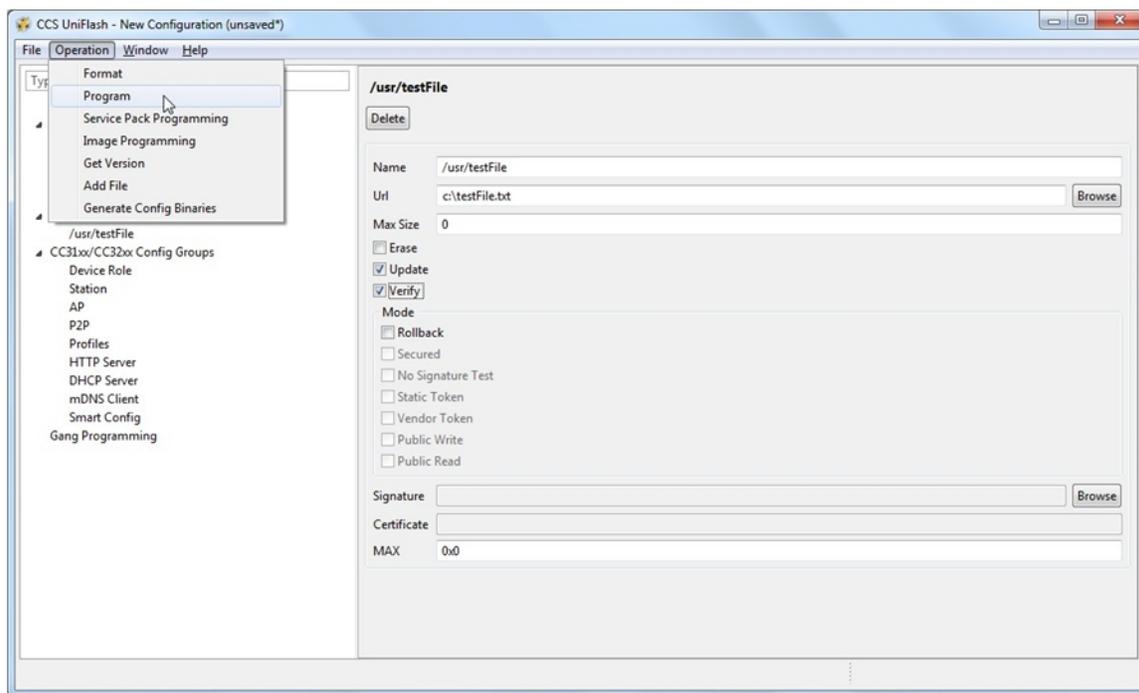
Uniflash does not do conversions for binary-to-text encoded files (hex, Base64).

### 7.5.2 Non-Secured File Programming

Non secured file programming may be applied on a secured device as well. The only option under 'mode' for a non secured file is rollback option.

The relevant configuration is listed below:

- Name: target file name. System filenames cannot be modified, whereas, user filenames may be modified by the user.
- Mode: nothing is checked, indicates non-secured file. Only rollback may be checked.
- MaxSize: 0 to indicate the original file size. User may configure a file size larger than the original if it is desired to save space for future appending.
- Url: full path where the file is located
- Update checkbox: checked for flashing
- Verify checkbox: checked for flashing verification



**Figure 10. Non-Secure Flashing**

Upon pressing the 'Program' button, the file is flashed. The progress bar is updated frequently and messages are printed on the 'Console' sub screen.

### 7.5.3 Secured File Programming

For more information, see [Section 6](#). Uniflash version v3.2.0.00123 supported security configuration options per file but this feature is deprecated from Uniflash until devices support this feature.

All security configuration options still appear on Uniflash GUI, but are greyed out.

## 7.6 Service Pack Programing

### 7.6.1 General

Service pack is a common name for the image required for updating the device (either upgrade or downgrade the device). The service pack is a single file containing images for all flavors of the device, regardless if it is CC3100/CC3200 or ROM/Flash device. The detection is done automatically by Uniflash.

The service pack binary file is not part of Uniflash and should be fetched from TI repository separately.

### 7.6.2 Procedure

Use these steps for a successful update procedure:

1. Format the device prior to flashing. This step is essential and should be performed at least once. With the next service pack updates, the format operation would not be required. For more details, follow the Format paragraph in [Section 7.3](#).
2. Invoke the Service Pack Update from a dedicated button or from the Operation → Service Pack Programming option in the top menu.
3. Choose the binary servicepack file and click OK.
4. Flashing should begin and the version flashed is printed just before flashing is started,

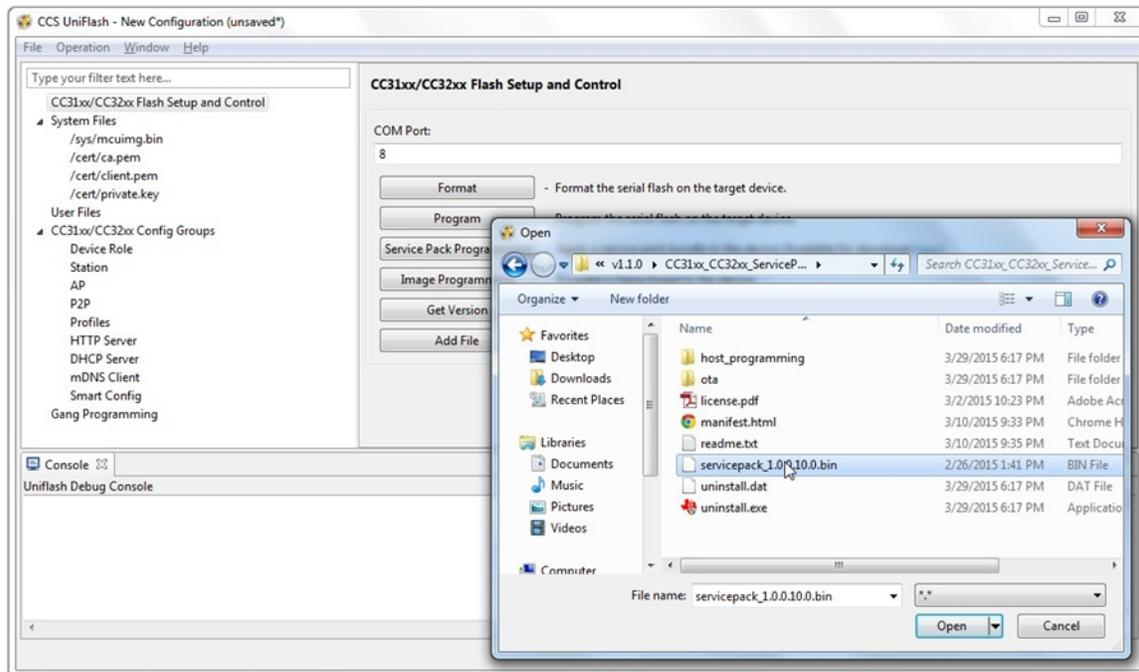


Figure 11. Servicepack Flashing

## 7.7 Adding a New File to the Device

**NOTE:** For web pages, see the *HTTP Server* chapter in the [CC3100/CC3200 SimpleLink™ Wi-Fi® Internet-on-a-Chip User's Guide](#). Either `www/` or `www/safe/` prefixes should be added to the filename when downloading files to the file system for the internal HTTP server to access those pages.

Uniflash enables the user to add any file via the GUI using the 'addFile' button. Another way is from the top bar by clicking Operation → Add File. This option applies to user files only as system files are reserved names on the device. Adding a file creates a blank template file that needs to be filled by the user.

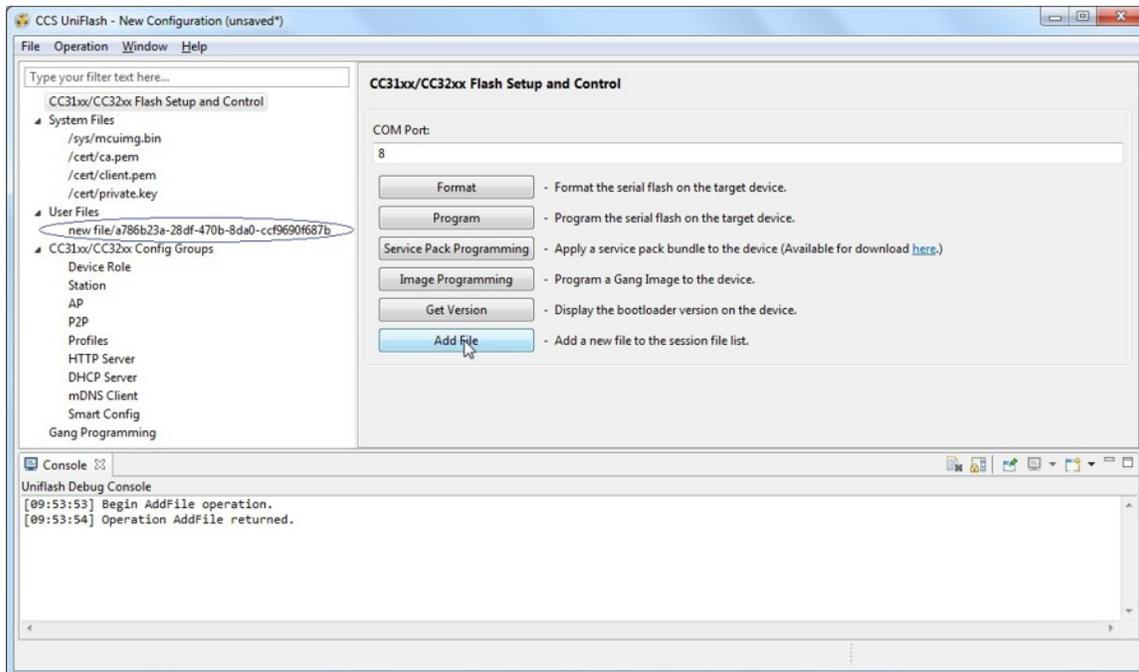


Figure 12. Add File

## 7.8 Erasing Existing File From the Device

Uniflash enables the user to remove any file via the GUI using the 'erase' parameter under each file. Checking the erase option and also the update option would first erase the file but then program it again.

Actual erasure starts upon invoking the program button. Note that erasing a file from the device does not delete it from the GUI.

Also note that checking the 'erase' option and also the 'update' option would first erase the file but then update it again.

## 7.9 Removing Existing File From the Template

Uniflash enables the user to delete any user file from the GUI and the serial flash via the GUI using the delete button under each file (located on the top next to the target filename).

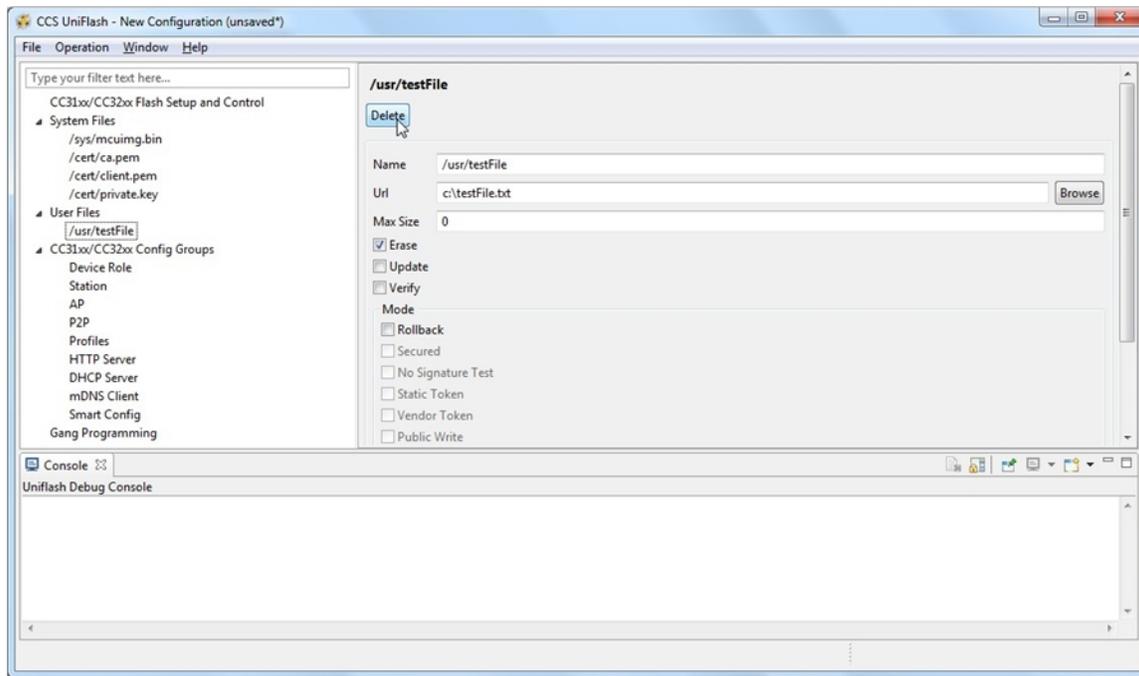


Figure 13. Deleting File

## 8 CC3200 Support

### 8.1 Overview

Like CC3100 chipset, CC3200 is a non-secured device. Additionally, CC3200 can also come as Flash part.

For better understanding:

- Current existing part:
  - CC3200R – non-secure MCU wherein customer image in external flash is not protected.
  - CC3200Z – non-secure MCU wherein customer image in external flash is not protected.
- Future parts:
  - CC3200 secured device – secure MCU wherein customer image in external flash is protected.
  - CC3200 flash part – customer image resides in the internal flash.

### 8.2 Configuration and Programming

Current available non-secured CC3200: User can only create a mcuimg file without any security and it will work.

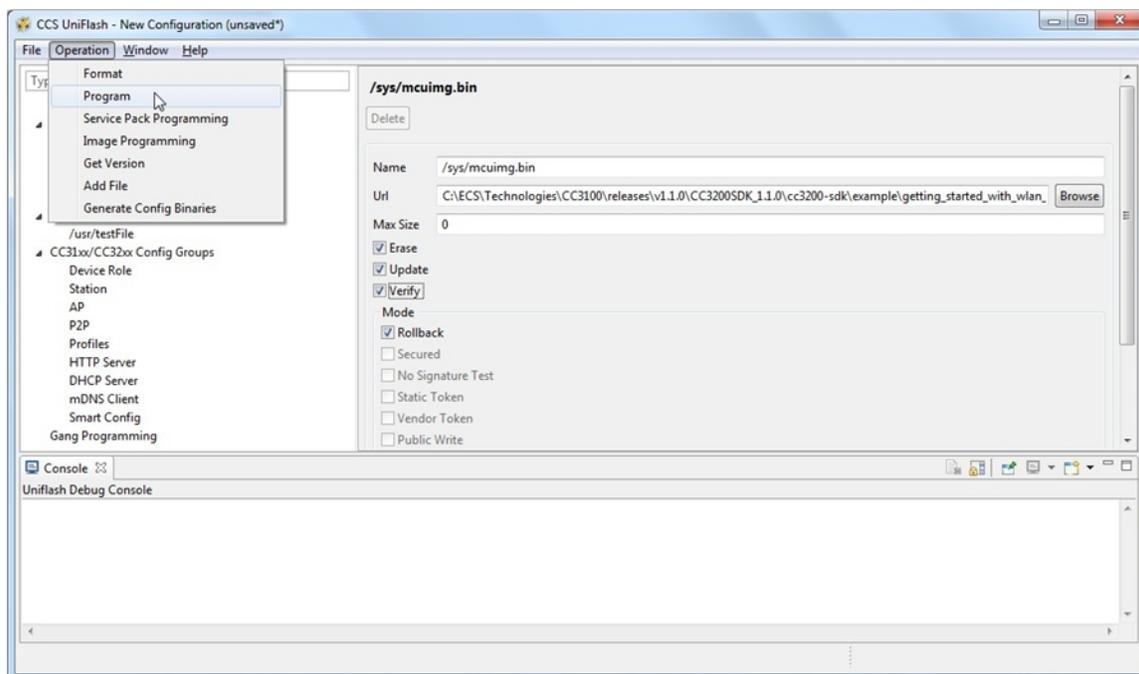
Future secured CC3200:

- User can create a mcuimg file as non-secured and it will work
- User can create a mcuimg file as secured and it will work

Another important parameter that requires a special care is the maximum size of the MCU image. On CC3200R devices, RAM size is 256KB but only 240KB is available (16KB is for system usage). On CC3200Z devices, only 176KB is available.

An example configuration of MCU image is listed below:

- Name: /sys/mcuimg.bin (not configurable by the user).
- MaxSize: 0 to indicate the original file size. Note that for production devices, the maximum possible size is 245760 bytes (240KB). The reason is that the RAM size is 256KB and 16KB are required by the system.
- Mode: nothing is checked, indicating a non-secured image. Rollback is mandatory in order to support OTA (non-configurable by the user).
- Signature: greyed out
- Certificate: greyed out
- Url: full path where the file is located
- Erase checkbox: checked for erasing
- Update checkbox: checked for flashing
- Verify checkbox: checked for flashing verification



**Figure 14. mcu\_flashing**

## 9 Command Line Support

### 9.1 Overview

Command line support is mainly required for automated setup and production line purposes. Command line supports any operation that is supported via the GUI interface except for file addition. It is implemented as a batch executable followed by a collection of arguments as inputs. The set of arguments are described in the next paragraph.

## 9.2 Programming

### 9.2.1 Command Line Syntax

The Uniflash command line is located on Uniflash main directory. For Windows, the user needs to open a DOS command line shell and navigate to Uniflash main directory.

The basic syntax for commands is as follows:

```
>> uniflashCLI [CONFIG] [SETTINGS] [OPERATIONS]
```

For Linux, the user needs to open a terminal shell and navigate to Uniflash main directory. The basic syntax for commands is as follows:

```
>> ./uniflashCLI.sh [CONFIG] [SETTINGS] [OPERATIONS]
```

Where:

- [CONFIG] indicates the configuration file to be used. The configuration file is the session \*.usf file. It is required mainly for the Interface and Device description but can also include other settings as described under the [SETTINGS] argument.
- [SETTINGS] indicates the various options for arguments settings. All settings can be either declared on the \*.usf file or explicitly declared on the command line. Several arguments may be set in a single command.
- [OPERATIONS] indicates the various operation options. These operations are equivalent to the GUI buttons. Operations may be concatenated (several operations can be invoked serially in a single command).

### 9.2.2 Help Manual

To get help on command syntax, run the following command:

On Windows:

```
>> uniflashCLI.bat -help
***** Texas Instruments Universal Flash Programmer *****
```

```
Usage: uniflash [CONFIG] [SETTINGS] [OPERATION]
```

```
[CONFIG]:
-config <configPath>
```

```
[SETTINGS]
-listOptions
-setOptions <OptID1>=<Value1> ... <OptIDN>=<ValueN>
```

```
[OPERATION]
-listOperations
-operation <opCode1> ... <opCodeN>
<End of command>
```

On Linux:

```
>> ./uniflashCLI.sh -help
***** Texas Instruments Universal Flash Programmer *****
```

```
Usage: uniflash [CONFIG] [SETTINGS] [OPERATION]
```

```
[CONFIG]:
-config <configPath>
```

```
[SETTINGS]
-listOptions
-setOptions <OptID1>=<Value1> ... <OptIDN>=<ValueN>
```

```
[OPERATION]
-listOperations
-operation <opCode1> ... <opCodeN>
```

<End of command>

To get help on the list of setting options and operations, run the following command:

On Windows:

```
uniflashCLI.bat -config <any existing *.ucf> -listOptions -listOperations
***** Texas Instruments Universal Flash Programmer *****
> Available Options:
* debug      - determine if debug output is enabled or not [true,false(default)]
* com        - the COM port to use when connecting to the device via serial interface
* capacity   - the capacity format flag [512KB, 1MB(default),2MB,4MB,8MB,16MB]
* spPath     - path to the service pack file (for the servicePackUpdate operation)
* imagePath  - path to the gang image file (for the imageProgramming operation)

> Available Operations:
* getVersion - Get the available version information from the device (Bootloader)
* format     -
Format the serial Flash (use secure, alert and capacity option to control the operation)
* program    - Program the serial Flash based on the given configuration
* servicePackUpdate - Apply a service pack bundle to the device
* imageProgramming - Programs the provided image to the device
<End of command>
```

On Linux:

```
./uniflashCLI.sh -config <any existing *.ucf> -listOptions -listOperations
***** Texas Instruments Universal Flash Programmer *****
> Available Options:
* debug      - determine if debug output is enabled or not [true,false(default)]
* com        - the COM port to use when connecting to the device via serial interface
* capacity   - the capacity format flag [512KB, 1MB(default),2MB,4MB,8MB,16MB]
* spPath     - path to the service pack file (for the servicePackUpdate operation)
* imagePath  - path to the gang image file (for the imageProgramming operation)
* imageVerify - determines if we need to verify after image programming [true,false(default)]

> Available Operations:
* getVersion - Get the available version information from the device (Bootloader)
* format     -
Format the serial Flash (use secure, alert and capacity option to control the operation)
* program    - Program the serial Flash based on the given configuration
* servicePackUpdate - Apply a service pack bundle to the device
* imageProgramming - Programs the provided image to the device
<End of command>
```

### 9.3 Available Options

<b>Argument</b>	com
<b>Description</b>	Indicates the COM port to be used when connecting to the device
<b>Options</b>	Decimal number indicating the COM port
<b>Default</b>	N/A
<b>Required?</b>	Yes, can also be set on the *.usf

<b>Argument</b>	imagePath
<b>Description</b>	Indicates the path to the Image file
<b>Options</b>	String of the full path (insensitive to '/' or '\')
<b>Default</b>	N/A
<b>Required?</b>	Yes, can also be set on the *.usf

<b>Argument</b>	debug
<b>Description</b>	Verbose mode for printing additional information
<b>Options</b>	[true, false]
<b>Default</b>	False
<b>Required?</b>	Yes, can also be set on the *.usf

<b>Argument</b>	Capacity
<b>Description</b>	Serial flash capacity for format operation
<b>Options</b>	[512KB, 1MB, 2MB, 4MB, 8MB, 16MB]
<b>Default</b>	1MB
<b>Required?</b>	Yes, can also be set on the *.usf

<b>Argument</b>	spPath
<b>Description</b>	Indicates the path to the service pack file
<b>Options</b>	String of the full path (insensitive to '/' or '\')
<b>Default</b>	N/A
<b>Required?</b>	Yes, can also be set on the *.usf

<b>Argument</b>	imageVerify
<b>Description</b>	Image verification after programming. Linux only
<b>Options</b>	[true, false]
<b>Default</b>	False
<b>Required?</b>	Yes, can also be set on the *.usf

### 9.3.1 Available Operations

<b>Argument</b>	getVersion
<b>Description</b>	Getting the bootloader version of the device

<b>Argument</b>	format
<b>Description</b>	Serial flash format

<b>Argument</b>	program
<b>Description</b>	Programming of any element except from service pack

<b>Argument</b>	servicePackUpdate
<b>Description</b>	Service pack programming

<b>Argument</b>	imageProgramming
<b>Description</b>	Image programming

### 9.3.2 Legend

Table 2 summarizes all of the command line arguments.

**Table 2. Command Line Arguments**

Argument	Required?	Default	Description
config	yes	N/A	
listOptions	no	N/A	
listOperations	no	N/A	
com	no	Decimal number indicating the COM port	Indicates the COM port to be used when connecting to the device
imagePath	no	String indicating the path	Indicates the path to the image file
imageVerify	no	false	image verification after programming. Linux only
debug	no	false	Verbose mode for printing additional information
capacity	no	1MB	Serial flash capacity for format operation
spPath	no	String indicating the path	Indicates the path to the service pack file
getVersion	no	N/A	Getting the bootloader version of the device
format	no	N/A	Serial flash format
program	no	N/A	Programming of any element except from service pack
servicePackUpdate	no	N/A	Service pack programming
imageProgramming	no	N/A	Image Programming

### 9.3.3 Programming Example

The following code illustrates a common production line operation of format → service pack update → program.

```
>>uniflashCLI.bat -config "c:\ti\uniflash_3.2\sessions\http_server\httpserver.usf" -
setOptions com=5 spPath="C:\ti\CC31xx_CC32xx_ServicePack_1.0.0.10.0\servicepack_1.0.0.10.0.bin" -
operations format servicePackUpdate program
```

```
***** Texas Instruments Universal Flash Programmer *****
```

```
> Configuring UniFlash with the following configuration: c:/ti/uniflash_3.2
/sessions/http_server/html/httpserver.usf
```

```
> Device: CC31x Flash Devices
> Interface: CC31x Flash Connections
```

```
> Template XML: C:\ti\uniflash_3.2
\sessions\http_server\httpserver_session\templates\CC3xxx_template.xml
> Token XML: C:\ti\uniflash_3.2
\sessions\http_server\httpserver_session\templates\CC3xxx_token.xml
> User setting [com]: 5
> User setting [spPath]: C:/ti/CC31xx_CC32xx_ServicePack_1.0.0.10.0/servicepack_1.0.0.10.0.bin
```

```
> Executing operation: format
> Relevant settings:
    Capacity: 1MB
INFO: > Executing Operation: Connect
INFO: setting break signal
INFO: detecting FTDI for device reset
INFO: connection succeeded
INFO: getting storage list
INFO: > Executing Operation: Init
INFO: reading version info
INFO: DEVICE CC3100 ES1.33
INFO: reading version info
```

```
Progress (SRAM): 16%
Progress (SRAM): 24%
Progress (SRAM): 41%
Progress (SRAM): 57%
Progress (SRAM): 65%
Progress (SRAM): 82%
Progress (SRAM): 98%
Progress (SRAM): 100%
INFO: > Executing Operation: Format
INFO: Erase storage SFLASH
INFO: erase storage succeeded
INFO: erase storage completed
> Finish Executing operation: format

> Executing operation: servicePackUpdate
> Relevant settings:
    Service Pack File: C:/ti/CC31xx_CC32xx_ServicePack_1.0.0.10.0/servicepack_1.0.0.10.0.bin
INFO: > Executing Operation: ServicePackProgramming
INFO: Path to the service pack file:
C:/ti/CC31xx_CC32xx_ServicePack_1.0.0.10.0/servicepack_1.0.0.10.0.bin
INFO: reading version info
INFO: CC3100R Device detected.
INFO: NWP/MAC/PHY Version from Service Pack:
INFO: NWP Patch version: 2.4.0.2
INFO: MAC Patch version: 1.3.0.1
INFO: PHY Patch version: 1.0.3.34
INFO: Downloading file "/sys/servicepack.ucf" with size 10100
Progress (/sys/servicepack.ucf): 40%
Progress (/sys/servicepack.ucf): 81%
Progress (/sys/servicepack.ucf): 100%
INFO:

New Token is 0x2B4874A0
INFO: Download complete
> Finish Executing operation: servicePackProgramming

> Executing operation: program
INFO: > Executing Operation: Program
INFO: > File name: /sys/mcuimg.bin, Update: true, Erase: true
INFO: > Erase File: /sys/mcuimg.bin
INFO: erasing file "/sys/mcuimg.bin"
INFO: deleting file "/sys/mcuimg.bin"
INFO: erase file completed
INFO: > Size of file = 37020
INFO: > Update File: /sys/mcuimg.bin
INFO: Downloading file "/sys/mcuimg.bin" with size 37020
Progress (/sys/mcuimg.bin): 11%
Progress (/sys/mcuimg.bin): 22%
Progress (/sys/mcuimg.bin): 33%
Progress (/sys/mcuimg.bin): 44%
Progress (/sys/mcuimg.bin): 55%
Progress (/sys/mcuimg.bin): 66%
Progress (/sys/mcuimg.bin): 77%
Progress (/sys/mcuimg.bin): 88%
Progress (/sys/mcuimg.bin): 99%
Progress (/sys/mcuimg.bin): 100%
INFO:

New Token is 0x0
INFO: Download complete
INFO: Verifying Data...
INFO: get file
Progress (/sys/mcuimg.bin): 11%
Progress (/sys/mcuimg.bin): 22%
Progress (/sys/mcuimg.bin): 33%
Progress (/sys/mcuimg.bin): 44%
```

```
Progress (/sys/mcuimg.bin): 55%
Progress (/sys/mcuimg.bin): 66%
Progress (/sys/mcuimg.bin): 77%
Progress (/sys/mcuimg.bin): 88%
Progress (/sys/mcuimg.bin): 99%
Progress (/sys/mcuimg.bin): 100%
INFO: Done. Reading 37020 bytes
INFO:
```

Verification OK

```
INFO: > Updated Token value: 0x0
INFO: > File name: /cert/ca.pem, Update: false, Erase: false
INFO: > File name: /cert/client.pem, Update: false, Erase: false
INFO: > File name: /cert/private.key, Update: false, Erase: false
INFO: > File name: /sys/macadd.bin, Update: true, Erase: false
INFO: > Size of file = 6
INFO: > Update File: /sys/macadd.bin
INFO: Downloading file "/sys/macadd.bin" with size 6
Progress (/sys/macadd.bin): 100%
INFO:
```

New Token is 0x0

```
INFO: Download complete
INFO: Verifying Data...
INFO: get file
Progress (/sys/macadd.bin): 100%
INFO: Done. Reading 6 bytes
INFO:
```

Verification OK

```
INFO: > Updated Token value: 0x0
INFO: > File name: /sys/mode.cfg, Update: true, Erase: false
INFO: > Size of file = 80
INFO: > Update File: /sys/mode.cfg
INFO: Downloading file "/sys/mode.cfg" with size 80
Progress (/sys/mode.cfg): 100%
INFO:
```

New Token is 0x0

```
INFO: Download complete
INFO: Verifying Data...
INFO: get file
Progress (/sys/mode.cfg): 100%
INFO: Done. Reading 80 bytes
INFO:
```

Verification OK

```
INFO: > Updated Token value: 0x0
INFO: > File name: /sys/ipcfg.ini, Update: true, Erase: false
INFO: > Size of file = 200
INFO: > Update File: /sys/ipcfg.ini
INFO: Downloading file "/sys/ipcfg.ini" with size 200
Progress (/sys/ipcfg.ini): 100%
INFO:
```

New Token is 0x0

```
INFO: Download complete
INFO: Verifying Data...
INFO: get file
Progress (/sys/ipcfg.ini): 100%
INFO: Done. Reading 200 bytes
INFO:
```

Verification OK

```
INFO: > Updated Token value: 0x0
INFO: > File name: /sys/ap.cfg, Update: false, Erase: false
```

```
INFO: > File name: /sys/devname.cfg, Update: false, Erase: false
INFO: > File name: /sys/mdns.cfg, Update: false, Erase: false
INFO: > File name: /sys/dhcpsrv.cfg, Update: false, Erase: false
INFO: > File name: /sys/httpdrv.cfg, Update: false, Erase: false
INFO: > File name: /sys/pref.net, Update: false, Erase: false
INFO: > File name: /sys/smartconfigkeys.cfg, Update: false, Erase: false
INFO: > File name: /sys/stacfg.ini, Update: true, Erase: false
INFO: > Size of file = 104
INFO: > Update File: /sys/stacfg.ini
INFO: Downloading file "/sys/stacfg.ini" with size 104
Progress (/sys/stacfg.ini): 100%
INFO:
```

```
New Token is 0x0
INFO: Download complete
INFO: Verifying Data...
INFO: get file
Progress (/sys/stacfg.ini): 100%
INFO: Done. Reading 104 bytes
INFO:
```

```
Verification OK
INFO: > Updated Token value: 0x0
INFO: > File name: /sys/p2p.cfg, Update: false, Erase: false
INFO: > File name: /sys/pmcfg.ini, Update: true, Erase: false
INFO: > Size of file = 10
INFO: > Update File: /sys/pmcfg.ini
INFO: Downloading file "/sys/pmcfg.ini" with size 10
Progress (/sys/pmcfg.ini): 100%
INFO:
```

```
New Token is 0x0
INFO: Download complete
INFO: Verifying Data...
INFO: get file
Progress (/sys/pmcfg.ini): 100%
INFO: Done. Reading 10 bytes
INFO:
```

```
Verification OK
INFO: > Updated Token value: 0x0
INFO: > File name: www/main.html, Update: true, Erase: true
INFO: > Erase File: www/main.html
INFO: erasing file "www/main.html"
INFO: deleting file "www/main.html"
INFO: erase file completed
INFO: > Size of file = 2714
INFO: > Update File: www/main.html
INFO: Downloading file "www/main.html" with size 2714
Progress (www/main.html): 100%
INFO:
```

```
New Token is 0x0
INFO: Download complete
INFO: Verifying Data...
INFO: get file
Progress (www/main.html): 100%
INFO: Done. Reading 2714 bytes
INFO:
```

```
Verification OK
INFO: > Updated Token value: 0x0
INFO: > File name: www/led_demo.html, Update: true, Erase: true
INFO: > Erase File: www/led_demo.html
INFO: erasing file "www/led_demo.html"
INFO: deleting file "www/led_demo.html"
```

```
INFO: erase file completed
INFO: > Size of file = 2573
INFO: > Update File: www/led_demo.html
INFO: Downloading file "www/led_demo.html" with size 2573
Progress (www/led_demo.html): 100%
INFO:

New Token is 0x0
INFO: Download complete
INFO: Verifying Data...
INFO: get file
Progress (www/led_demo.html): 100%
INFO: Done. Reading 2573 bytes
INFO:

Verification OK
INFO: > Updated Token value: 0x0
INFO: > File name: www/images/demo-lightswitch.jpg, Update: true, Erase: true
INFO: > Erase File: www/images/demo-lightswitch.jpg
INFO: erasing file "www/images/demo-lightswitch.jpg"
INFO: deleting file "www/images/demo-lightswitch.jpg"
INFO: erase file completed
INFO: > Size of file = 2404
INFO: > Update File: www/images/demo-lightswitch.jpg
INFO: Downloading file "www/images/demo-lightswitch.jpg" with size 2404
Progress (www/images/demo-lightswitch.jpg): 100%
INFO:

New Token is 0x0
INFO: Download complete
INFO: Verifying Data...
INFO: get file
Progress (www/images/demo-lightswitch.jpg): 100%
INFO: Done. Reading 2404 bytes
INFO:

Verification OK
INFO: > Updated Token value: 0x0
> Finish Executing operation: program
INFO: > Executing Operation: Disconnect
<End of command>
```

## 10 Configuration File Support

### 10.1 Overview

Configuration file is a binary file composed from a collection of parameters. Not all offsets in a configuration file are configurable by the user. The parameters that may be configured by the user are listed in CC3xxx\_cfg\_parameters.xml file. Uniflash utility then uses this XML file and places the content in the appropriate file system templates.

All parameters are exposed to the user via GUI interface and are further divided into logical groups. Consequently, the user is guided on what parameters need to be configured for each use case. Additionally, Uniflash implement internal sanity checks on different levels to make sure these parameters are valid and also checking conditional dependencies between parameters. For example, IP addresses and subnet masks are analyzed for validity.

## 10.2 Configuration Groups – Parameters' Description

### 10.2.1 Device Role

The following table lists all of the parameters applicable for Device Role group.

<b>Parameter</b>	Device Role
<b>Description</b>	WiFi role of the device
<b>Options</b>	Single selection between Station, AP and Peer2Peer Device
<b>Dependencies</b>	None

### 10.2.2 Station

The following tables list all of the parameters applicable for Station group.

<b>Parameter</b>	Mac Address
<b>Description</b>	device MAC address
<b>Options</b>	6 bytes valid MAC address written in HEX format and separated by ':' or '-', (12:34:56:78:90:ab). Note that it is restricted to set the LSB of the most significant byte
<b>Dependencies</b>	None

<b>Parameter</b>	Station Tx Power Level
<b>Description</b>	Back-off number from maximum power. Applicable for Station and Peer2Peer client roles only
<b>Options</b>	Number in range 0-15, dB units
<b>Dependencies</b>	None

<b>Parameter</b>	Power Policy
<b>Description</b>	Power policy of the device
<b>Options</b>	Single selection between PM_NORMAL, PM_ACTIVE and PM_LONG_SLEEP
<b>Dependencies</b>	None

<b>Parameter</b>	Long Sleep Interval
<b>Description</b>	Denotes the desired sleep interval between two consecutive wakeups for beacon reception
<b>Options</b>	Interval in mSec units
<b>Dependencies</b>	Applies only when Power Policy is set to PM_LONG_SLEEP

<b>Parameter</b>	Scan Channel Mask
<b>Description</b>	Channels mask during scan process
<b>Options</b>	Multiple choice between Channel1-Channel13
<b>Dependencies</b>	None

<b>Parameter</b>	Scan RSSI Threshold
<b>Description</b>	RSSI threshold during scan process
<b>Options</b>	Negative number in dBm units, represented as 2's complement
<b>Dependencies</b>	None

<b>Parameter</b>	Station Addressing Scheme
<b>Description</b>	Setting static IP parameters or get it dynamically via DHCP
<b>Options</b>	Static or Dynamic. Applicable to station and Peer2Peer client roles only
<b>Dependencies</b>	None

<b>Parameter</b>	Station IP Address
<b>Description</b>	IP address
<b>Options</b>	Valid IP address separated by '.', (192.168.1.1). Applicable to station and Peer2Peer client roles only. Note that Station IP and Station Default Gateway must reside on the same network.
<b>Dependencies</b>	Applies only when Station Addressing Scheme is set to Static

<b>Parameter</b>	Station Subnet Mask
<b>Description</b>	Subnet mask
<b>Options</b>	Valid subnet mask separated by '.', (255.255.255.0). Applicable to station and Peer2Peer client roles only
<b>Dependencies</b>	Applies only when Station Addressing Scheme is set to Static

<b>Parameter</b>	Station Default Gateway
<b>Description</b>	Default gateway
<b>Options</b>	Valid IP address separated by '.', (192.168.1.254). Applicable to station and Peer2Peer client roles only. Note that Station IP and Station Default Gateway must reside on the same network.
<b>Dependencies</b>	Applies only when Station Addressing Scheme is set to Static

<b>Parameter</b>	Station DNS Server
<b>Description</b>	DNS Server
<b>Options</b>	Valid IP address separated by '.', (192.168.1.254). Applicable to station and Peer2Peer client roles only
<b>Dependencies</b>	Applies only when Station Addressing Scheme is set to Static

<b>Parameter</b>	Station Network Applications
<b>Description</b>	Enabled network applications
<b>Options</b>	Multiple choices between HTTP server, mDNS client and Device configuration. Applicable to station role only
<b>Dependencies</b>	None

### 10.2.3 Access Point (AP)

The following tables list all of the parameters applicable for AP group.

<b>Parameter</b>	Mac Address
<b>Description</b>	device MAC address
<b>Options</b>	6 bytes valid MAC address written in HEX format and separated by ':' or '-', (12:34:56:78:90:ab). Note that it is restricted to set the LSB of the most significant byte
<b>Dependencies</b>	None

<b>Parameter</b>	SSID
<b>Description</b>	SSID indicates the name of the network in AP role
<b>Options</b>	SSID is a string up to 32 bytes
<b>Dependencies</b>	None

<b>Parameter</b>	Security Type
<b>Description</b>	WiFi Security Type in AP role
<b>Options</b>	Single selection between Open, WEP and WPA
<b>Dependencies</b>	None

<b>Parameter</b>	WEP Key Format
<b>Description</b>	WEP key format in AP role
<b>Options</b>	Either Hexadecimal or ASCII
<b>Dependencies</b>	None

<b>Parameter</b>	Password
<b>Description</b>	WiFi password in AP role
<b>Options</b>	For WEP, it is either 5 bytes or 13 bytes, Hexadecimal or ASCII format. For WPA, it is 8-63 bytes
<b>Dependencies</b>	Applies only when Security Type is set to WEP or WPA

<b>Parameter</b>	Country Code
<b>Description</b>	Country Code for regulatory purposes
<b>Options</b>	Single choice between US, EU and JP
<b>Dependencies</b>	None

<b>Parameter</b>	Channel
<b>Description</b>	Operating WiFi channel
<b>Options</b>	Single choice between Channel x, where x denotes the channel number 1-13
<b>Dependencies</b>	None

<b>Parameter</b>	Hidden SSID
<b>Description</b>	Setting this option makes SSID not being broadcast in AP role
<b>Options</b>	Boolean, False or True
<b>Dependencies</b>	None

<b>Parameter</b>	Device Domain Name
<b>Description</b>	In AP role, part of the default SSID
<b>Options</b>	String up to 33 bytes
<b>Dependencies</b>	None

<b>Parameter</b>	AP Tx Power Level
<b>Description</b>	Back-off number from maximum power. Applicable for AP and Peer2Peer group owner roles only
<b>Options</b>	Number in range 0-15, dB units
<b>Dependencies</b>	None

<b>Parameter</b>	Ignore Force AP
<b>Description</b>	Applicable for CC3100 only. In case AP is forced via GPIO, setting this option makes the device boot as open AP
<b>Options</b>	Boolean, False or True
<b>Dependencies</b>	None

<b>Parameter</b>	Beacon Interval
<b>Description</b>	Beacon Interval is the interval between AP beacons notifications
<b>Options</b>	It is represented in TUs where each TU (Time Unit) is 1024 $\mu$ Sec
<b>Dependencies</b>	None

<b>Parameter</b>	DTIM Interval
<b>Description</b>	DTIM Interval is the interval between beacons containing broadcast information
<b>Options</b>	It is represented in Beacon Intervals
<b>Dependencies</b>	None

<b>Parameter</b>	AP Addressing Scheme
<b>Description</b>	Setting static IP parameters or get it dynamically via DHCP
<b>Options</b>	Static or Dynamic. Applicable to AP and Peer2Peer group owner roles only
<b>Dependencies</b>	None

<b>Parameter</b>	AP IP Address
<b>Description</b>	IP address
<b>Options</b>	Valid IP address separated by '.', (192.168.1.1). Applicable to AP and Peer2Peer group owner roles only. Note that AP IP and AP Default Gateway must reside on the same network.
<b>Dependencies</b>	None

<b>Parameter</b>	AP Subnet Mask
<b>Description</b>	Subnet mask
<b>Options</b>	Valid subnet mask separated by '.', (255.255.255.0). Applicable to AP and Peer2Peer group owner roles only
<b>Dependencies</b>	None

<b>Parameter</b>	AP Default Gateway
<b>Description</b>	Default gateway
<b>Options</b>	Valid IP address separated by '.', (192.168.1.254). Applicable to AP and Peer2Peer group owner roles only. Note that AP IP and AP Default Gateway must reside on the same network.
<b>Dependencies</b>	Applies only when AP Addressing Scheme is set to Static

<b>Parameter</b>	AP DNS Server
<b>Description</b>	DNS Server
<b>Options</b>	Valid IP address separated by '.', (192.168.1.254). Applicable to AP and Peer2Peer group owner roles only
<b>Dependencies</b>	Applies only when AP Addressing Scheme is set to Static

<b>Parameter</b>	AP Network Applications
<b>Description</b>	Enabled network applications
<b>Options</b>	Multiple choices between HTTP server, mDNS client, DNS Server and Device configuration. Applicable to AP role only
<b>Dependencies</b>	None

### 10.2.4 Peer 2 Peer (P2P)

The following tables list all of the parameters applicable for P2P group.

<b>Parameter</b>	Mac Address
<b>Description</b>	device MAC address
<b>Options</b>	6 bytes valid MAC address written in HEX format and separated by ':' or '-', (12:34:56:78:90:ab). Note that it is restricted to set the LSB of the most significant byte
<b>Dependencies</b>	None

<b>Parameter</b>	Station Tx Power Level
<b>Description</b>	Back-off number from maximum power. Applicable for Station and Peer2Peer client roles only
<b>Options</b>	Number in range 0-15, dB units.
<b>Dependencies</b>	None

<b>Parameter</b>	AP Tx Power Level
<b>Description</b>	Back-off number from maximum power. Applicable for Station and Peer2Peer client roles only
<b>Options</b>	Number in range 0-15, dB units.
<b>Dependencies</b>	None

<b>Parameter</b>	Power Policy
<b>Description</b>	Power policy of the device
<b>Options</b>	Single selection between PM_NORMAL, PM_ACTIVE and PM_LONG_SLEEP
<b>Dependencies</b>	None

<b>Parameter</b>	Long Sleep Interval
<b>Description</b>	Denotes the desired sleep interval between two consecutive wakeups for beacon reception
<b>Options</b>	Interval in mSec units
<b>Dependencies</b>	Applies only when Power Policy is set to PM_LONG_SLEEP

<b>Parameter</b>	Device URN Name
<b>Description</b>	In Peer2Peer Device role, represents the device name
<b>Options</b>	String up to 33 bytes
<b>Dependencies</b>	None

<b>Parameter</b>	Peer2Peer Oper Channels
<b>Description</b>	The device operation band following Peer2Peer connection
<b>Options</b>	Should be one of the social channels, 1, 6 or 11
<b>Dependencies</b>	None

<b>Parameter</b>	Peer2Peer Intent
<b>Description</b>	Score indicating how eager the device is to become Peer2Peer client vs. Peer2Peer Group Owner
<b>Options</b>	Number between 0-15 where 0 indicates Client and 15 indicates Group Owner
<b>Dependencies</b>	None

<b>Parameter</b>	Peer2Peer Negotiation Initiator
<b>Description</b>	Policy for initialing negotiation request once a preferred device found
<b>Options</b>	Single selection between Active, Passive and Random Back off
<b>Dependencies</b>	None

<b>Parameter</b>	Station Addressing Scheme
<b>Description</b>	Setting static IP parameters or get it dynamically via DHCP
<b>Options</b>	Static or Dynamic. Applicable to station and Peer2Peer client roles only
<b>Dependencies</b>	None

<b>Parameter</b>	Station IP Address
<b>Description</b>	IP address
<b>Options</b>	Valid IP address separated by '.', (192.168.1.1). Applicable to station and Peer2Peer client roles only. Note that Station IP and Station Default Gateway must reside on the same network.
<b>Dependencies</b>	Applies only when Station Addressing Scheme is set to Static

<b>Parameter</b>	Station Subnet Mask
<b>Description</b>	Subnet mask
<b>Options</b>	Valid subnet mask separated by '.', (255.255.255.0). Applicable to station and Peer2Peer client roles only
<b>Dependencies</b>	Applies only when Station Addressing Scheme is set to Static

<b>Parameter</b>	Station Default Gateway
<b>Description</b>	Default gateway
<b>Options</b>	Valid IP address separated by '.', (192.168.1.254). Applicable to station and Peer2Peer client roles only. Note that Station IP and Station Default Gateway must reside on the same network.
<b>Dependencies</b>	Applies only when Station Addressing Scheme is set to Static

<b>Parameter</b>	Station DNS Server
<b>Description</b>	DNS Server
<b>Options</b>	Valid IP address separated by '.', (192.168.1.254). Applicable to station and Peer2Peer client roles only
<b>Dependencies</b>	Applies only when Station Addressing Scheme is set to Static

<b>Parameter</b>	AP Addressing Scheme
<b>Description</b>	Setting static IP parameters or get it dynamically via DHCP
<b>Options</b>	Static or Dynamic. Applicable to AP and Peer2Peer group owner roles only
<b>Dependencies</b>	None

<b>Parameter</b>	AP IP Address
<b>Description</b>	IP address
<b>Options</b>	Valid IP address separated by '.', (192.168.1.1). Applicable to AP and Peer2Peer group owner roles only. Note that AP IP and AP Default Gateway must reside on the same network.
<b>Dependencies</b>	Applies only when AP Addressing Scheme is set to Static

<b>Parameter</b>	AP Subnet Mask
<b>Description</b>	Subnet mask
<b>Options</b>	Valid subnet mask separated by '.', (255.255.255.0). Applicable to AP and Peer2Peer group owner roles only
<b>Dependencies</b>	Applies only when AP Addressing Scheme is set to Static

<b>Parameter</b>	AP Default Gateway
<b>Description</b>	Default gateway
<b>Options</b>	Valid IP address separated by '.', (192.168.1.254). Applicable to AP and Peer2Peer group owner roles only. Note that AP IP and AP Default Gateway must reside on the same network.
<b>Dependencies</b>	Applies only when AP Addressing Scheme is set to Static

<b>Parameter</b>	AP DNS Server
<b>Description</b>	DNS Server
<b>Options</b>	Valid IP address separated by '.', (192.168.1.254). Applicable to AP and Peer2Peer group owner roles only.
<b>Dependencies</b>	Applies only when AP Addressing Scheme is set to Static

<b>Parameter</b>	Peer2Peer Client Network Applications
<b>Description</b>	Enabled network applications
<b>Options</b>	Multiple choices between HTTP server, mDNS client and Device configuration. Applicable to P2P Client role only
<b>Dependencies</b>	None

<b>Parameter</b>	Peer2Peer Group Owner Network Applications
<b>Description</b>	Enabled network applications
<b>Options</b>	Multiple choices between HTTP server, mDNS client, DNS Server and Device configuration. Applicable to P2P Group Owner role only
<b>Dependencies</b>	None

## 10.2.5 Profiles

The following tables list all of the parameters applicable for Profiles group.

<b>Parameter</b>	Auto Start Enable
<b>Description</b>	Setting this option enables connecting according to stored Profiles
<b>Options</b>	State, Enable or Disable
<b>Dependencies</b>	None

<b>Parameter</b>	Fast Connect Enable
<b>Description</b>	Setting this option enables Fast Connection
<b>Options</b>	State, Enable or Disable
<b>Dependencies</b>	None

<b>Parameter</b>	Auto Smart Config
<b>Description</b>	State of Smart Config upon init
<b>Options</b>	State, Enable or Disable
<b>Dependencies</b>	None

<b>Parameter</b>	Connect To Open AP
<b>Description</b>	Setting this option enables connecting to any open AP
<b>Options</b>	State, Enable or Disable
<b>Dependencies</b>	None

<b>Parameter</b>	Connect To Any Peer 2 Peer
<b>Description</b>	Setting this option enables connecting to any Peer 2 Peer device
<b>Options</b>	State, Enable or Disable
<b>Dependencies</b>	None

For each Profile 1-7, the following parameters can be configured (replace x with the profile number).

<b>Parameter</b>	Profile x SSID
<b>Description</b>	Peer SSID in station role or remote device in Peer2Peer role
<b>Options</b>	String up to 32 bytes
<b>Dependencies</b>	None

<b>Parameter</b>	Profile x Priority
<b>Description</b>	Profile x Priority
<b>Options</b>	Number between 0-255
<b>Dependencies</b>	None

<b>Parameter</b>	Profile x BSSID set
<b>Description</b>	Profile x peer BSSID mechanism. Setting this option make use of BSSID (and SSID) during WiFi connection while in station role. In Peer2Peer role, it denotes the peer MAC address
<b>Options</b>	State, Enable or Disable
<b>Dependencies</b>	None

<b>Parameter</b>	Profile x BSSID
<b>Description</b>	Profile x peer BSSID in station mode or Peer2Peer remote device MAC address
<b>Options</b>	6 bytes valid MAC address written in HEX format and separated by ':' or '-', (12:34:56:78:90:ab)
<b>Dependencies</b>	BSSID set is enabled

<b>Parameter</b>	Profile x Security Type
<b>Description</b>	Profile x security type
<b>Options</b>	Single selection between Open, WEP, WPA, Peer2Peer and Enterprise
<b>Dependencies</b>	None

<b>Parameter</b>	Profile x P2p Security
<b>Description</b>	Profile x Peer2Peer security type
<b>Options</b>	Single selection between P2P_NONE, P2P_PBC, P2P_PIN_KEYPAD and P2P_PIN_DISPLAY
<b>Dependencies</b>	Security Type is Peer2Peer

<b>Parameter</b>	Profile x P2p Pin
<b>Description</b>	Profile x Peer2Peer PIN code
<b>Options</b>	Should be 8 bytes long
<b>Dependencies</b>	Security Type is Peer2Peer and P2p Security is either P2P_PIN_KEYPAD or P2P_PIN_DISPLAY

<b>Parameter</b>	Profile x WEP Key Format
<b>Description</b>	Profile x WEP key format
<b>Options</b>	Either Hexadecimal or ASCII
<b>Dependencies</b>	Security Type is WEP

<b>Parameter</b>	Profile x WEP Key
<b>Description</b>	Profile x WEP key
<b>Options</b>	Either 5 bytes or 13 bytes key, Hexadecimal or ASCII format
<b>Dependencies</b>	Security Type is WEP

<b>Parameter</b>	Profile x WPA Key
<b>Description</b>	Profile x WPA passphrase
<b>Options</b>	It is 8-63 bytes long
<b>Dependencies</b>	Security Type is WPA

<b>Parameter</b>	Profile x EAP Type
<b>Description</b>	Profile x EAP type
<b>Options</b>	Single selection between TLS, TTLS, PEAP and FAST
<b>Dependencies</b>	Security Type is Enterprise

<b>Parameter</b>	Profile x User Identity
<b>Description</b>	Profile x user identity
<b>Options</b>	Up to 32 bytes long
<b>Dependencies</b>	Security Type is Enterprise

<b>Parameter</b>	Profile x Anonymous Identity
<b>Description</b>	Profile x anonymous identity. Does not apply for TLS EAP Type
<b>Options</b>	Up to 32 bytes long
<b>Dependencies</b>	Security Type is Enterprise and EAP Type is not TLS

<b>Parameter</b>	Profile x User Password
<b>Description</b>	Profile x user password. Does not apply for TLS EAP Type
<b>Options</b>	Up to 32 bytes long
<b>Dependencies</b>	Security Type is Enterprise and one of the following configurations exist: <ul style="list-style-type: none"> <li>EAP Type is FAST</li> <li>EAP Type is PEAP and EAP Sub Type is either PSK or MSCHAPv2</li> <li>EAP Type is TTLS and EAP Sub Type is either PSK or MSCHAPv2</li> </ul>

<b>Parameter</b>	Profile x EAP Sub Type
<b>Description</b>	Profile x EAP sub type
<b>Options</b>	Single selection between MSCHAPv2, TLS and PSK. Applies only if EAP Type is either TTLS or PEAP
<b>Dependencies</b>	Security Type is Enterprise and EAP Type is either TTLS or PEAP

<b>Parameter</b>	Profile x PEAP Version
<b>Description</b>	Profile x PEAP version
<b>Options</b>	Single selection between Version0 and Version1
<b>Dependencies</b>	Security Type is Enterprise and EAP Type is PEAP

<b>Parameter</b>	Profile x EAP Fast Provisioning
<b>Description</b>	Profile x EAP fast provisioning
<b>Options</b>	Single selection between Unauthenticated and Authenticated
<b>Dependencies</b>	Security Type is Enterprise and EAP Type is FAST

## 10.2.6 HTTP Server

The following tables list all of the parameters applicable for HTTP Server group.

<b>Parameter</b>	HTTP Server Port
<b>Description</b>	HTTP server TCP port
<b>Options</b>	Any valid TCP port number
<b>Dependencies</b>	None

<b>Parameter</b>	HTTP Server Access Rom Pages
<b>Description</b>	Setting this option allows the HTTP server an access to HTML pages placed in ROM
<b>Options</b>	State, Enable or Disable
<b>Dependencies</b>	None

<b>Parameter</b>	HTTP Server Auth
<b>Description</b>	HTTP server authentication
<b>Options</b>	State, Enable or Disable
<b>Dependencies</b>	None

<b>Parameter</b>	HTTP Server User
<b>Description</b>	HTTP server authenticated username
<b>Options</b>	User is a string up to 20 bytes
<b>Dependencies</b>	Applies only when AP HTTP Server Auth is set to Enable

<b>Parameter</b>	HTTP Server Password
<b>Description</b>	HTTP server authenticated password
<b>Options</b>	Password is a string up to 20 bytes
<b>Dependencies</b>	Applies only when AP HTTP Server Auth is set to Enable

<b>Parameter</b>	HTTP Server Realm
<b>Description</b>	HTTP server authenticated Realm
<b>Options</b>	Realm is a string up to 20 bytes
<b>Dependencies</b>	Applies only when AP HTTP Server Auth is set to Enable

### 10.2.7 DHCP Server

The following tables list all of the parameters applicable for DHCP Server group.

<b>Parameter</b>	DHCP Server Lease Time
<b>Description</b>	DHCP Lease time
<b>Options</b>	Time in seconds
<b>Dependencies</b>	None

<b>Parameter</b>	DHCP Server IP Start
<b>Description</b>	Start IP Address provided by the DHCP server
<b>Options</b>	Valid IP address separated by '.', e.g. 192.168.1.1
<b>Dependencies</b>	None

<b>Parameter</b>	DHCP Server IP End
<b>Description</b>	Last IP Address provided by the DHCP server
<b>Options</b>	Valid IP address separated by '.', e.g. 192.168.1.100
<b>Dependencies</b>	None

### 10.2.8 mDNS Client

The following tables list all of the parameters applicable for mDNS client group.

<b>Parameter</b>	Device URN Name
<b>Description</b>	Uniform Resource Name that is published on mDNS multicasts
<b>Options</b>	It is a string up to 33 bytes
<b>Dependencies</b>	None

For each mDNS client 1-5, following parameters can be configured (replace x with the client number):

<b>Parameter</b>	mDNS Client x Service Name
<b>Description</b>	mDNS service name
<b>Options</b>	Service name is a string up to 100 bytes
<b>Dependencies</b>	mDNS client x is enabled

<b>Parameter</b>	mDNS Client x Test Record
<b>Description</b>	mDNS service text record
<b>Options</b>	Service name is a string up to 255 bytes
<b>Dependencies</b>	mDNS client x is enabled

<b>Parameter</b>	mDNS Client x Port
<b>Description</b>	mDNS service port
<b>Options</b>	Valid UDP port number
<b>Dependencies</b>	mDNS client x is enabled

<b>Parameter</b>	mDNS Client x TTL
<b>Description</b>	mDNS service TTL
<b>Options</b>	TTL in seconds
<b>Dependencies</b>	mDNS client x is enabled

### 10.2.9 Smart Config

The following tables list all of the parameters applicable for Smart Config group.

<b>Parameter</b>	Default Group Key
<b>Description</b>	Default group AES key for Smart Config
<b>Options</b>	16 bytes long
<b>Dependencies</b>	None

<b>Parameter</b>	Group Id 1
<b>Description</b>	1st Group ID for Smart Config
<b>Options</b>	Number between 1-15
<b>Dependencies</b>	None

<b>Parameter</b>	Group Key 1
<b>Description</b>	1st group AES key for Smart Config
<b>Options</b>	16 bytes long
<b>Dependencies</b>	None

<b>Parameter</b>	Group Id 2
<b>Description</b>	2nd Group ID for Smart Config
<b>Options</b>	Number between 1-15
<b>Dependencies</b>	None

<b>Parameter</b>	Group Key 2
<b>Description</b>	2nd group AES key for Smart Config
<b>Options</b>	16 bytes long
<b>Dependencies</b>	None

## 10.3 Configuration Groups – GUI Interface

### 10.3.1 Device Role

#### 10.3.1.1 Restore to Default

At any phase during configuration, it is possible to revert and go back to default configuration. It is applied by clicking the Restore to Default button.

#### 10.3.1.2 Programming

Programming of the device is applied upon clicking the Program button. Another way is from the top bar by clicking Operation → Program. Upon pressing the Program button, Uniflash utility checks for Update checkbox. If Update is checked, programming to target device is executed.

#### 10.3.1.3 Generation of Binary Templates

Unlike programming that generates the binary templates and programs those to the target device, it is possible to just generate those binary templates offline. This option can be used for debug purposes but also to make an offline session that would be used as part of Image Programming. The latest is crucial since all configuration files need to pass integrity test before introduced into an image (since the image cannot apply any integrity test).

The configuration binaries can be generated via the top bar by clicking Operation → Generate Config Binaries.

#### 10.3.1.4 Configuration

Figure 15 shows a screenshot of all the Device Role parameters. In this screen, the following parameters are defined:

- DeviceRole is set to Station

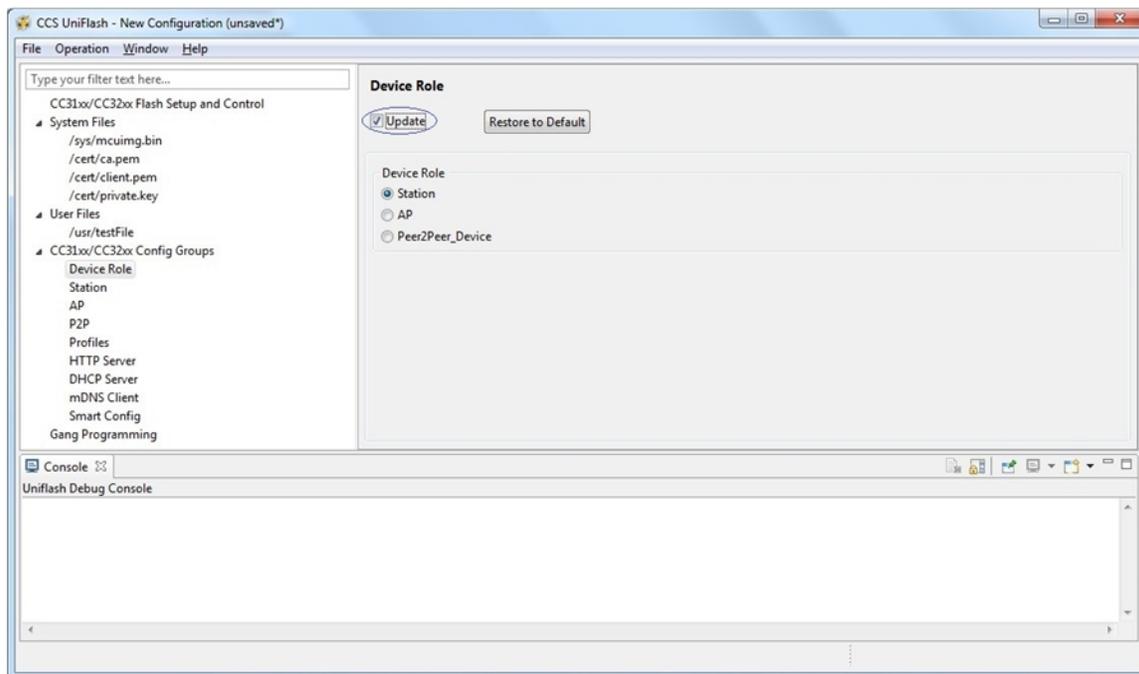


Figure 15. Device Role Configuration

### 10.3.2 Station

#### 10.3.2.1 Restore to Default

At any phase during configuration, it is possible to revert and go back to default configuration. It is applied by clicking the Restore to Default button.

#### 10.3.2.2 Programming

Programming of the device is applied upon clicking the Program button. Another way is from the top bar by clicking Operation → Program. Upon pressing the Program button, Uniflash utility checks for Update checkbox. If Update is checked, programming to target device is executed.

#### 10.3.2.3 Generation of Binary Templates

Unlike programming that generates the binary templates and programs those to the target device, it is possible just to generate those binary templates offline. This option can be used for debug purposes but also to make an offline session that would be used as part of Image Programming. The latest is crucial since all configuration files need to pass integrity test before introduced into an image (since the image cannot apply any integrity test).

The configuration binaries can be generated via the top bar by clicking Operation → Generate Config Binaries.

#### 10.3.2.4 Flow

Figure 16 illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. You may choose to go through configuration steps differently.

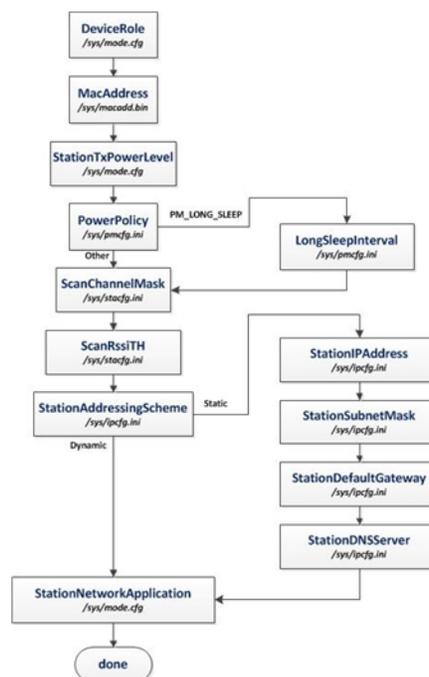


Figure 16. Station Flow

### 10.3.2.5 Configuration

Figure 17 shows a screenshot of all the Station parameters. Most parameters have default values in case the user decides to skip configuration. Mac Address parameter is the only one that does not have any default. Not setting this parameter is valid. If not set, the device burnt MAC address is used. In this screen, the following parameters are defined:

- MacAddress is not set. Default MAC address is the already burnt MAC address.
- Station Tx Power Level is set to default 0, (transmit at maximum power).
- Power Policy is set to default PM\_NORMAL
- Long Sleep Interval is not relevant for PM\_NORMAL
- ScanChannelMask is set to default where all 13 channels are scanned
- Scan Rssi Threshold is set to default -95 dBm. All received beacons/probe requests stronger than this level would be received by the device.
- Station Addressing Scheme is set to default Dynamic, (retrieve network parameters from DHCP server).
- Station IP Address, Station Subnet Mask, Station Default Gateway and Station DNS Server are all not relevant since Station Addressing Scheme is not set to Static.
- Station Network Application is set to default where HTTP Server and mDNS Client application are enabled.

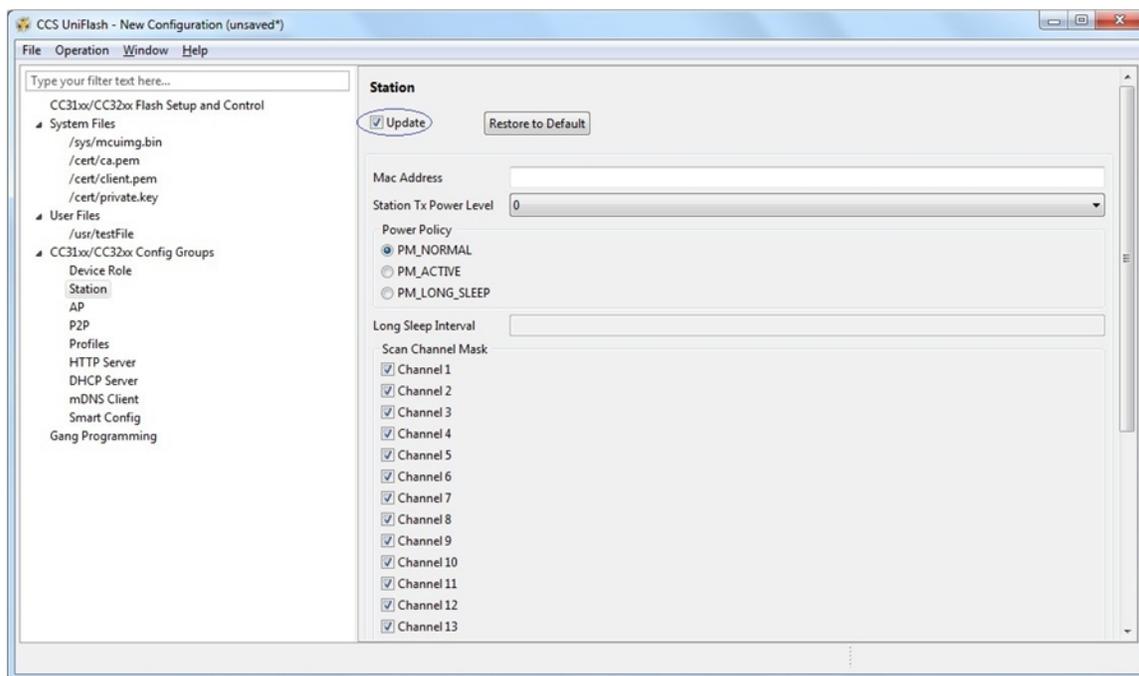


Figure 17. Station Configuration

## 10.3.3 Access Point (AP)

### 10.3.3.1 Restore to Default

At any phase during configuration, it is possible to revert and go back to the default configuration. It is applied by clicking the Restore to Default button.

### 10.3.3.2 Programming

Programming of the device is applied upon clicking the Program button. Another way is from the top bar by clicking Operation → Program. Upon pressing the Program button, the Uniflash utility checks for an Update checkbox. If Update is checked, programming to the target device is executed.

### 10.3.3.3 Generation of Binary Templates

Unlike programming that generates the binary templates and programs those to the target device, it is possible just to generate those binary templates offline. This option can be used for debug purposes but also to make an offline session that would be used as part of Image Programming. The latest is crucial since all configuration files need to pass integrity test before introduced into an image (since the image cannot apply any integrity test).

The configuration binaries can be generated via the top bar by clicking Operation → Generate Config Binaries.

### 10.3.3.4 Flow

Figure 18 illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.

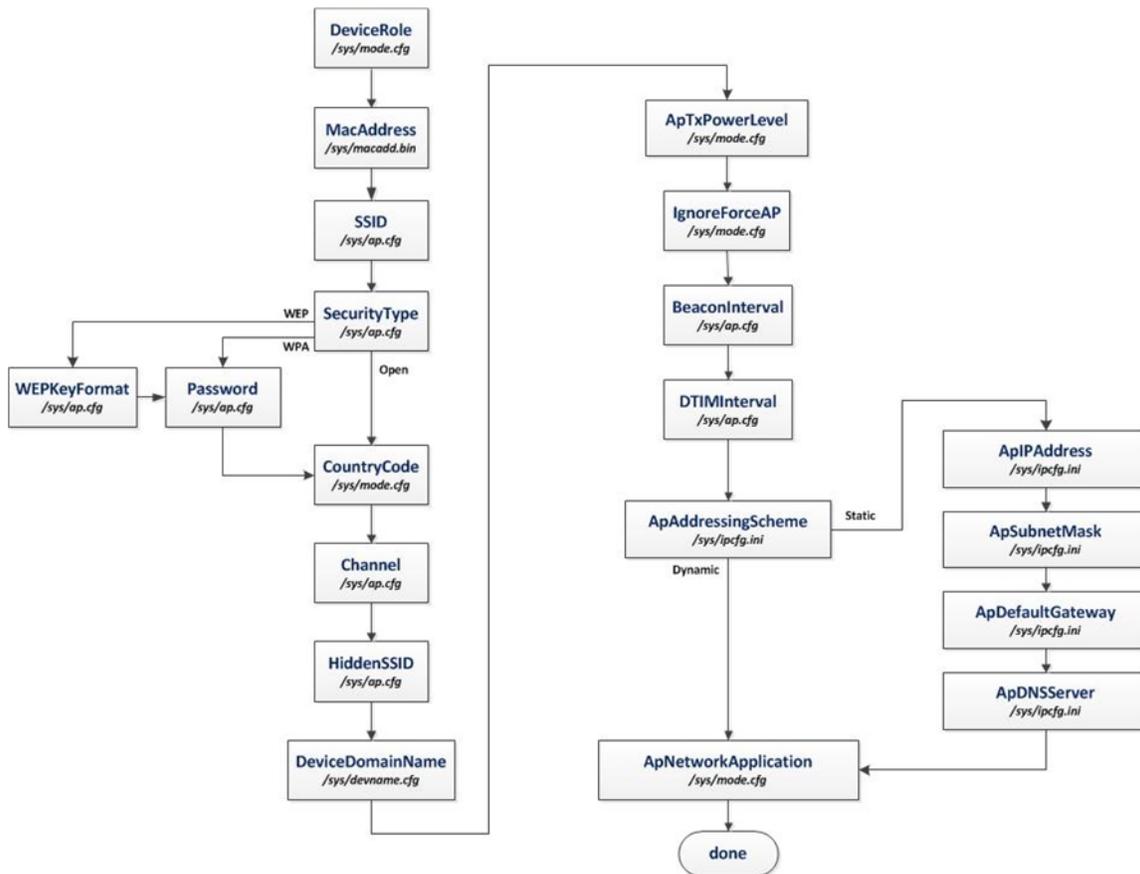


Figure 18. Ap Flow

### 10.3.3.5 Configuration

Figure 19 shows a screenshot of all the Station parameters. Most parameters have default values in case the user decides to skip configuration. Mac Address parameter is the only one that does not have any default. Not setting this parameter is valid. If not set, the device burnt MAC address is used. In this screen, the following parameters are defined:

- MacAddress is not set. Default MAC address is the already burnt MAC address.
- SSID is set to default mysimplelink
- Security Type is set to default Open
- Password is not relevant since Security Type is set to Open
- WEP Key Format is set to default ASCII
- Country Code is set to default US
- Channel is set to default, channel # 6
- Hidden SSID is set to default Disable
- Device domain Name is set to default mysimplelink.net
- AP Tx Power Level is set to default 0 (transmit at maximum power)
- Ignore Force AP is set to default Disable
- Beacon Interval is set to default 100 TU
- DTIM Interval is set to default 2 BI
- Ap Addressing Scheme is set to default Dynamic (retrieve network parameters from DHCP server)
- Ap IP Address is set to default 192.168.1.1
- Ap Subnet Mask is set to default 255.255.255.0
- Ap Default Gateway is set to default 192.168.1.1
- Ap Default Gateway is set to default 192.168.1.1.
- Ap Network Application is set to default where HTTP Server, DHCP Server and mDNS Client application are enabled.

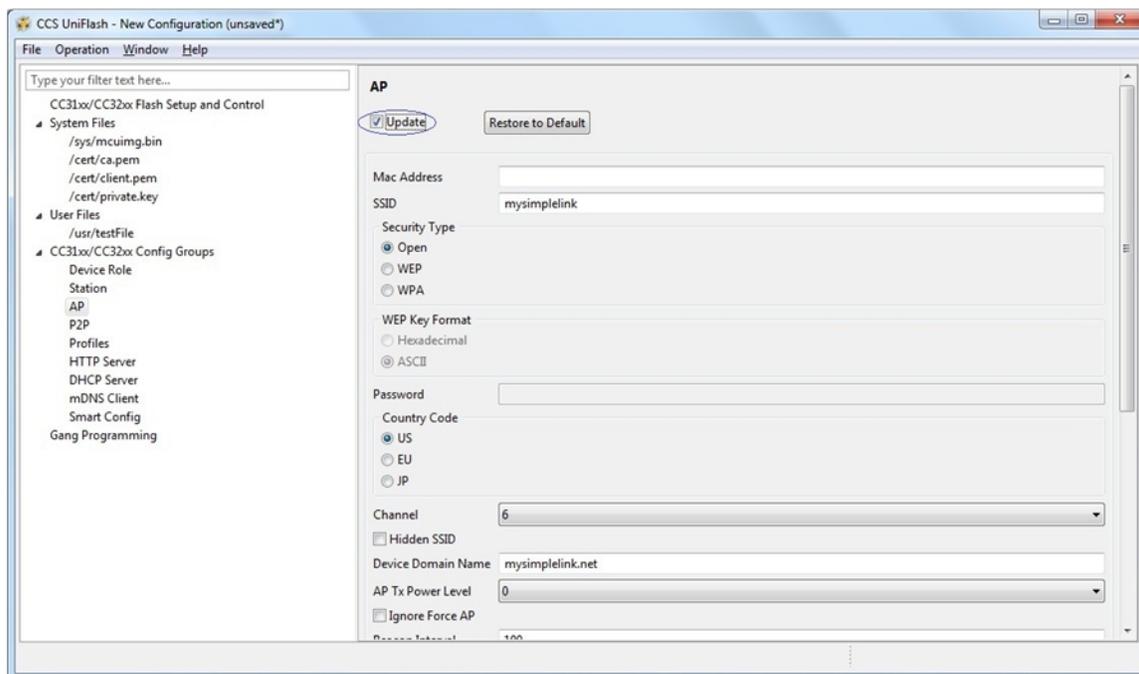


Figure 19. Ap Configuration

### 10.3.4 Peer 2 Peer (P2P)

#### 10.3.4.1 Restore to Default

At any phase during configuration, it is possible to revert and go back to default configuration. It is applied by clicking the Restore to Default button.

#### 10.3.4.2 Programming

Programming of the device is applied upon clicking the Program button. Another way is from the top bar by clicking Operation → Program. Upon pressing the Program button, Uniflash utility checks for Update checkbox. If Update is checked, programming to the target device is executed.

#### 10.3.4.3 Generation of Binary Templates

Unlike programming that generates the binary templates and programs those to the target device, it is possible just to generate those binary templates offline. This option can be used for debug purposes but also to make an offline session that would be used as part of Image Programming. The latest is crucial since all configuration files need to pass integrity test before introduced into an image (since the image cannot apply any integrity test).

The configuration binaries can be generated via the top bar by clicking Operation → Generate Config Binaries.

#### 10.3.4.4 Flow

Figure 20 illustrates the parameters' flow during configuration. It is highly advised to follow this flow, but it is not mandatory. The user may choose to go through configuration steps differently.

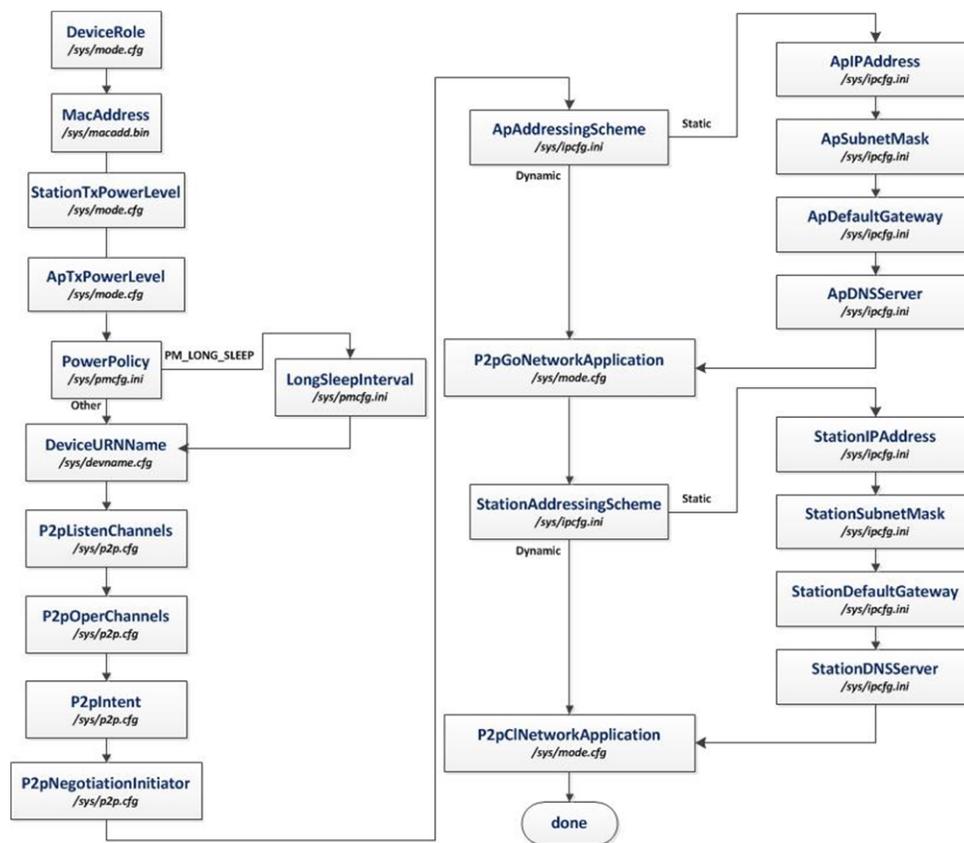


Figure 20. P2p Flow

### 10.3.4.5 Configuration

Figure 21 shows a screenshot of all the P2P parameters. Most parameters have default values in case the user decides to skip configuration. Mac Address parameter is the only one that does not have any default. Not setting this parameter is valid. If not set, the device burnt MAC address is used. In this screen, the following parameters are defined:

- MacAddress is not set. Default MAC address is the already burnt MAC address.
- Station Tx Power Level is set to default 0, (transmit at maximum power).
- AP Tx Power Level is set to default 0, (transmit at maximum power).
- Power Policy is set to default PM\_NORMAL.=
- Long Sleep Interval is not relevant for PM\_NORMAL
- Device URN Name is set to default mysimplelink
- Peer 2 Peer Listen Channels is set to default, channel # 1
- Peer 2 Peer Oper Channels is set to default, channel # 1
- Peer 2 Peer Intent is set to default 7
- Peer 2 Peer Negotiation Initiator is set to default Random
- Station Addressing Scheme is set to default Dynamic (retrieve network parameters from DHCP server).
- Station IP Address, Station Subnet Mask, Station Default Gateway and Station DNS Server are all not relevant since Station Addressing Scheme is not set to Static.
- Ap Addressing Scheme is set to default Dynamic (retrieve network parameters from DHCP server).
- Ap IP Address is set to default 192.168.1.1
- Ap Subnet Mask is set to default 255.255.255.0
- Ap Default Gateway is set to default 192.168.1.1.
- Ap DNS Server is set to default 192.168.1.1
- Station Network Application is set to default where HTTP Server and mDNS Client application are enabled.
- Ap Network Application is set to default where HTTP Server, DHCP Server and mDNS Client application are enabled.

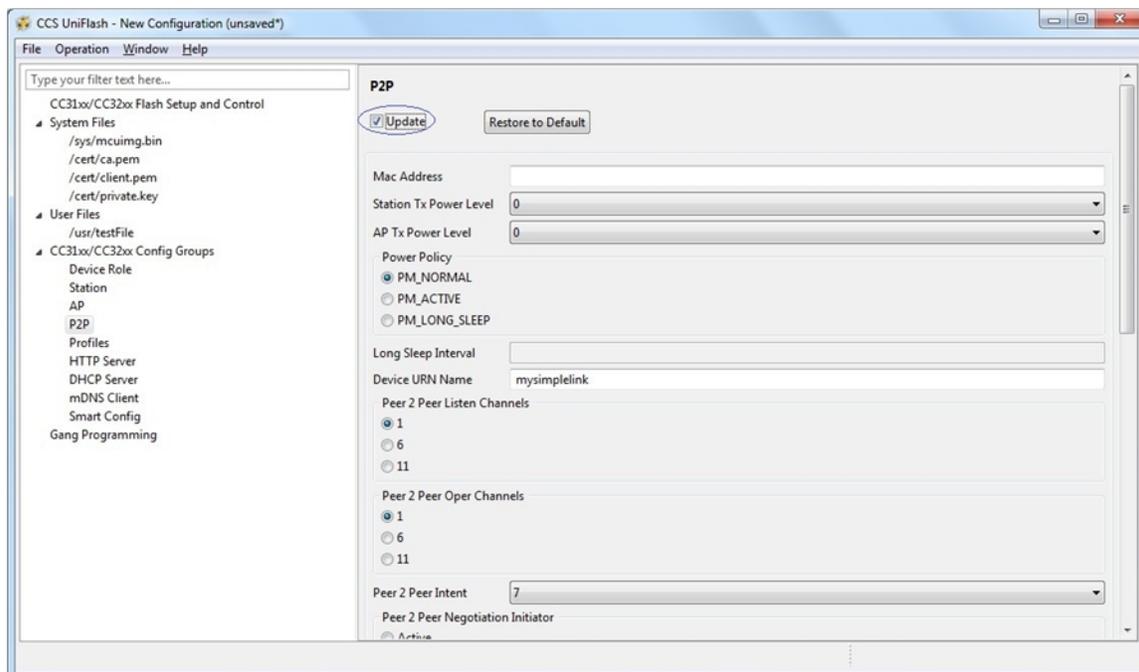


Figure 21. P2p Configuration

## 10.3.5 HTTP Server

### 10.3.5.1 Restore to Default

At any phase during configuration, it is possible to revert and go back to default configuration. It is applied by clicking the Restore to Default button.

### 10.3.5.2 Programming

Programming of the device is applied upon clicking the Program button. Another way is from the top bar by clicking Operation → Program. Upon pressing the Program button, Uniflash utility checks for Update checkbox. If Update is checked, programming to the target device is executed.

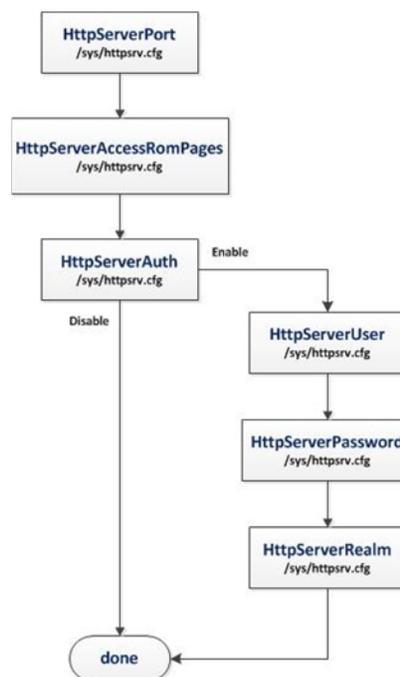
### 10.3.5.3 Generation of Binary Templates

Unlike programming that generates the binary templates and programs those to the target device, it is possible just to generate those binary templates offline. This option can be used for debug purposes but also to make an offline session that would be used as part of Image Programming. The latest is crucial since all configuration files need to pass integrity test before introduced into an image (since the image cannot apply any integrity test).

The configuration binaries can be generated via the top bar by clicking Operation->Generate Config Binaries.

### 10.3.5.4 Flow

Figure 22 illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.

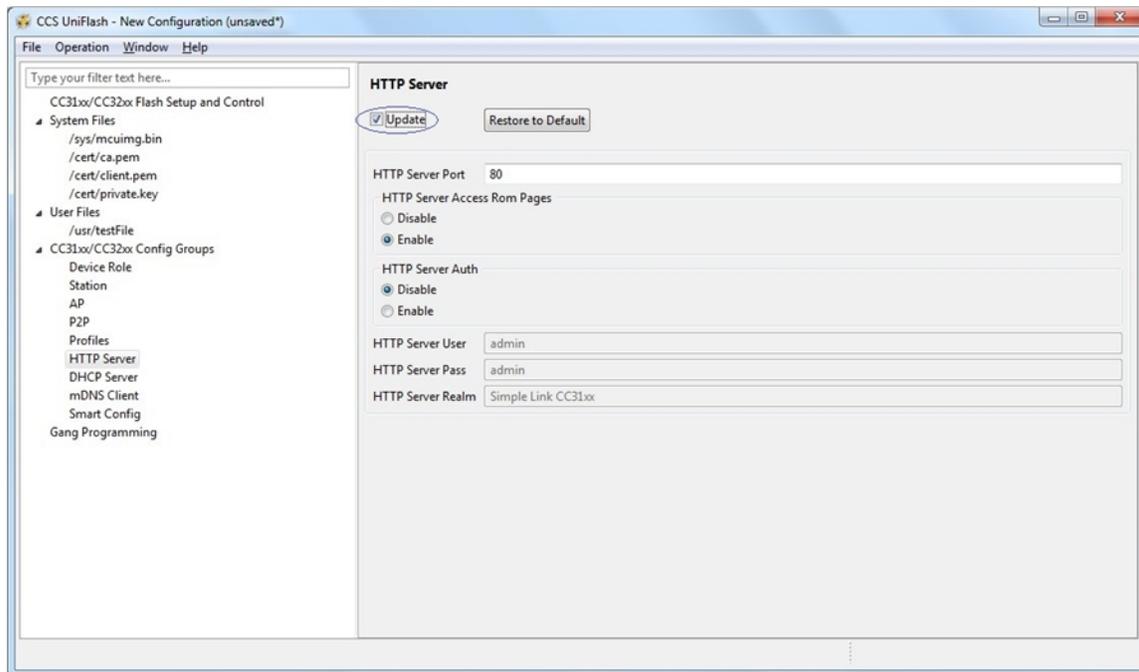


**Figure 22. Http Server Flow**

### 10.3.5.5 Configuration

Figure 23 shows a screenshot of all HTTP Server parameters. In this screen, the following parameters are defined:

- Http Server Port is set to default 80
- Http Server Access Rom Pages is set to default Enable
- Http Server Auth is set to default Disable
- Http Server User, Http Server Pass and Http Server Realm are not relevant since Http Server Auth is set to Disable.



**Figure 23. Http Server Configuration**

### 10.3.6 DHCP Server

#### 10.3.6.1 Restore to Default

At any phase during configuration, it is possible to revert and go back to default configuration. It is applied by clicking the Restore to Default button.

#### 10.3.6.2 Programming

Programming of the device is applied upon clicking the Program button. Another way is from the top bar by clicking Operation → Program. Upon pressing the Program button, Uniflash utility checks for Update checkbox. If Update is checked, programming to target device is executed.

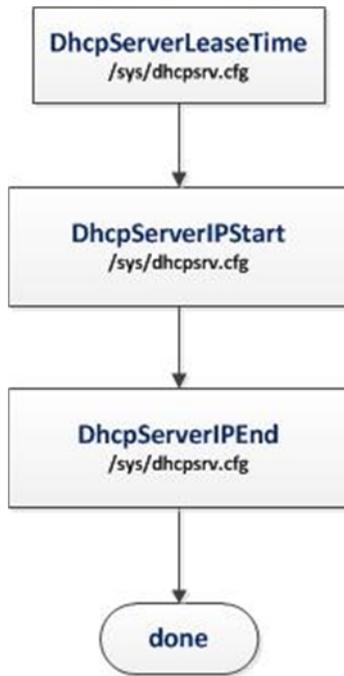
#### 10.3.6.3 Generation of Binary Templates

Unlike programming that generates the binary templates and programs those to the target device, it is possible just to generate those binary templates offline. This option can be used for debug purposes but also to make an offline session that would be used as part of Image Programming. The latest is crucial since all configuration files need to pass integrity test before introduced into an image (since the image cannot apply any integrity test).

The configuration binaries can be generated via the top bar by clicking Operation → Generate Config Binaries.

**10.3.6.4 Flow**

Figure 24 illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.



**Figure 24. Dhcp Server Flow**

### 10.3.6.5 Configuration

Figure 25 shows a screenshot of all DHCP Server parameters. In this screen, the following parameters are defined:

- Dhcp Server Lease Time is set to default 86400
- Dhcp Server IP Start is set to default 192.168.1.2
- Dhcp Server IP End is set to default 192.168.1.254

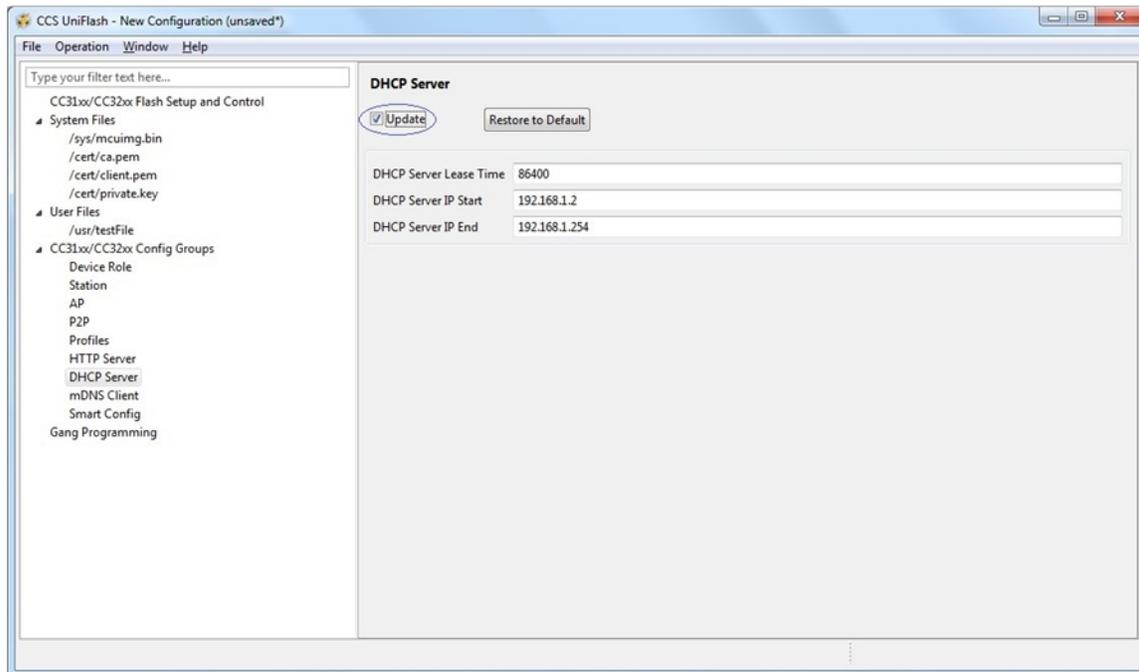


Figure 25. Dhcp Server Configuration

### 10.3.7 mDNS Client

#### 10.3.7.1 Restore to Default

At any phase during configuration, it is possible to revert and go back to default configuration. It is applied by clicking the Restore to Default button.

#### 10.3.7.2 Programming

Programming of the device is applied upon clicking the Program button. Another way is from the top bar by clicking Operation → Program. Upon pressing the Program button, Uniflash utility checks for Update checkbox. If Update is checked, programming to target device is executed.

#### 10.3.7.3 Generation of Binary Templates

Unlike programming that generates the binary templates and programs those to the target device, it is possible just to generate those binary templates offline. This option can be used for debug purposes but also to make an offline session that would be used as part of Image Programming. The latest is crucial since all configuration files need to pass integrity test before introduced into an image (since the image cannot apply any integrity test).

The configuration binaries can be generated via the top bar by clicking Operation->Generate Config Binaries.

### 10.3.7.4 Adding/Deleting a Service

Adding a service is done by checking the Enable mDNS Client x (where x is an index 1-5) checkbox. Deleting a service is done by unchecking the Enable mDNS Client x (where x is an index 1-5) checkbox. For both operations, Programming procedure needs to follow.

### 10.3.7.5 Flow

Figure 26 illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.

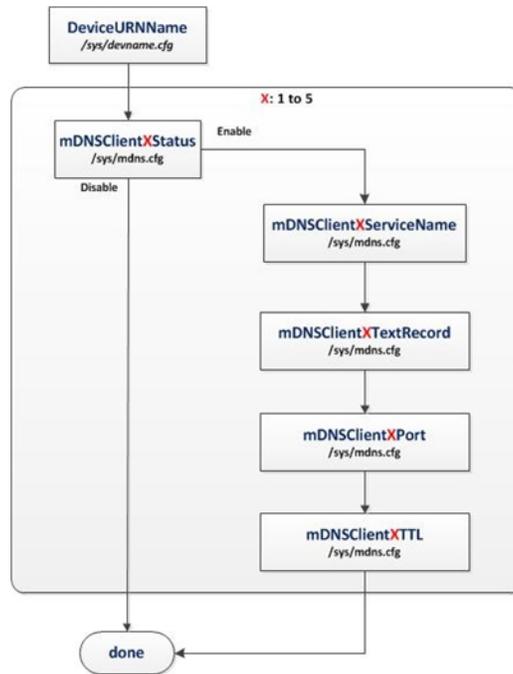


Figure 26. Mdns Flow

### 10.3.7.6 Configuration

Figure 27 shows a screenshot of all the mDNS Client parameters. Some parameters have default values in case the user decides to skip configuration. However, mDNS services are mostly user specific and require the user configuration. There are two relevant screens: the main screen and per service screen. In the main screen, the following parameters are defined:

- Enable mDNS Client x (where x is an index 1-5). In this example, two mDNS services are enabled
- Device URN Name is set to default mysimplelink

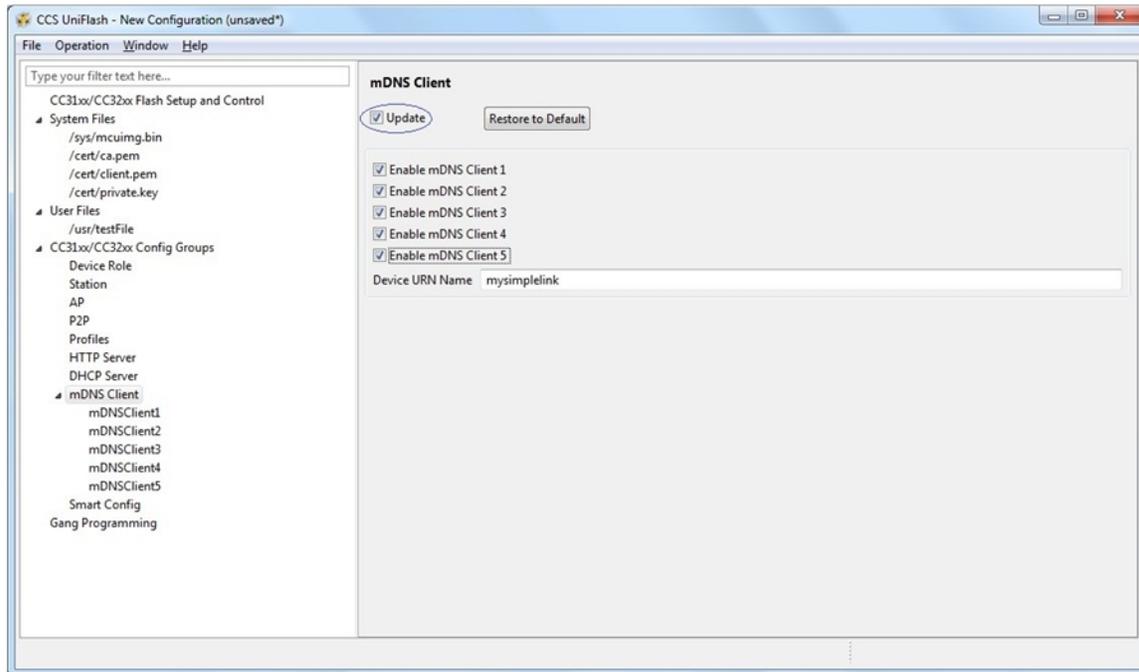
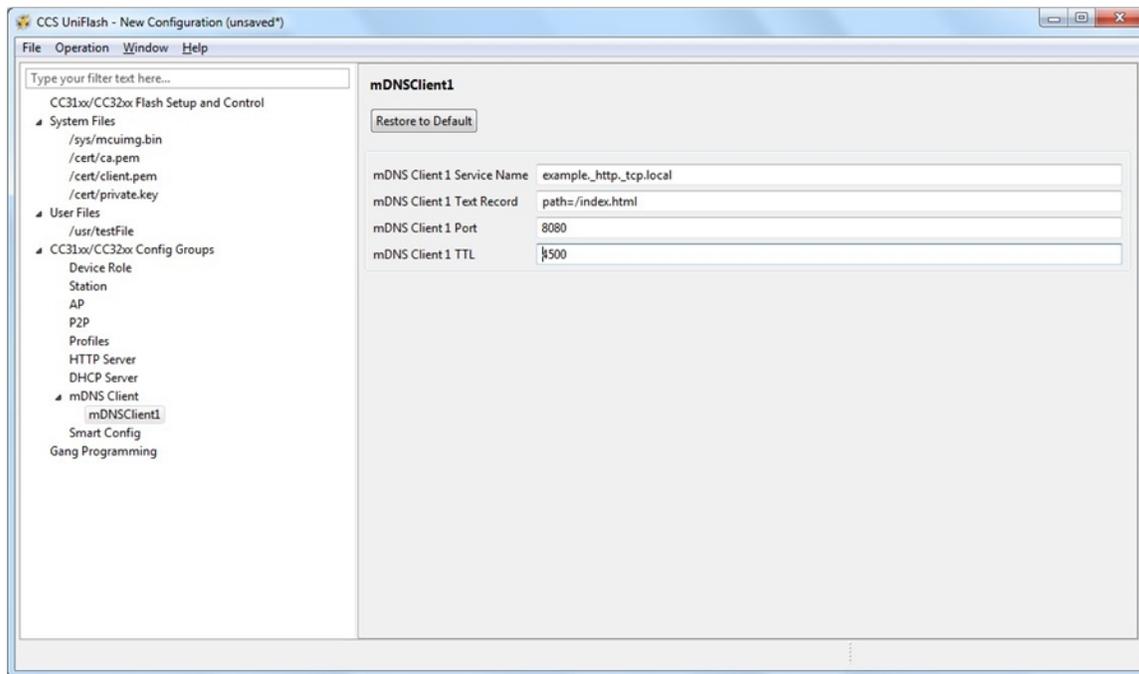


Figure 27. Mdns Main Screen

In the service screen, the following parameters are defined for service #1:

- mDNS Client 1 Service Name. In this example, the service name is set to example.\_http.\_tcp.local.
- mDNS Client 1 Text Record. In this example, the text record is set to path=/index.html.
- mDNS Client 1 Port. In this example, the port is set to 8080.
- mDNS Client 1 TTL. In this example, TTL is set to 4500.



**Figure 28. Mdns Configuration**

## 10.3.8 Profiles

### 10.3.8.1 Restore to Default

At any phase during configuration, it is possible to revert and go back to default configuration. It is applied by clicking the Restore to Default button.

### 10.3.8.2 Programming

Programming of the device is applied upon clicking the Program button. Another way is from the top bar by clicking Operation → Program. Upon pressing the Program button, Uniflash utility checks for Update checkbox. If Update is checked, programming to target device is executed.

### 10.3.8.3 Generation of Binary Templates

Unlike programming that generates the binary templates and programs those to the target device, it is possible just to generate those binary templates offline. This option can be used for debug purposes but also to make an offline session that would be used as part of Image Programming. The latest is crucial since all configuration files need to pass integrity test before introduced into an image (since the image cannot apply any integrity test).

The configuration binaries can be generated via the top bar by clicking Operation → Generate Config Binaries.

### 10.3.8.4 Adding/Deleting a Profile

Adding a profile is done by checking the Enable Profile x (where x is an index 1-7) checkbox. Deleting a profile is done by unchecking the Enable Profile x (where x is an index 1-7) checkbox. For both operations, Programming procedure needs to follow.

### 10.3.8.5 Flow

Figure 29 illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.

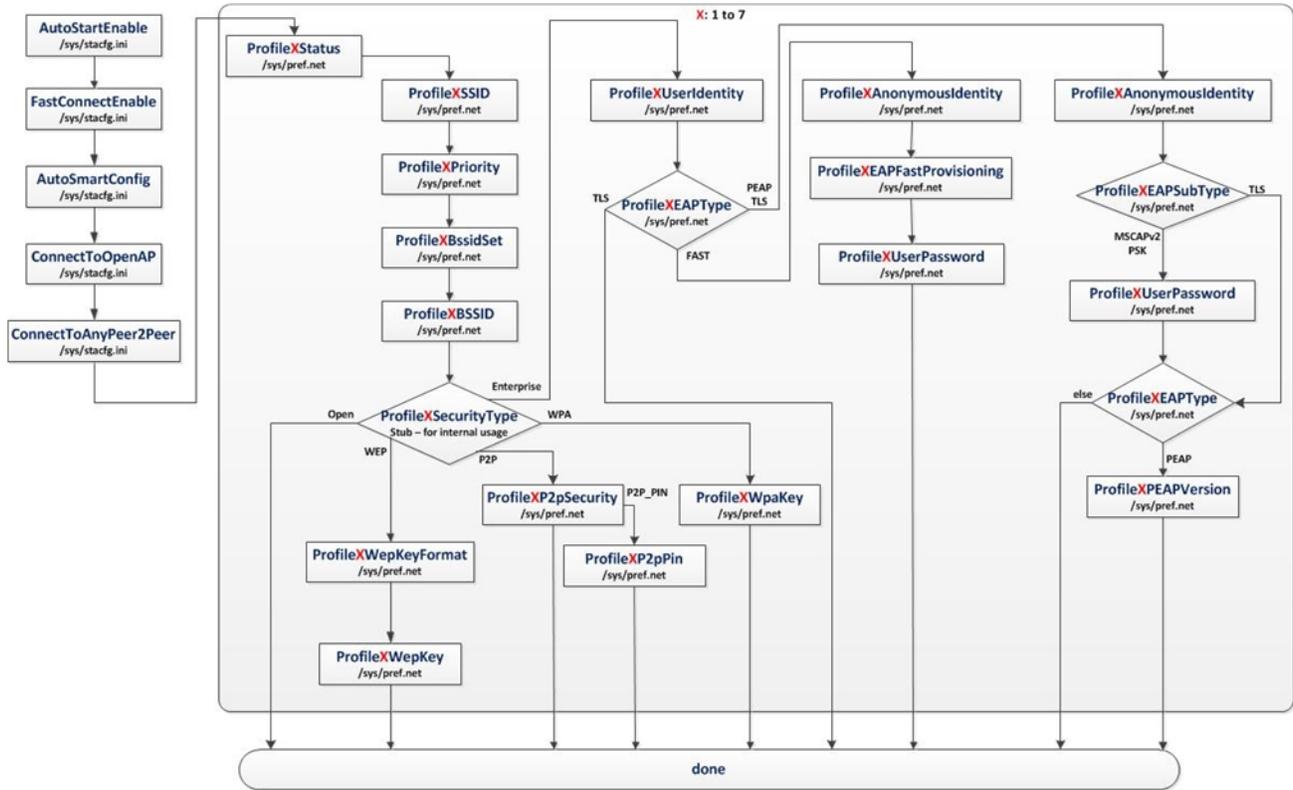
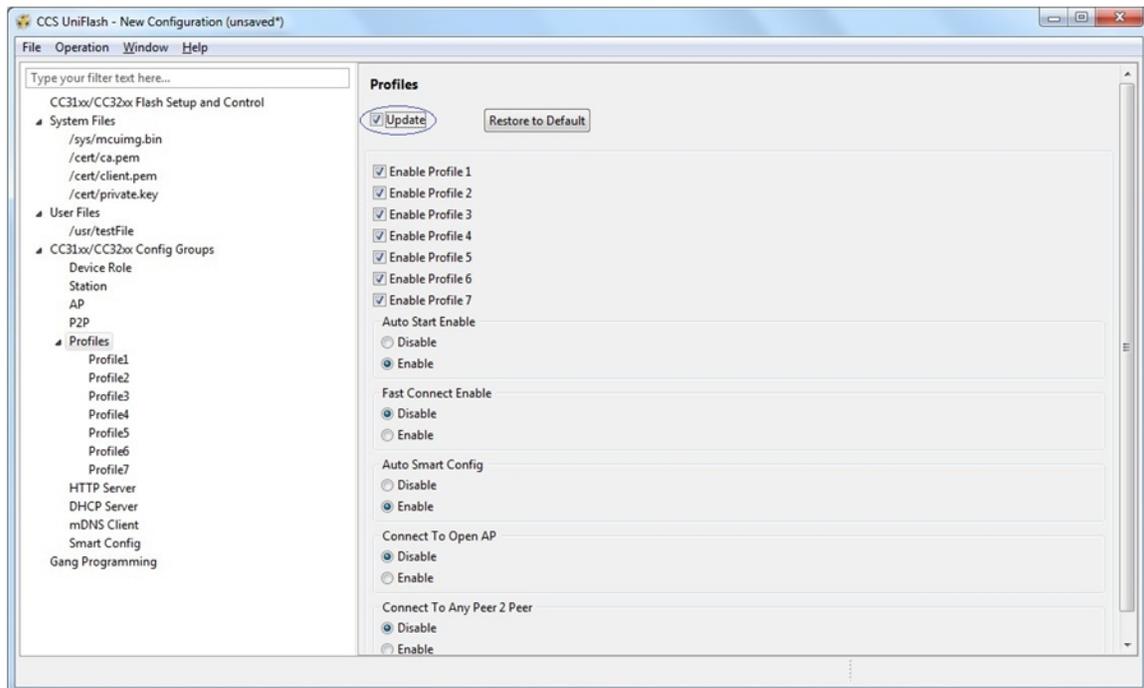


Figure 29. Profiles Flow

### 10.3.8.6 Configuration

Figure 30 shows a screenshot of all the Profiles parameters. Some parameters have default values in case the user decides to skip configuration. However, profiles are mostly user specific and thus require user configuration. There are two screens relevant, the main screen and per profile screen. In the main screen, the following parameters are defined:

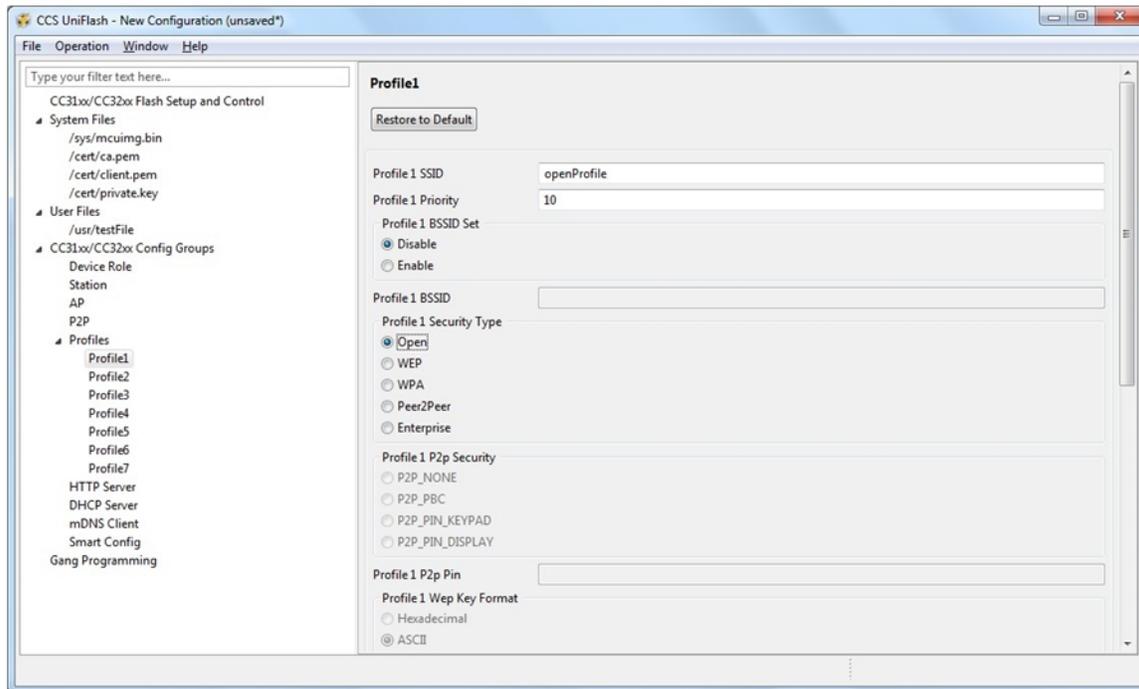
- Enable Profile x (where x is an index 1-7). In this example, two profiles are enabled.
- Auto Start Enable is set to default Enable.
- Fast Connect Enable is set to default Disable.
- Auto Smart Config is set to default Enable.
- Auto To Open AP is set to default Disable.
- Connect To Any Peer 2 Peer is set to default Disable.



**Figure 30. Profiles Main Screen**

In the per profile screen, several example profiles are described. For Open security profile, the following parameters are defined:

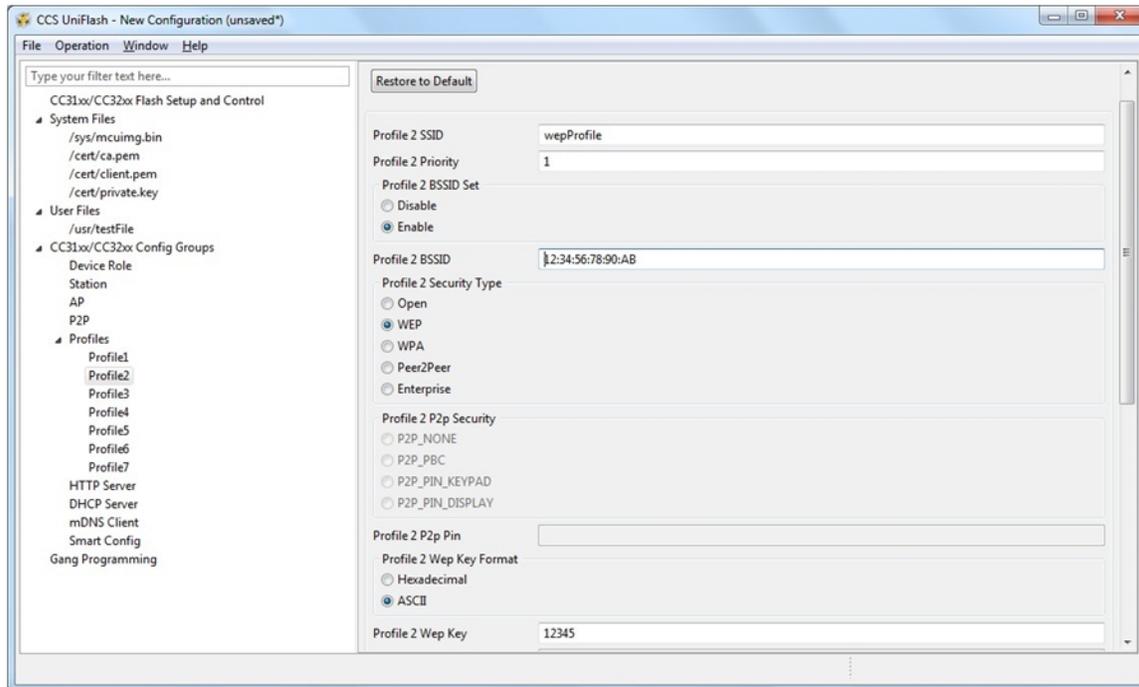
- Profile 1 SSID is set to openProfile.
- Profile 1 Priority is set to 10.
- Profile 1 Bssid Set is set to Disable.
- Profile 1 Security Type is set to Open.
- All rest of parameters is not relevant for an Open security profile.



**Figure 31. Open Profile Screen**

For WEP security profile, the following parameters are defined:

- Profile 2 SSID is set to wepProfile.
- Profile 2 Priority is set to 1.
- Profile 2 Bssid Set is set to Enable.
- Profile 2 BSSID is set to 12:34:56:78:90:AB.
- Profile 2 Security Type is set to WEP.
- WEP Key Format is set to ASCII.
- Profile 2 Wep Key is set to 12345.
- All rest of parameters is not relevant for a WEP security profile.



**Figure 32. Wep Profile Screen**

For WPA security profile, the following parameters are defined:

- Profile 3 SSID is set to wpaProfile.
- Profile 3 Priority is set to 5.
- Profile 3 Bssid Set is set to Disable.
- Profile 3 Security Type is set to WPA.
- Profile 3 Wpa Key is set to 1234567890.
- All rest of parameters is not relevant for a WPA security profile.

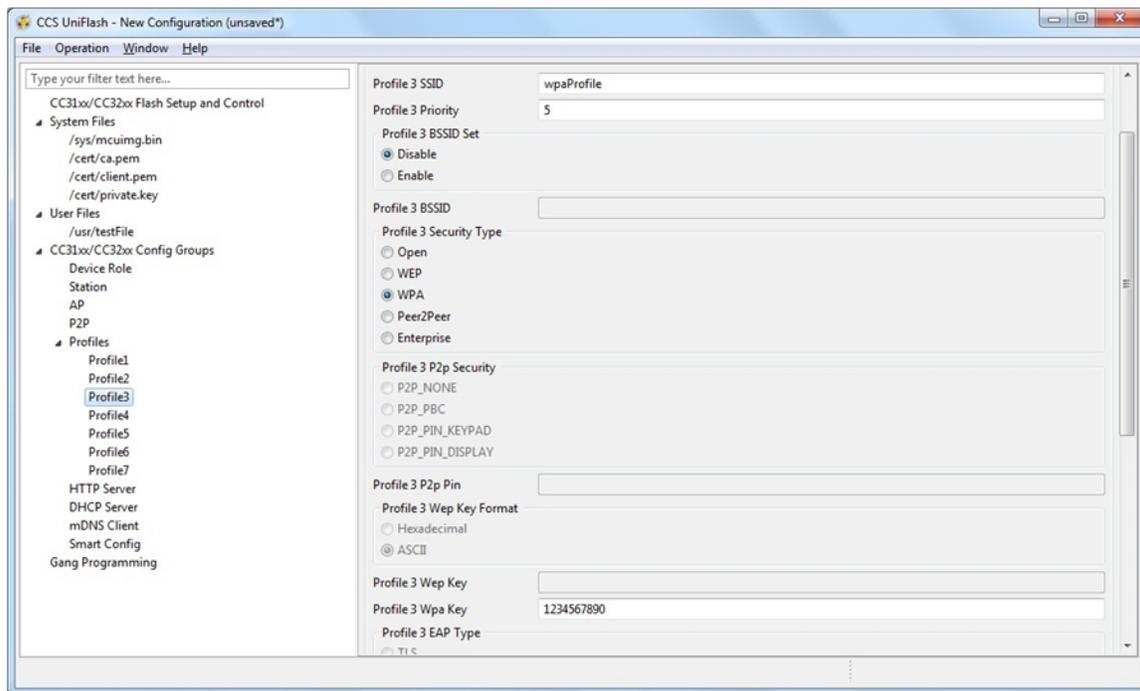
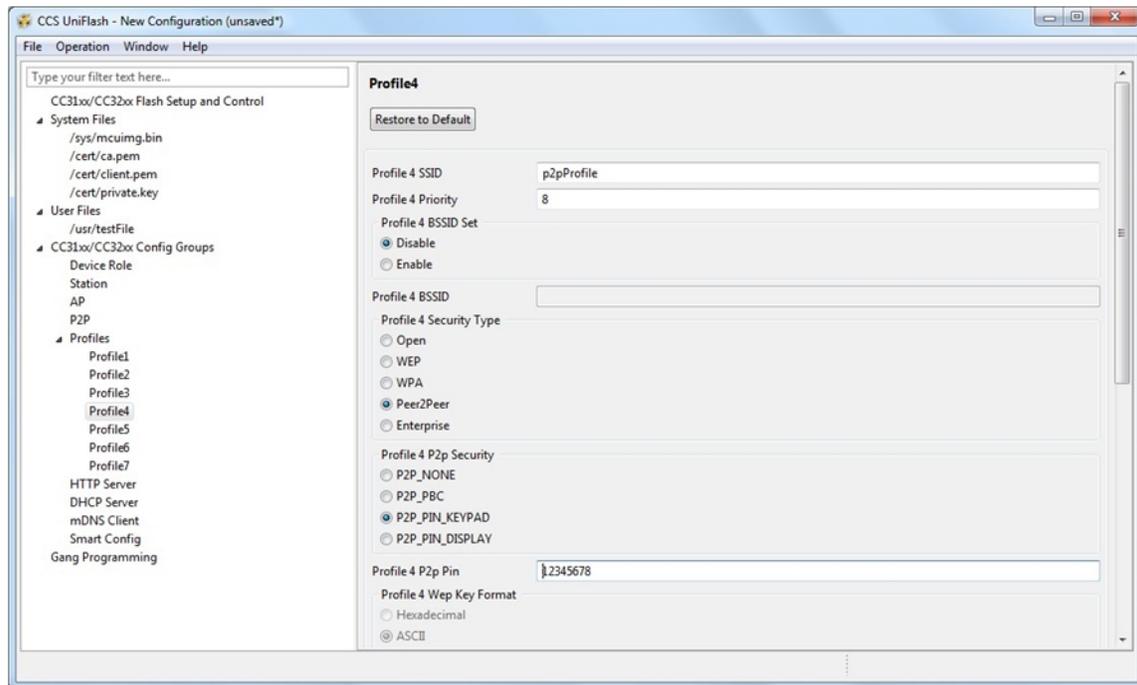


Figure 33. Wpa Profile Screen

For Peer 2 Peer security profile, the following parameters are defined:

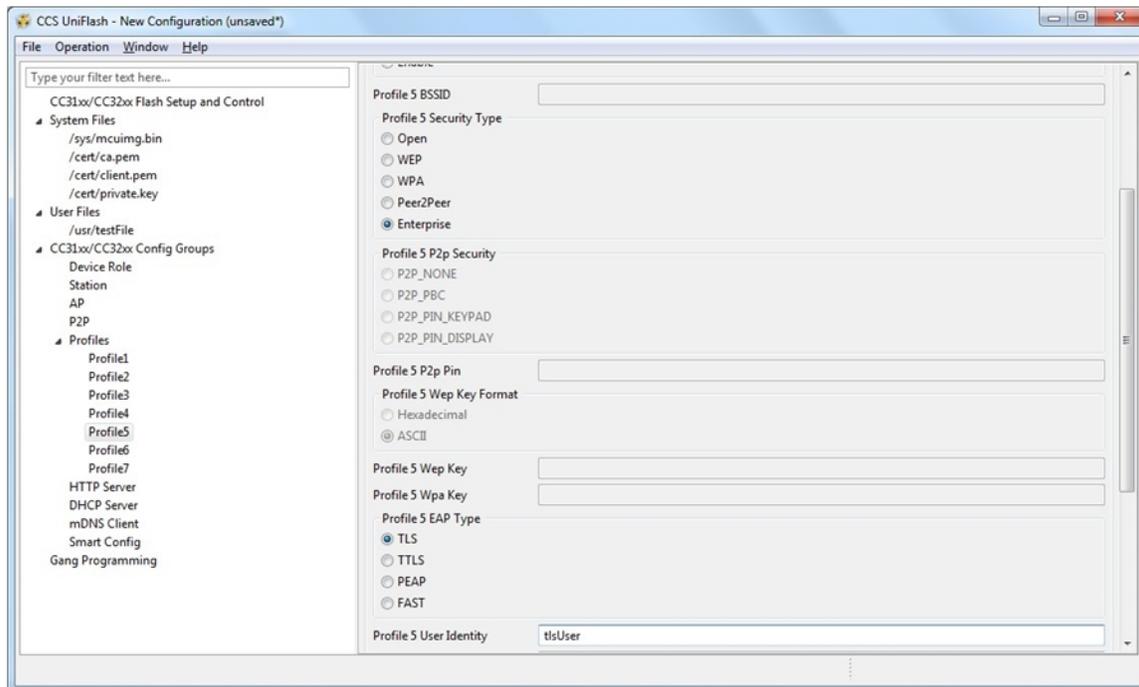
- Profile 4 SSID is set to p2pProfile.
- Profile 4 Priority is set to 8.
- Profile 4 Bssid Set is set to Disable.
- Profile 4 Security Type is set to Peer2Peer.
- Profile 4 P2p Security is set to P2P\_PIN\_KEYPAD.
- Profile 4 P2P Pin is set to 12345678.
- All rest of parameters is not relevant for a Peer 2 Peer security profile.



**Figure 34. P2p Profile Screen**

For Enterprise security with TLS profile, the following parameters are defined:

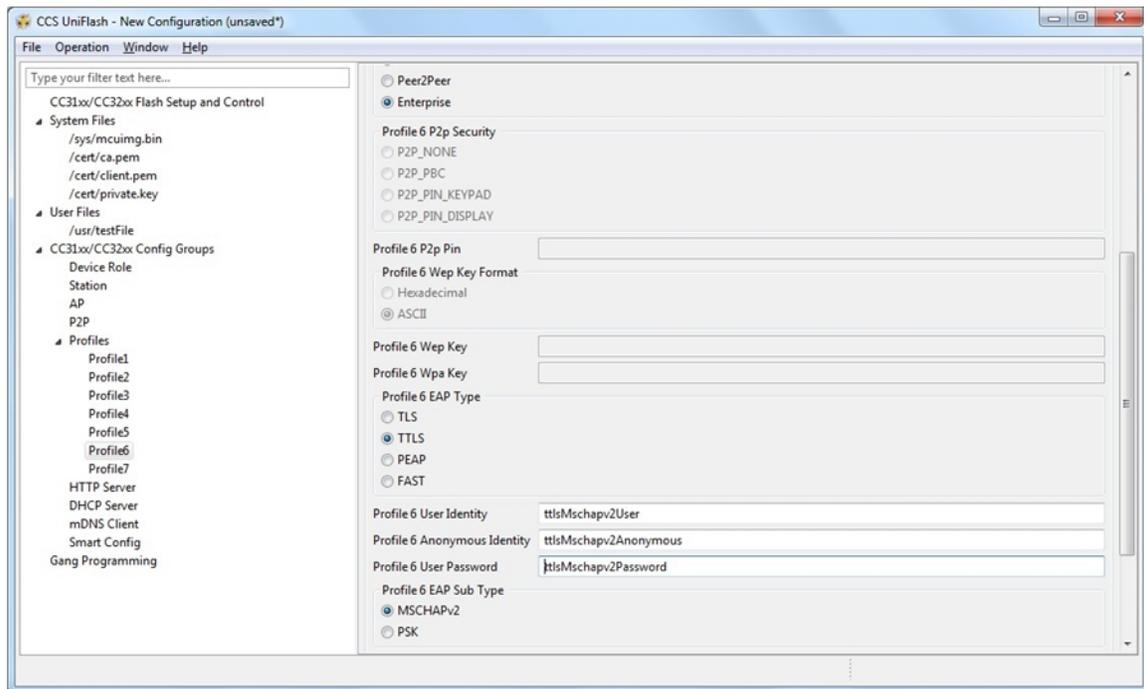
- Profile 5 SSID is set to tlsProfile.
- Profile 5 Priority is set to 1.
- Profile 5 Bssid Set is set to Disable.
- Profile 5 Security Type is set to Enterprise.
- Profile 5 EAP Type is set to TLS.
- Profile 5 User Identity is set to tlsUser.
- All rest of parameters is not relevant for a TLS Enterprise security profile.



**Figure 35. Tls Profile Screen**

For Enterprise security with TTLS/Mschapv2 profile, the following parameters are defined:

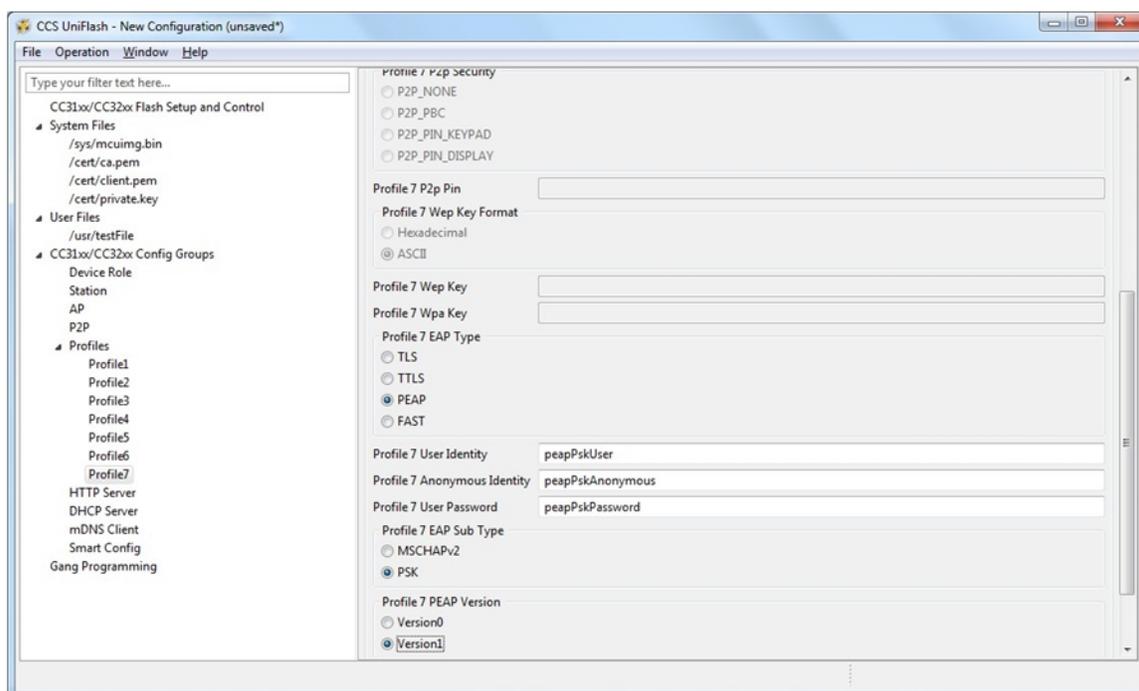
- Profile 6 SSID is set to `ttlsMschapv2Profile`.
- Profile 6 Priority is set to 7.
- Profile 6 Bssid Set is set to Disable.
- Profile 6 Security Type is set to Enterprise.
- Profile 6 EAP Type is set to TTLS.
- Profile 6 User Identity is set to `ttlsMschapv2User`.
- Profile 6 Anonymous Identity is set to `ttlsMschapv2Anonymous`.
- Profile 6 User password is set to `ttlsMschapv2Password`.
- Profile 6 EAP Sub Type is set to MSCHAPv2.
- All of the rest of the parameters are not relevant for a TTLS/Mschapv2 Enterprise security profile.



**Figure 36. TtIsMschapv2 Profile Screen**

For Enterprise security with PEAP/PSK profile, the following parameters are defined:

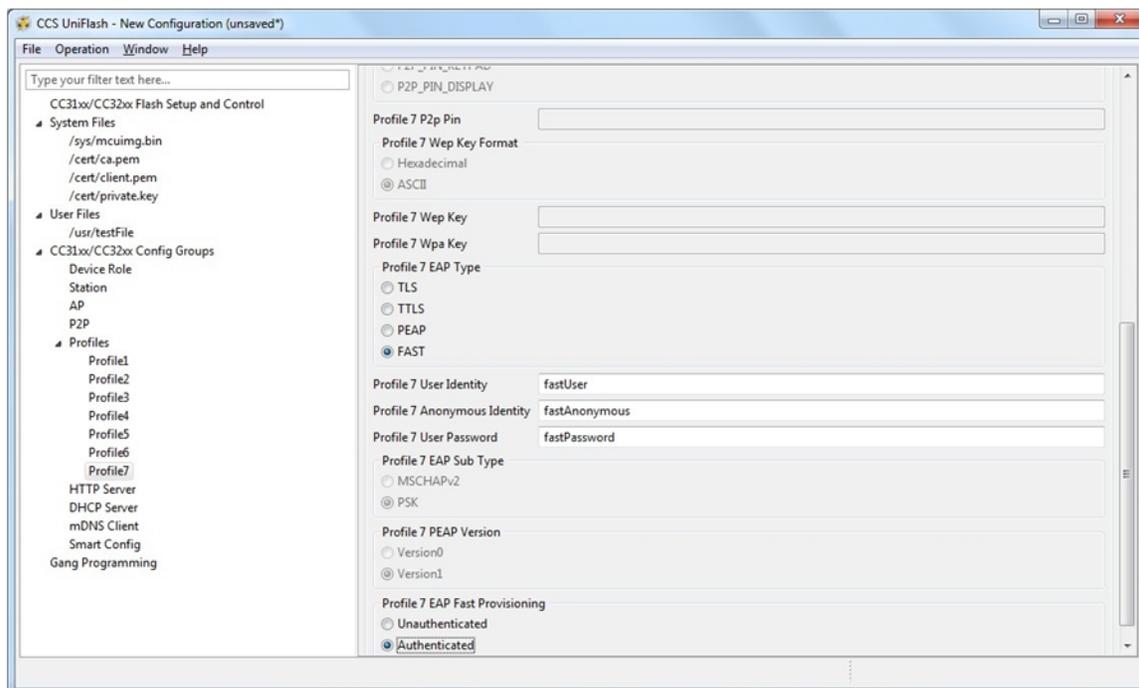
- Profile 7 SSID is set to peapPskProfile.
- Profile 7 Priority is set to 2.
- Profile 7 Bssid Set is set to Disable.
- Profile 7 Security Type is set to Enterprise.
- Profile 7 EAP Type is set to PEAP.
- Profile 7 User Identity is set to peapPskUser.
- Profile 7 Anonymous Identity is set to peapPskAnonymous.
- Profile 7 User password is set to peapPskPassword.
- Profile 7 EAP Sub Type is set to PSK.
- Profile 7 PEAP Version is set to Version1.
- All of the rest of the parameters are not relevant for a PEAP/PSK Enterprise security profile.



**Figure 37. PeapPsk Profile Screen**

For Enterprise security with EAP\_FAST profile, the following parameters are defined:

- Profile 7 SSID is set to fastProfile.
- Profile 7 Priority is set to 3.
- Profile 7 Bssid Set is set to Disable.
- Profile 7 Security Type is set to Enterprise.
- Profile 7 EAP Type is set to FAST.
- Profile 7 User Identity is set to fastUser.
- Profile 7 Anonymous Identity is set to fastAnonymous.
- Profile 7 User password is set to fastPassword.
- Profile 7 EAP Fast Provisioning is set to Authenticated.
- All of the rest of the parameters are not relevant for an EAP\_FAST Enterprise security profile.



**Figure 38. Fast Profile Screen**

## 10.3.9 Smart Config

### 10.3.9.1 Restore to Default

At any phase during configuration, it is possible to revert and go back to default configuration. It is applied by clicking the Restore to Default button.

### 10.3.9.2 Programming

Programming of the device is applied upon clicking the Program button. Another way is from the top bar by clicking Operation → Program. Upon pressing the Program button, Uniflash utility checks for Update checkbox. If Update is checked, programming to target device is executed.

### 10.3.9.3 Generation of Binary Templates

Unlike programming, that generates the binary templates and programs those to the target device, it is possible to just generate those binary templates offline. This option can be used for debug purposes but also to make an offline session that would be used as part of Image Programming. The latest is crucial since all configuration files need to pass integrity test before introduced into an image (since the image cannot apply any integrity test).

The configuration binaries can be generated via the top bar by clicking Operation → Generate Config Binaries.

### 10.3.9.4 Flow

Figure 39 illustrates the parameters' flow during configuration. It is highly advised to follow this flow but it is not mandatory. The user may choose to go through configuration steps differently.

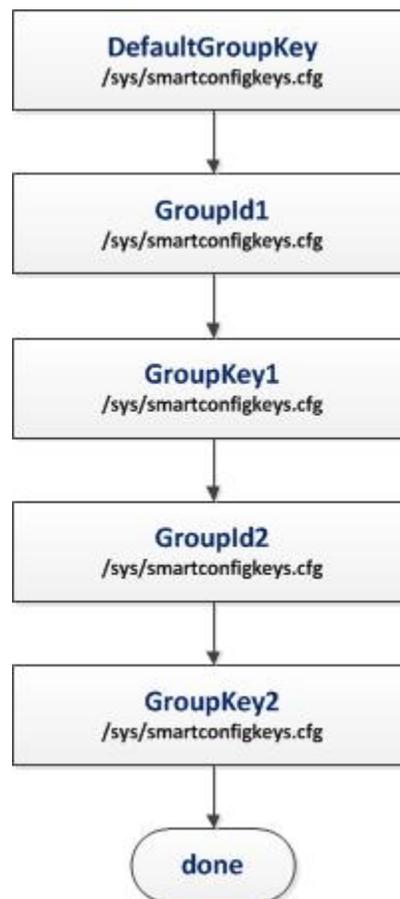


Figure 39. Smartconfig Flow

### 10.3.9.5 Configuration

Figure 40 shows a screenshot of all the Smart Config parameters. Some parameters have default values in case the user decides to skip configuration. However, Smart Config parameters are mostly user specific and thus require user configuration. In this screen, the following parameters are defined:

- Default Group Key is by default not set.
- Group Id 1 is by default not set.
- Group Key 1 is by default not set.
- Group Id 2 is by default not set.
- Group Key 2 is by default not set.

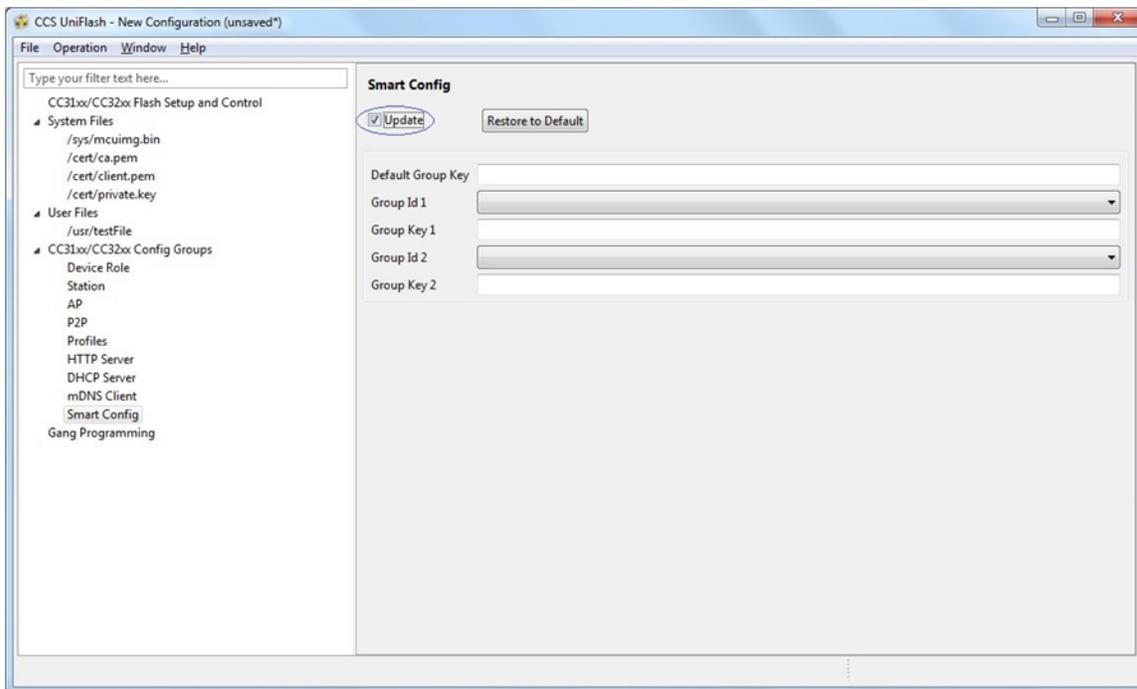


Figure 40. Smartconfig Configuration

## 11 Image Creation and Programming

### 11.1 Overview

Image creation procedure allows building a complete operating image, ensuring that its integrity is not compromised. This feature may be used during development but is more relevant to production line purposes. The image can either get programmed by using the Uniflash (and thus via the UART lines) or by using any off-the-shelf SPI programmers directly via the serial flash SPI lines.

### 11.2 Basic Concept and Operation

The image stored on the serial flash is extracted by the SimpleLink device upon first power up and file system is then created.

The device itself is responsible to self-extract the image and build the file system. This conversion process performed during the first power-up and before the Host receives the initialization complete event. On successful completion of the conversion, the SimpleLink device continues with the regular initialization process.

As mentioned earlier, Image creation is mainly used during production line. The image can either get programmed by an external flash programmer when the serial flash is not yet assembled on the board (not attached to CC3xxx device) or when the serial flash is already assembled on the board.

The Production Line procedure is as follows:

1. Prepare the image.
2. If the serial flash is not assembled on the target board:
  - a. Program the serial flash devices with the image using any off-the-shelf SPI programmer before assembling them on the target boards.
  - b. Assemble the programmed serial flash devices on the target boards.
3. If the serial flash is already assembled on the target board:
  - a. Program the devices with the image using Uniflash (via UART lines).
4. Power-on the board as part of its final tests.
5. Wait for indication from the Host MCU for successful power-up.
6. Continue with the rest of the final tests.

The Host MCU is responsible to generate the indication of successful power-up only after the SimpleLink device sends the initialization complete event.

As mentioned, if the conversion process fails after the power-up from any reason, the device will not complete the power-up and the Host MCU will not get the initialization complete event. To exit this state, the user has to replace the serial flash or fix the image and retry.

### 11.3 Image Creation – Supported Commands

#### 11.3.1 Overview

This section lists all possible commands that can be contained in an image. Each command that is added to the image appears on the right side Gang Programming window in Uniflash. Double clicking on the added command would list all its configured parameters.

#### 11.3.2 General Settings

This is not a command but a general settings.

<b>Attribute</b>	Storage Capacity Bytes
<b>Description</b>	Capacity of serial flash
<b>Options</b>	Drop Down selection. Possible values are: 512KB, 1MB, 2MB, 4MB, 8MB, 16MB
<b>Default</b>	1MB
<b>Dependencies</b>	None

#### 11.3.3 Format Storage Command

This command formats and optionally erases the serial flash to allow creating the file system. By default it erases only the necessary locations but optionally it allow erasing all the serial flash except the gang image location.

---

**NOTE:** Depending on the serial flash used, erasing blocks takes relatively long time. If the serial flash is already erased, then it is not necessary.

---

<b>Attribute</b>	Erase Storage
<b>Description</b>	Boolean value indicating if the blocks on the target serial flash should be erased prior the writing. Erasing the target blocks is the safe side but will increase the processing time
<b>Options</b>	Checkbox indicating true or false
<b>Default</b>	Unchecked
<b>Dependencies</b>	None

### 11.3.4 Write Service Pack Command

This command writes the service pack of the networking subsystem.

<b>Attribute</b>	File Location
<b>Description</b>	Full path indicating the location of the ServicePack file on the local machine
<b>Options</b>	Browse button
<b>Default</b>	N/A
<b>Dependencies</b>	None

<b>Attribute</b>	Service Pack Version
<b>Description</b>	Type of device the Service Pack is used for
<b>Options</b>	UCF_FLASH (pre-production devices), UCF_ROM (production devices)
<b>Default</b>	UCF_ROM
<b>Dependencies</b>	None

### 11.3.5 Write File Command

The command creates and writes a file.

<b>Attribute</b>	File System Name
<b>Description</b>	String indicating the file name on the target device
<b>Options</b>	Text box
<b>Default</b>	N/A
<b>Dependencies</b>	None

<b>Attribute</b>	File Location
<b>Description</b>	Full path indicating the location of the file on the local machine
<b>Options</b>	Browse button
<b>Default</b>	N/A
<b>Dependencies</b>	None

<b>Attribute</b>	File Open Flags List
<b>Description</b>	File flags
<b>Options</b>	Multiple selection checkboxes. Only FILE_OPEN_FLAG_FAILSAFE flag is configurable. This is identical to the rollback flag
<b>Default</b>	Unchecked
<b>Dependencies</b>	None

<b>Attribute</b>	File Open Flags List
<b>Description</b>	Maximum size of the file. 0 indicates the original file size
<b>Options</b>	Text box
<b>Default</b>	0
<b>Dependencies</b>	None

### 11.3.6 Write Gang Image Command

This command keeps the gang image as a file. Storing the gang image as a file keeps the entire source gang image as a file in the File System of the device and will allow in the future using this image to enable returning to the factory defaults. The return to factory defaults feature is not supported yet.

---

**NOTE:** Since the image may take considerable amount of space and since it is not usable for current production devices, it is highly advised not to include this command in the image.

---

### 11.3.7 Start Logger Command

This command creates a debug log for the conversion process done by the device itself. It is possible to enable the log during final production image. However, in this case it makes sense that the "delete log when no errors" attribute be true otherwise the log would be kept on the file system as regular file.

If the extraction process fails, there is no file system and in order to read the log the user would have to read the entire content of the SFLASH using external tools and search for the first text line of the log: "Start Log" or 617274204C6F67 in hexadecimal.

<b>Attribute</b>	File System Name
<b>Description</b>	String indicating the Log file name on the target device
<b>Options</b>	Text box
<b>Default</b>	Gang.log
<b>Dependencies</b>	None

<b>Attribute</b>	Max File Size
<b>Description</b>	Number indicating the maximum size for the Log file
<b>Options</b>	Text box, capacity in bytes
<b>Default</b>	3000
<b>Dependencies</b>	None

<b>Attribute</b>	Enable Log
<b>Description</b>	Enable/Disable the logging mechanism during image extraction
<b>Options</b>	Checkbox indicating true or false
<b>Default</b>	Unchecked
<b>Dependencies</b>	None

<b>Attribute</b>	Delete Log When No errors
<b>Description</b>	Boolean attribute indicates whether to delete the file in case of successful extraction
<b>Options</b>	Checkbox indicating true or false
<b>Default</b>	Unchecked
<b>Dependencies</b>	None

## 11.4 Image Creation – GUI Interface

### 11.4.1 Image Programming Summary Window

The Gang Programming summary window lists all the commands that eventually get written into the template XML database file. The commands are then compiled into a binary image either via the Export button or the Program button. The list is ordered (executed in top down order).

The left side column list the command name and the right side column list the target filename (format command does not list anything).

The operations that may be applied on the commands via the Gang Programming window are:

- Double clicking a command. Presents the command configuration
- Multiple selections of several commands. Used mainly for removal of several command in a single click
- Moving a command up or down the list. May be applied by dragging the command in the desired direction
- Tooltip support. Hovering over the commands pops a short description window.

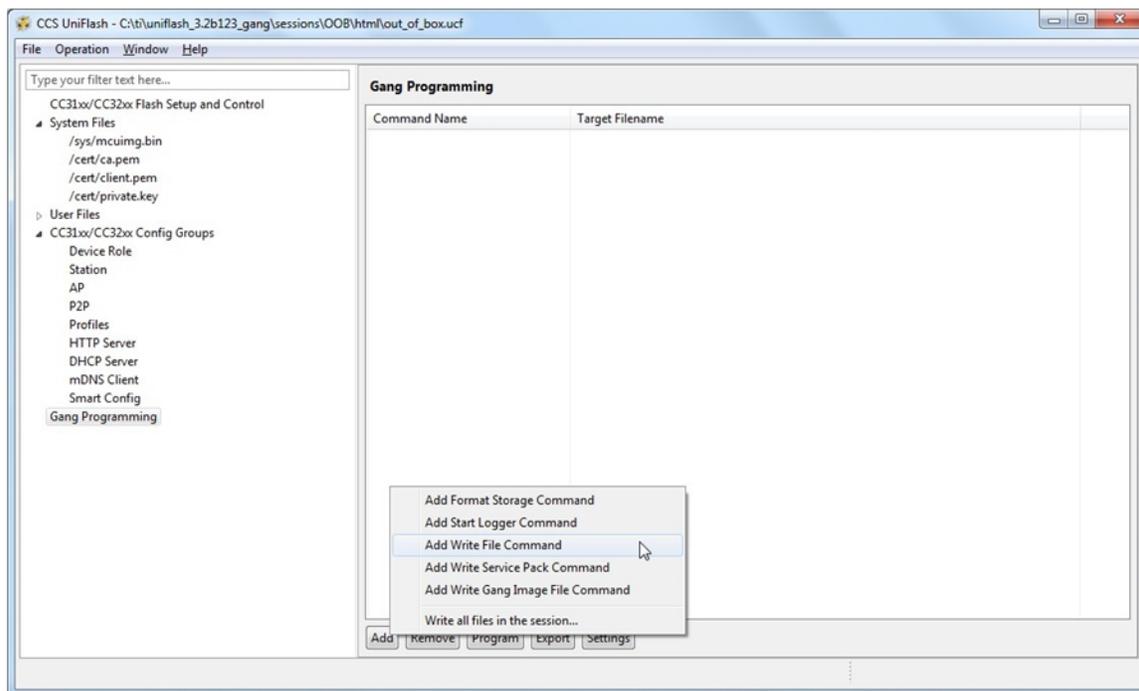
### 11.4.2 Image Programming Buttons

**NOTE:** Tooltip support. Hovering over the buttons pops a short description window.

#### 11.4.2.1 Add

The Add button enables the user to add a command to the image. This command gets extracted and executed upon image extraction.

All possible commands are listed in Image Creation – supported commands paragraph. In addition to manual addition of Writefile commands, it is possible to add multiple WriteFile commands in a single click. It is done by including all required files as part of a session and then import the session into the image by clicking Add → Write all files in session. All options are shown in [Figure 41](#).



**Figure 41. Add Command**

### 11.4.2.2 Remove

The Remove button enables the user to remove any command from the image. Also multiple commands can be removed using multiple selections as shown in Figure 42. Also note that it is possible to remove commands by selecting items and pressing the delete button on the keyboard.

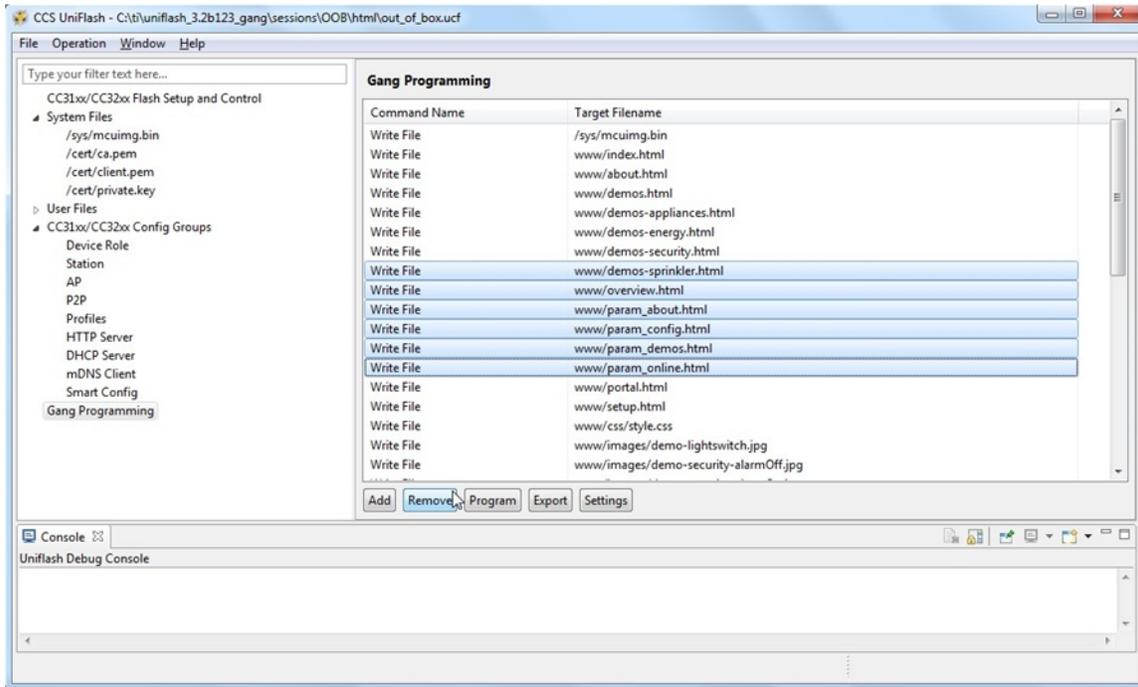
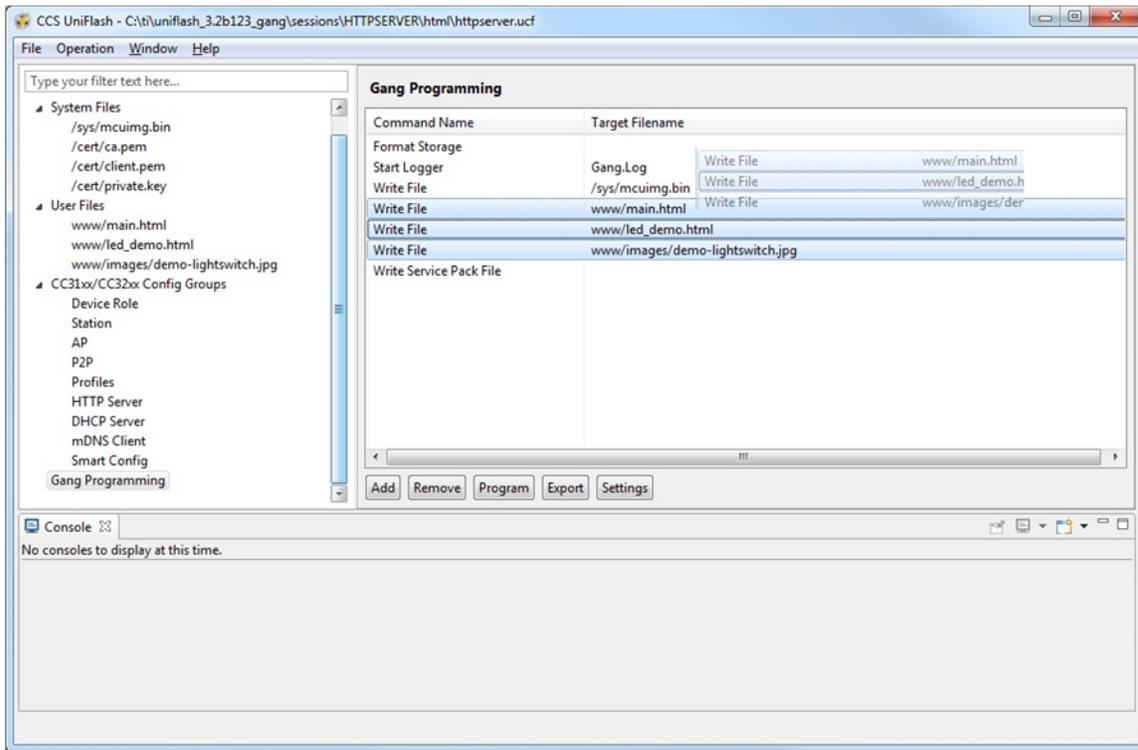


Figure 42. Multiple Remove

### 11.4.2.3 Move

This is not a button, just an option to move command up and down by selecting commands (multiple selection is also possible) and moving those with the computer mouse.



**Figure 43. Move Command**

### 11.4.2.4 Program

The Program button generates the XML template and the binary image. In addition, it programs the binary image to the connected target device.

### 11.4.2.5 Export

The Export button generates the XML template and the binary image. The binary image can then be programmed to the device. Output directory containing the binary image is created on the same location where the XML template is created.

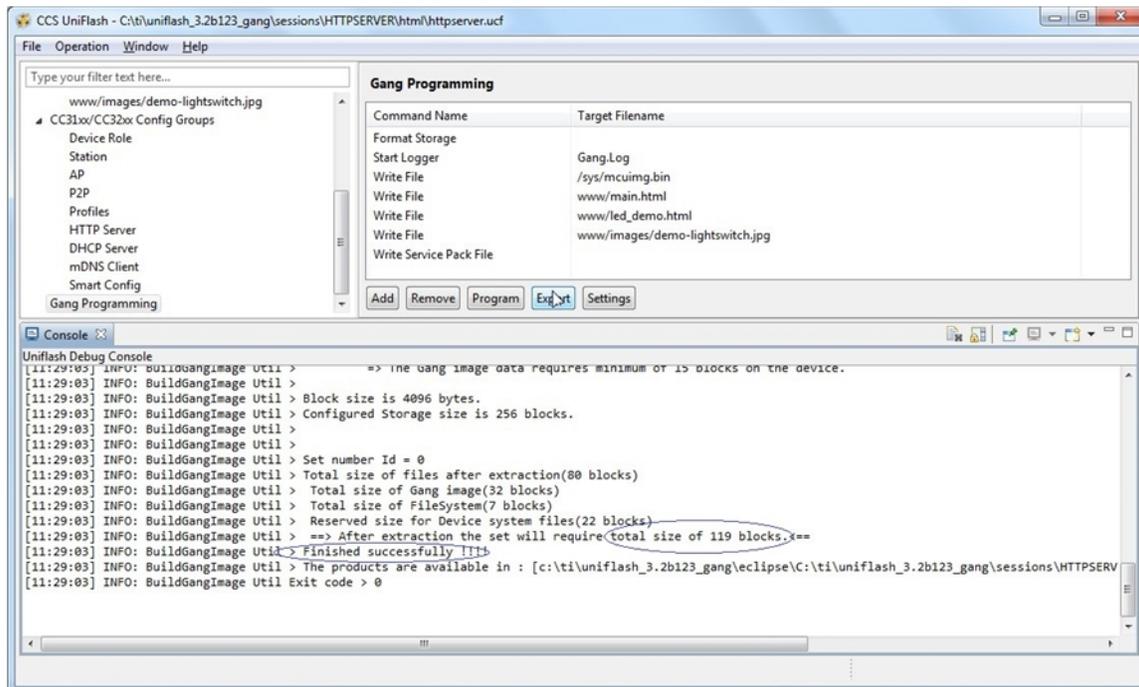
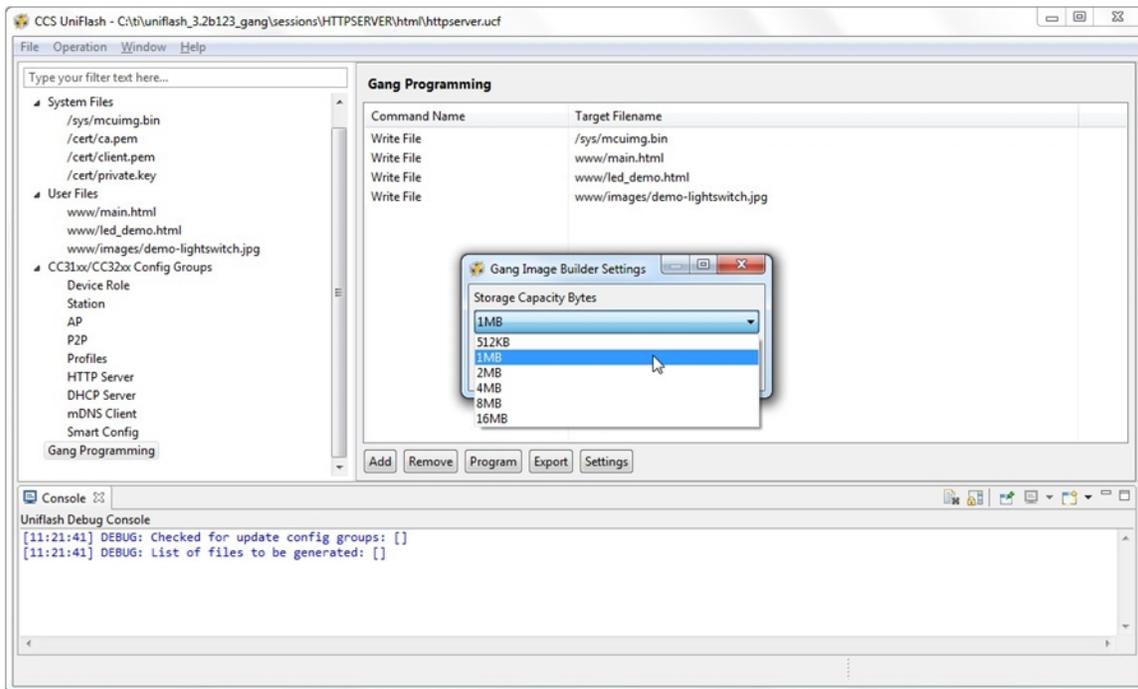


Figure 44. Export Button

### 11.4.2.6 Settings

The Settings button opens a general settings window. Currently, it is required to configure the size of the serial flash.



**Figure 45. General Setting Button**

## 11.5 Image Programming to Target

Image programming of the device is applied upon clicking the 'Image Programmig' button. Another way is from the top bar by clicking Operation → Image Programming. Uniflash then requires the binary image created by Export operation. Please note that the full sized image is required and not the IntelHex format binary.

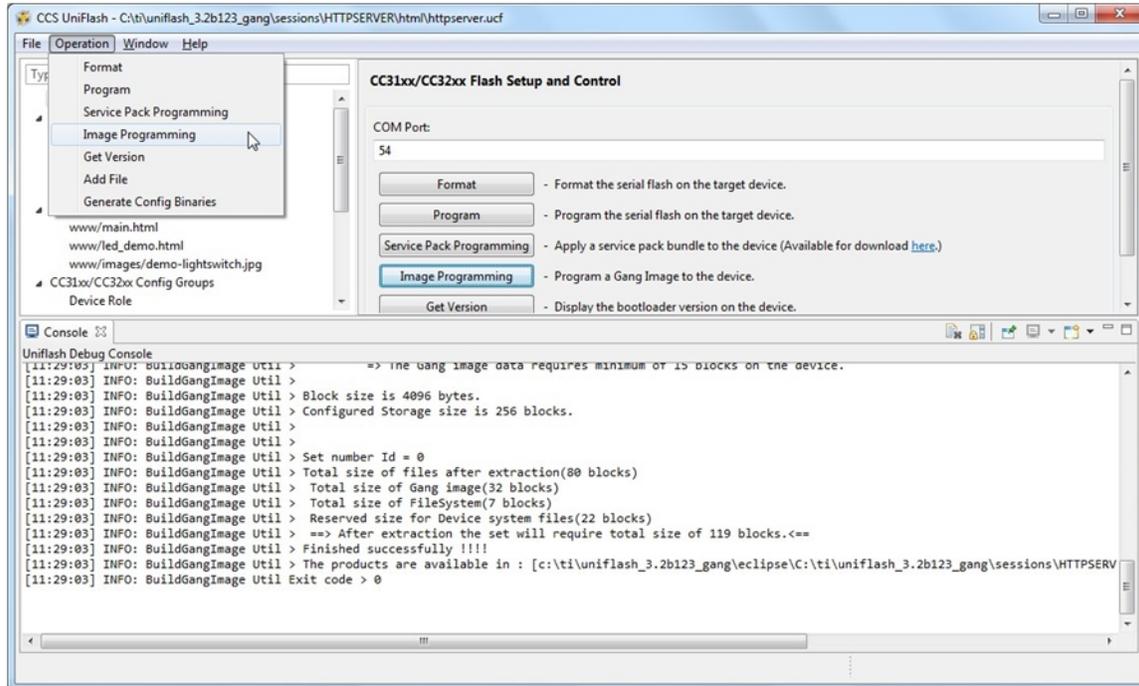


Figure 46. Programming Button

### 11.6 How to Build an Image Guide

This section describes the recommended procedure to build a binary image. Before starting, please verify the following checklist:

- Capacity of the serial flash
- Location of all user files
- Desired setup of all configuration files
- Latest Service Pack binary file

Figure 47 describes the flow that must be followed in order to generate an image.

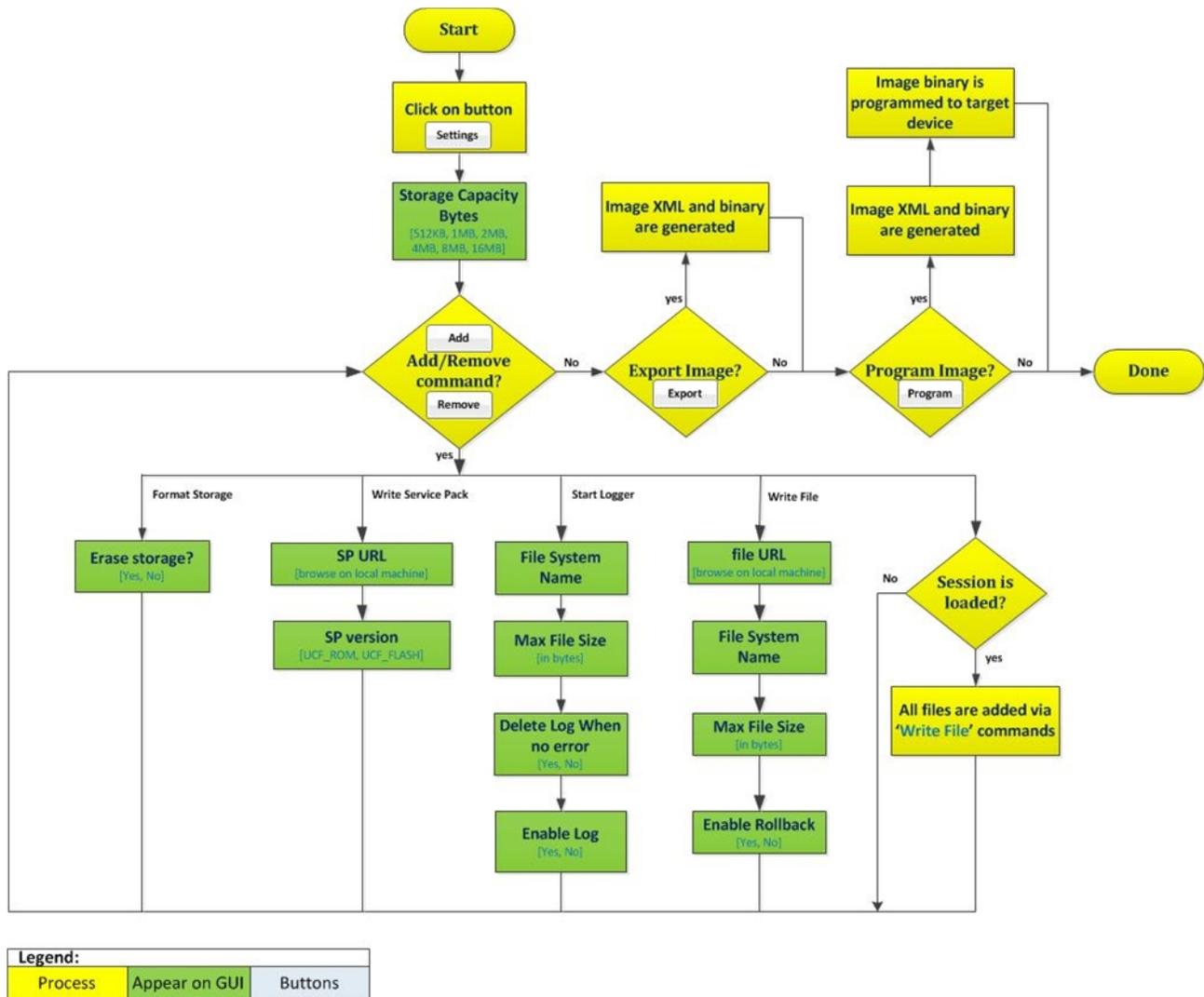
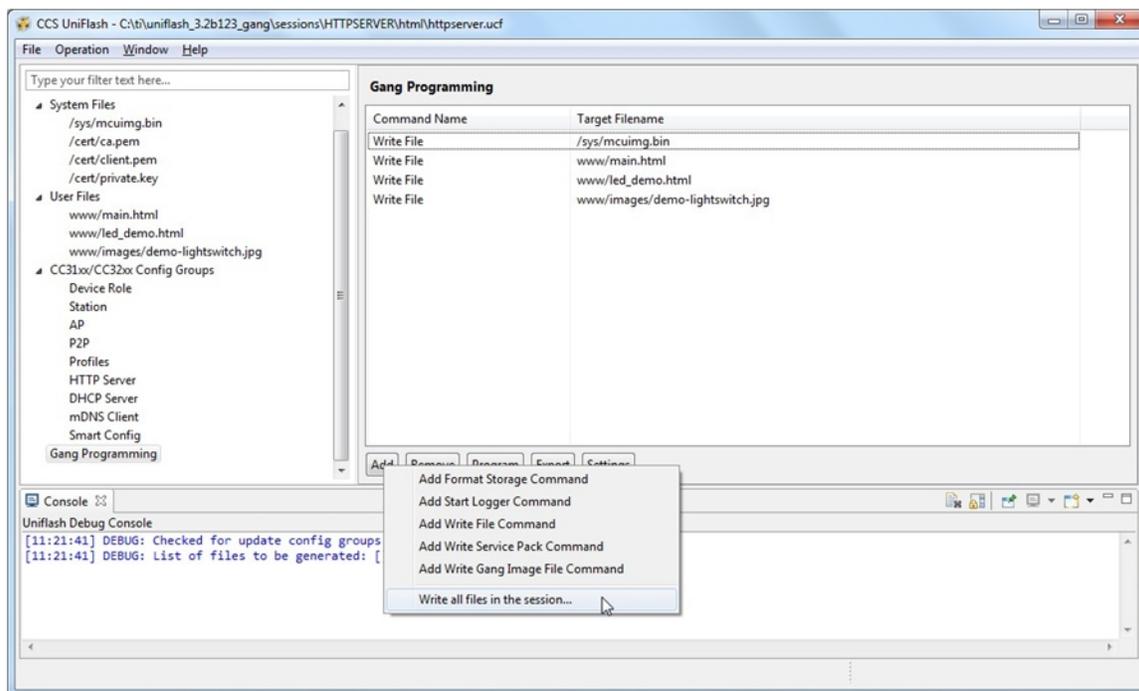


Figure 47. Programming GUI Flow

The recommended steps to build the image:

1. Build a session and include the following files:
  - a. System files: by configuring any system file from the System Files group.
  - b. Configuration files: by configuring the desired Configuration Groups and checking their respective Update checkbox.
  - c. User files: by manually adding files via the Add File button.
2. Save the session.
3. Go to Gang Programming group.
4. Import the session into the Gang Programming window by clicking AddM Write all files in the session option and then pick the \*.usf session you created. All files should be imported into the window.



**Figure 48. Import Session**

- Click on Settings button and configure the Storage Capacity Bytes from the drop down window.

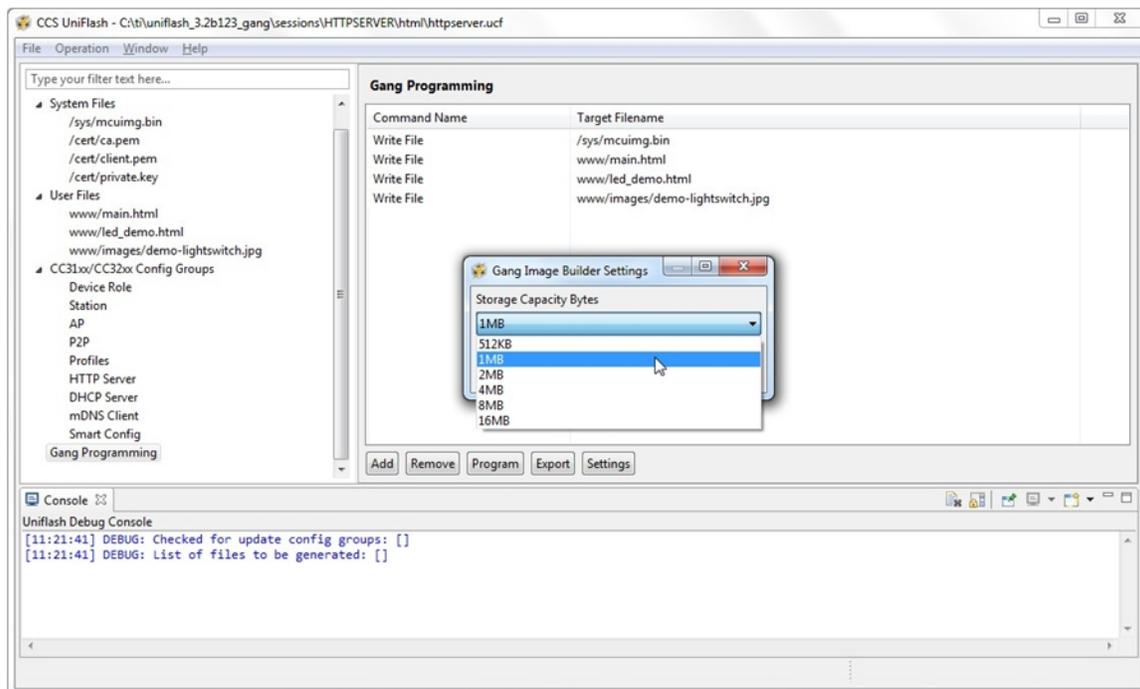


Figure 49. General Settings

- Add a format command by clicking the Add → Add Format Storage Command. This step is mandatory as it creates the FAT table.

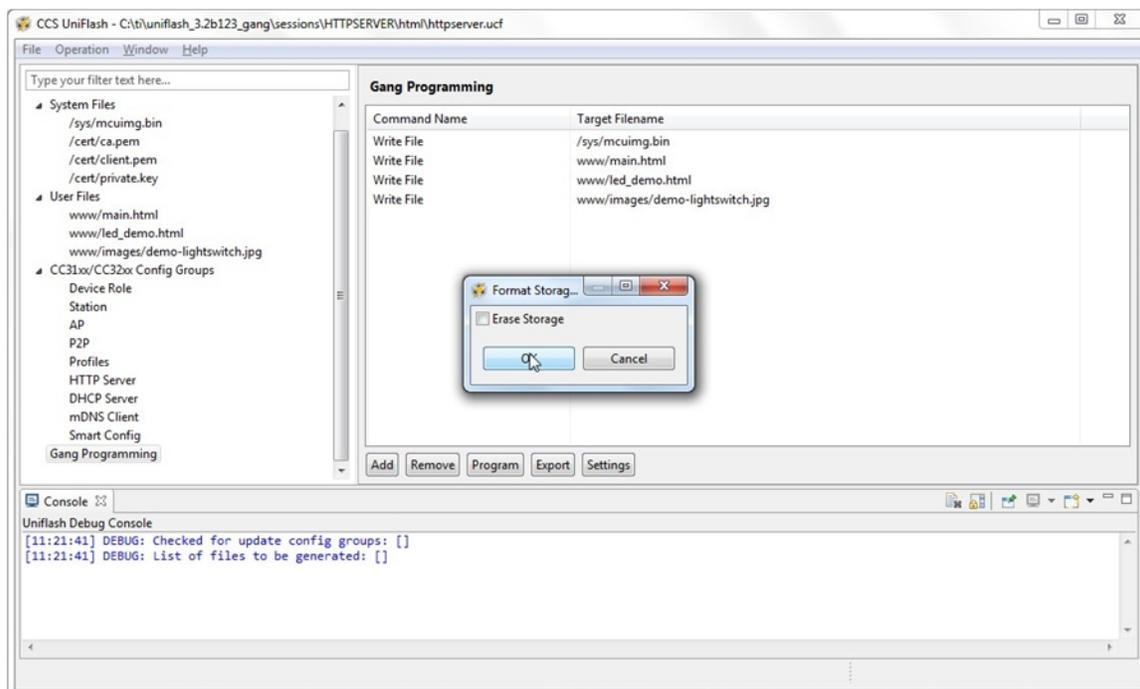


Figure 50. Add Format

- It is optional to add logging during image extraction. It can be done by clicking the Add → Add Start Logger Command. It is recommended mainly in development phase and not during production line.

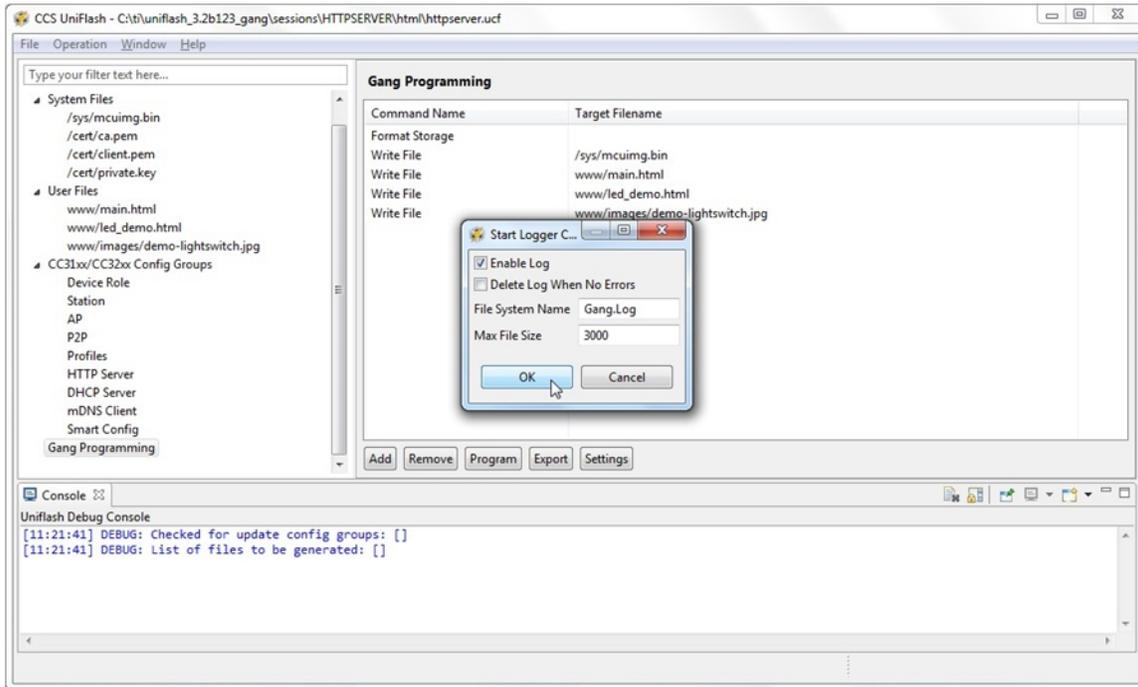


Figure 51. Add Logger

- Add the Service Pack. This step is also mandatory as the device gets formatted and the latest Service Pack should be used. It can be done by clicking the Add → Add Write Service Pack Command.

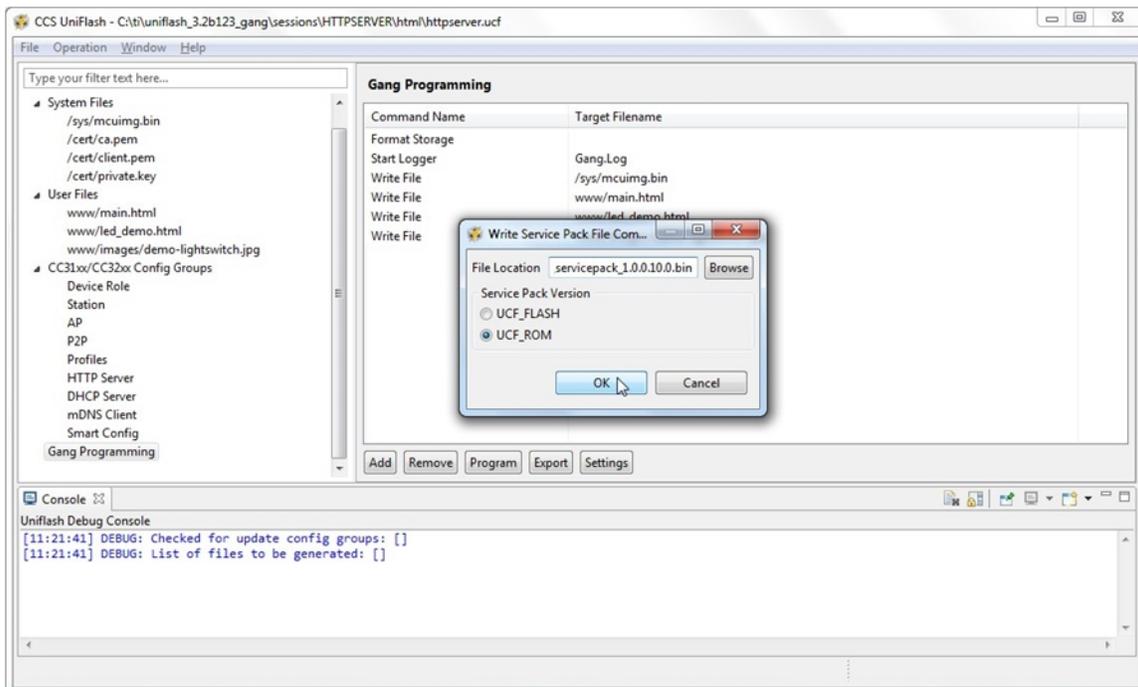


Figure 52. Add ServicePack

9. Export to create the binary image or Program to create and program the binary image.

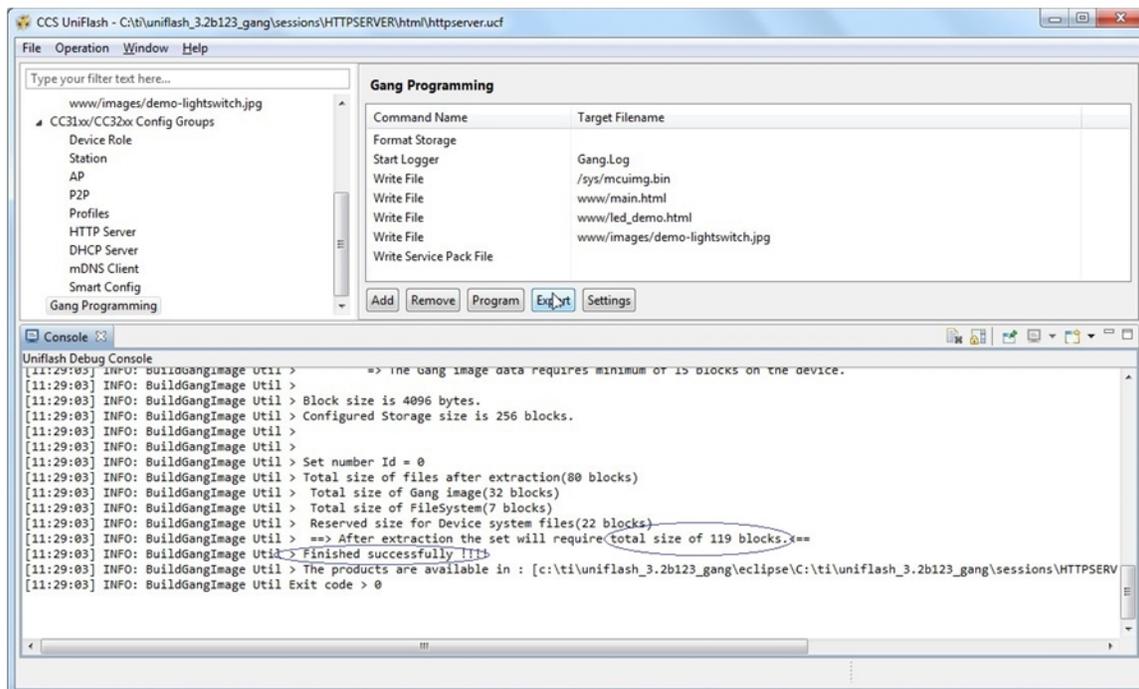


Figure 53. Export

10. Make sure you get no errors on the Console window. You should get the Finished Successfully message and the total amount of required blocks as shown in Figure 53.

11. Reset the device. Upon initialization, the device extracts the binary image and executes the command, one-by-one. If the procedure is successful, SimpleLink device sends the initialization complete event.

## 11.7 Serial Flash usage and Design

### 11.7.1 Overview

Since Image Programming is the preferred choice for production line, it is highly important to understand the guidelines and memory usage of the file system so users would be able to design their system wisely. Some of the information is device generic and some is Image Programming specific.

### 11.7.2 File System Guidelines

The guidelines are device generic and not related specifically to the Image Creation and Programming. These guidelines are applicable also while programming via Uniflash or MCU host.

- The File System by itself requires 12KB (3 blocks)
- Each file as a minimum consume:
  - One block (4KB) for file with no fail safe support
  - Two blocks (8KB) for file with fail safe support (twice the original size)
- Two blocks (8KB) for file with fail safe support (twice the original size)
- The most demanding system file in terms of space is the ServicePack file
- It is possible to set the maximum size attribute provided upon file creation (the file system reserves space). In this case, the actual size of the file is irrelevant as file system occupies the allocated space.
- There is no fragmentation in the file system. It means that an existing file which is removed would leave a "hole" in memory. This "hole" may be reallocated by the file system if the new file can fit into it.

- Each file includes a header which enlarges the actual size of the file. The overhead is 440 bytes. If the file size is very close to a multiple of block (4KB), additional block may be allocated. For example, a file of actual size 4000 bytes would result in allocating 2 blocks as  $4000+440 > 4096$ .

### 11.7.3 Memory Usage

#### 11.7.3.1 Device Generic

Following is a file system breakdown:

- File System FAT: 3 blocks
- Reserved blocks: 2 blocks
- ServicePack: 66 blocks
- System files: 22 blocks
- Configuration files: 28 blocks

Table 3 describes all configuration files.

**Table 3. Configuration Files**

Filename	Description	Fail-Safe	#Blobks
/sys/macadd.bin	MAC address	true	2
/sys/stacfg.ini	Station configuration	true	2
/sys/ap.cfg	AP configuration	true	2
/sys/p2p.cfg	Peer2Peer configuration	true	2
/sys/mode.cfg	WLAN Mode	true	2
/sys/pref.net	Preferred networks	true	4
/sys/pmcfp.ini	Power Management	true	2
/sys/ipcfp.ini	IP configuration	true	2
/sys/devname.cfg	Device Name	true	2
/sys/dhcpsrv.cfg	DHCP Server configuration	true	2
/sys/httpsrv.cfg	HTTP Server configuration	true	2
/sys/mdns.cfg	mDNS configuration	true	2
/sys/smartconfigkeys.cfg	SmartConfig Keys	true	2
<b>TOTAL</b>			<b>28</b>

It is highly important to realize that not all system and configuration files are mandatory for a proper device operation. It is use case dependent. While it is easier to understand whether a configuration file is eventually used or not, it is less obvious with system files. This is the reason why it is mandatory to keep the entire 22 blocks space for system files. Actually, this space is allocated as part of the Image creation procedure and cannot be altered by the user.

#### 11.7.3.2 Image Specific

The image includes not only the file content but also some metadata that is required during image extraction and programming. The following are guidelines that apply for image creation only:

- The Image resides on the serial flash as well. The memory occupied by the image cannot be reused during image extraction. The memory space can only be used after the image is extracted (so it may be seen as reserved memory allocation).
- It is the responsibility of the user to reserve memory for any configuration file. If possible, it is highly advised to save the entire 28 blocks.
- It is the responsibility of the user to reserve memory for any user file

## 11.8 Image Creation – Troubleshooting

If you got to this section, it means something in the process went wrong. Before debugging Gang Programming, please make sure you turn on verbose mode for maximum debugging information. Pick Window → Preferences from the top toolbar. Under Uniflash Preferences, the Print out additional debug information for the supported modules should be checked for full debug messages.

The following checklist includes the main nodes in the process and expected results:

- Image creation: upon a successful creation, a Finished successfully!!!! message should appear on the Console window with some additional debug messages
- Image Programming
  - Using any off-the-shelf SPI programmer (mainly before assembly on the target boards): a successful programming is tool specific.
  - Using Uniflash (via UART lines): upon a successful programming, Uniflash should not print any error messages on the Console window.
- Image extraction: upon resetting the board, an indication from the Host MCU for successful power-up should be generated.
- Post extraction: after image extraction is completed, user application should be executed with no issues (as if files were programmed separately).

Possible issues at each node and respective solutions:

- Image creation
  - same filename is used
    - You should see an exclamation mark indicating this error. Please avoid using the same filename more than once.
  - Total files exceed the size of the serial flash:
    - Try to use 0 as Max file size so the minimum size is allocated (original file size)
    - Make sure you do not add the Write Gang Image File command. It reduces the size of the image
    - Try to remove some unnecessary files is possible
  - Any other error:
    - Capture the error code on the Console window. The error code is the number following the 'Exit code >' prefix.
- Image Programming:
  - Using any off-the-shelf SPI programmer:
    - Make sure all SPI lines are connected properly
    - Make sure either raw binary (\*.bin) or IntelHex binary (\*.hex) are used properly with the tool
    - Read back image and verify
  - Using Uniflash:
    - Make sure you use the raw binary (\*.bin) and not the IntelHex binary (\*.hex)
    - Uniflash should not print any error messages on the Console window. For any error, please open an E2E ticket.
- Image extraction:
  - Serial flash is not formatted:
    - Make sure Format Storage command is added to the image
  - Any other error:
    - If Start Logger command is added, try to raw read the entire content using external tools and search for the first text line of the log: "Start Log" or 617274204C6F67 in hexadecimal. Open an E2E and attach the capture
    - For help, open an E2E ticket in all other cases.

- Post extraction
  - Experiencing issues when invoking APIs that implicitly creates files on the serial flash. These APIs creates configuration files.
  - It is possible that the free space left on the serial flash after extraction is not enough. Please try to free some space on the original image and retry
  - For help, open an E2E ticket in all other cases.

## 12 Troubleshoot/Debugging

Debug messages are printed on the 'Console' screen. It is possible to control the verbose level of these printouts from the 'Window → Preferences'. Under 'Uniflash Preferences', the 'Print out additional debug information for the supported modules' should be checked for full debug messages.

By default, this mode is enabled.

The following is a list of common Uniflash behaviors that might be perceived as erroneous but are actually not:

- Pressing 'cancel' during the program operation. In this case, the progress bar on the message box is frozen and the Uniflash seems stuck. However, the Uniflash continues flashing the current file and when it is done, the message box is released. This is desired since there is no point in canceling during flashing as it may cause unexpected behavior in the device. In addition, if debugging information is enabled, the flashing messages can be seen on the Console screen.
- Connecting to the device is applied on every operation (Program, Format, Get Version). When 'cancel' button is pressed during connection, the message box seems stuck. However, since the Uniflash tries to interact with the target device over UART and the user does not reset the board, it can take up to UART timeout for the cancel operation to take effect. UART timeout is 15 seconds. When the time elapses, the message box is released.

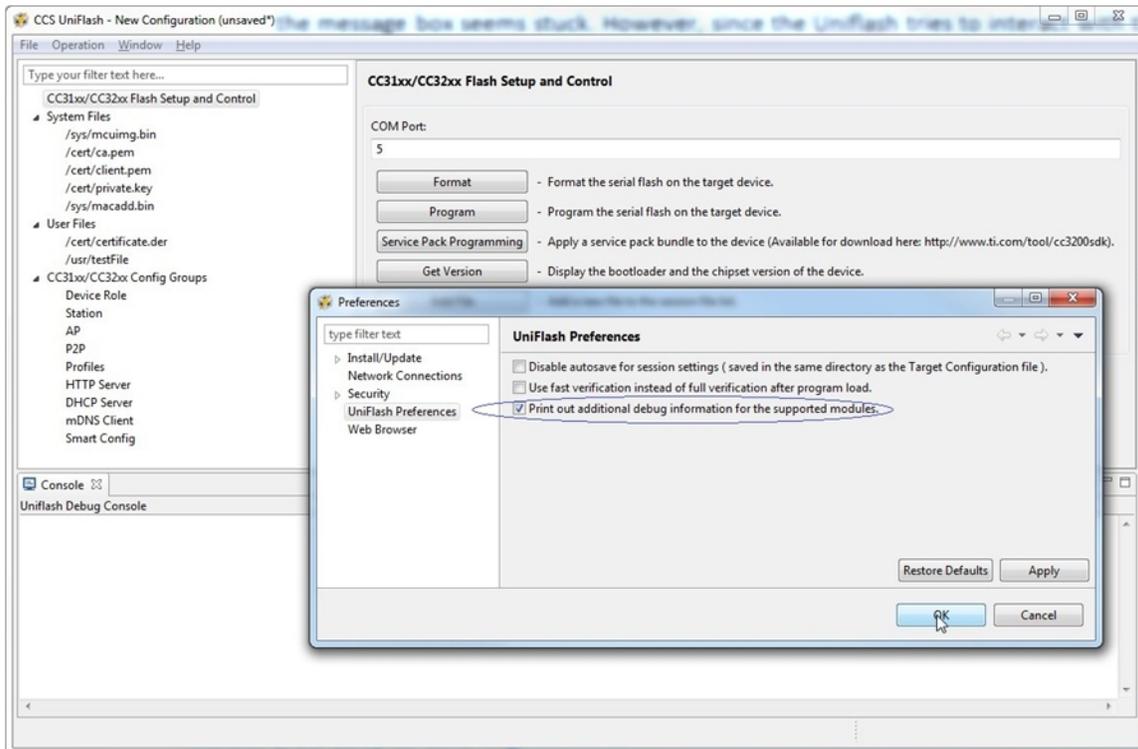


Figure 54. Debugging

## 13 Limitations

- File removal: few notes regarding this option:
  - Since no 'directory' option exists (reading the files list from the device), there is no way that the user can know what files are already flashed into the device
  - File removal is harmless in case the file does not exist on the device
  - The limitation (that might be confusing): after deletion is completed, the file is still reflected in the GUI.

```
{{#if: |
|}}
```

---

**NOTE:** `{{#if: |}}` it is possible to remove it from the GUI as well as from the device. For more information, see [Section 7.9](#).

---

- Storage Format: it is the user responsibility to know the capacity of the serial flash and format it accordingly
- UART connection: only 921600bps is supported
- Linux related:
  - No automatic reset of the connected target is supported (unlike Windows version)
  - Using 'Verify' option (for files or image) results in much slower content reading than the Windows version
  - It has been observed occasionally that communication with the target device is halted. It mainly happens during large content reading (with 'Verify' flag checked)

## 14 References

- Texas Instruments: [CC3100/CC3200 SimpleLink™ Wi-Fi® Internet-on-a-Chip User's Guide](#)

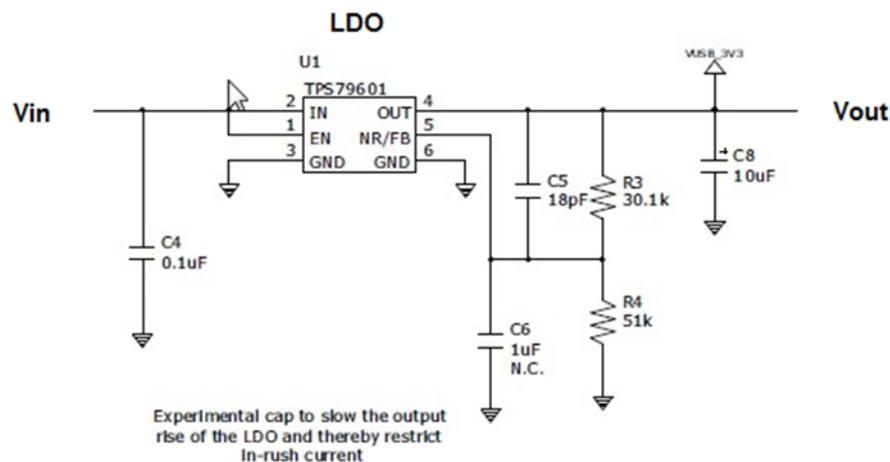
## UART Connection

### A.1 UART Connection

Connection to Simple Link CC3x00 devices is applied via standard UART lines. Evaluation boards (either Booster Pack for CC3100 or Launch Pad for CC3200) expose the UART line as VCP via FTDI chipset. FTDI drivers should be installed prior to working with the evaluation board. These drivers are part of the Uniflash installation. It is also possible to work directly with the UART lines.

Regarding the voltage level on the UART lines, it should match the voltage supplied to the device (any IO line is tight to VBAT). Note that an external level shifter may be required if the voltage on the UART line is different than VBAT.

For example, in the Boosterpack evaluation board, an LDO is used to drop the 5 V coming from the USB to 3.3 V ( $V_{ref} \cdot (1 + R2/R1) = 1.2246 \text{ V} \cdot (1 + 51\text{K}/30.1\text{K}) = 3.3 \text{ V}$ ). In this mode, all IO lines are tight to 3.3 V and the board works in a pre-regulated 3.3 V scheme. In this case, the UART lines used for the tool need to work in 3.3 V as well.



**Figure 55. UART Connection**

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on [ti.com](http://ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2022, Texas Instruments Incorporated